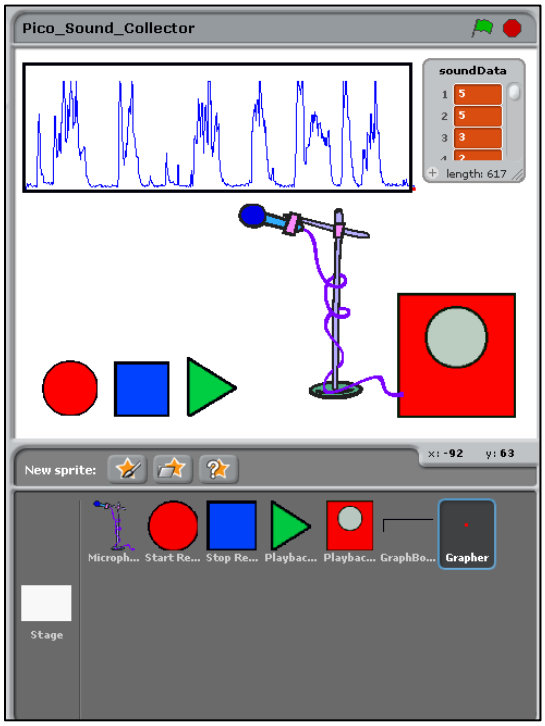
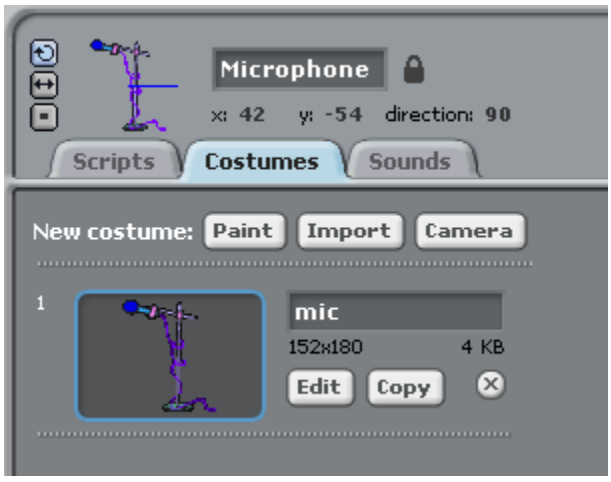
Handout: Directions for Creating the Pico Sound Collector

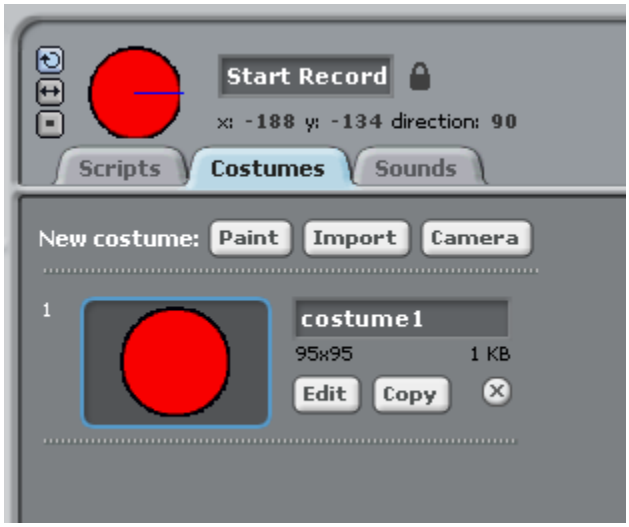


Process:

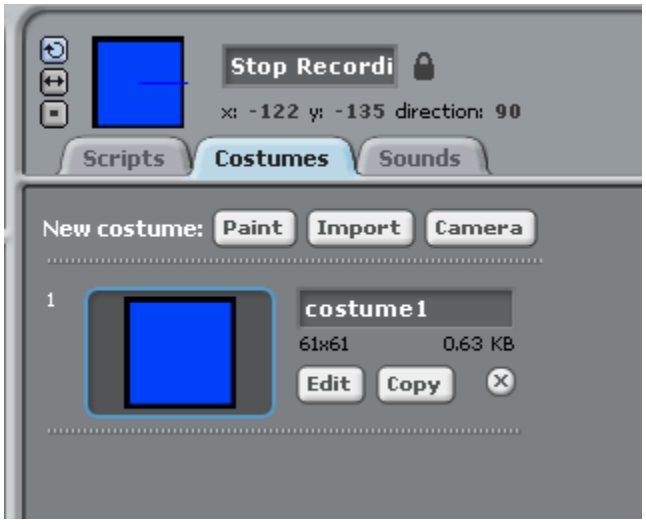
1. Start Scratch and Connect the PicoBoard to your computer via the USB Port.
2. Save the Scratch Project as “Pico_Sound_Collector”
3. Create the following Sprites
    a. Microphone -> Will “Listen” to Pico Sound Sensor and record Data in a List



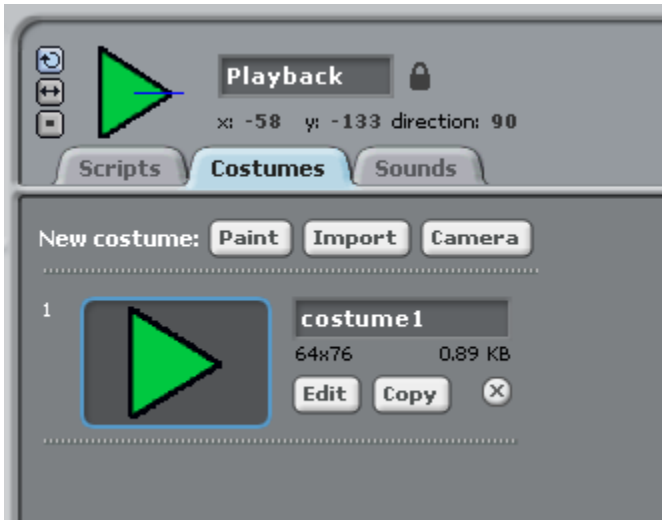
b. StartRecording -> Triggers the Broadcast that starts the recording loop



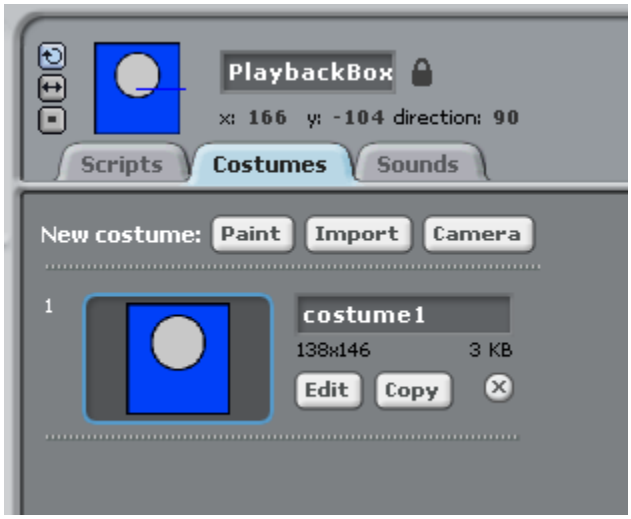
c. StopRecording -> Triggers the Broadcast to stop the recording loop



d. Playback -> Triggers the Broadcast to start the playback loop.

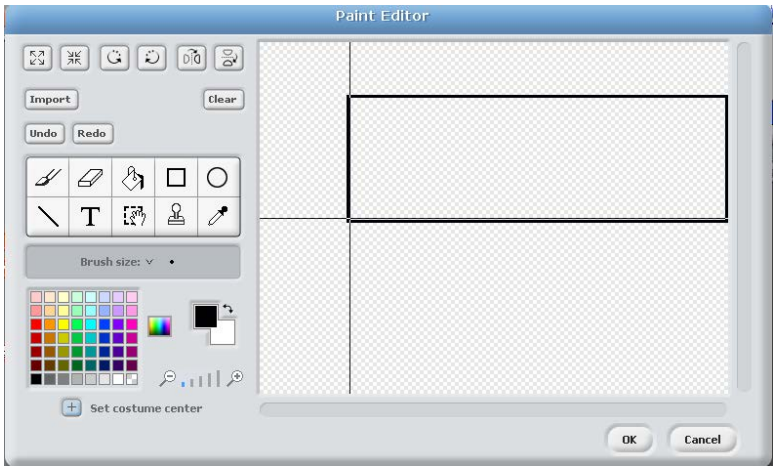


e. PlaybackBox -> Performs the action of iterating through the soundData List and playing Percussion and Pitch Sounds

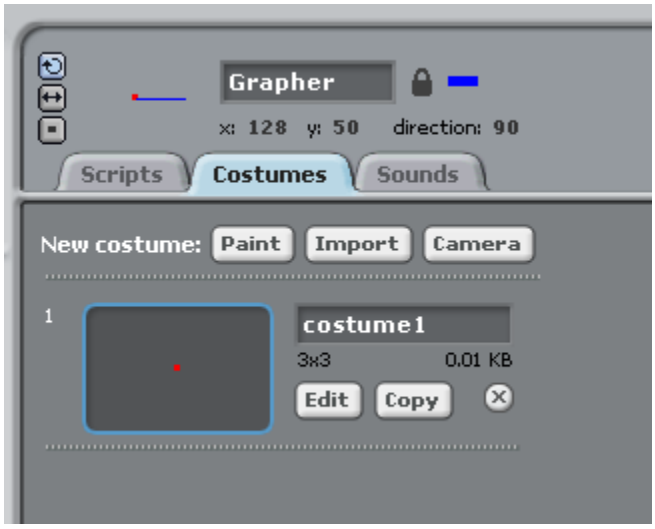


f. GraphBox -> Acts as background for the graph

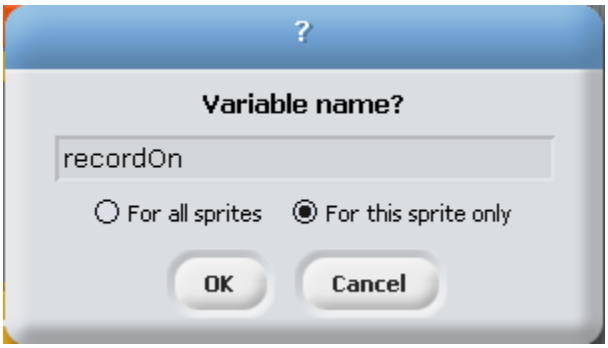
For the GraphBox, make sure the costume center is in the lower left corner.



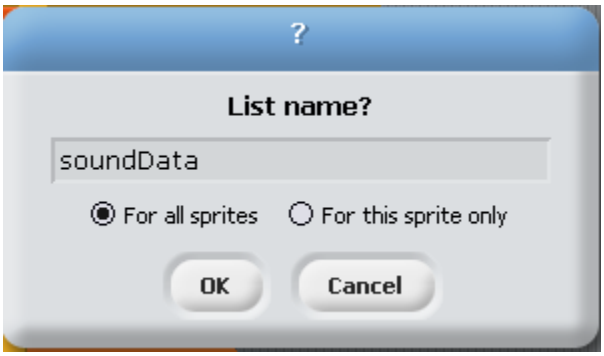
g. Grapher -> A single Dot that will move and create a line by iterating through the soundData list.



4. Go to the Microphone Sprite and Click on Variables.
5. Click "Make a Variable" and name the variable 'recordOn'. Make sure that the 'For this sprite only' is checked. 'recordOn' will function as a boolean variable (True and False) that will hold the 'state' of the recording. (1 means record, 0 means stop recording)



6. Click ‘Make a list’ in Variables. Name the list ‘soundData’ and make sure that ‘For all sprites’ is clicked. This will be a public List that will hold the Data from the recorded Sound amplitude from the PicoBoard.



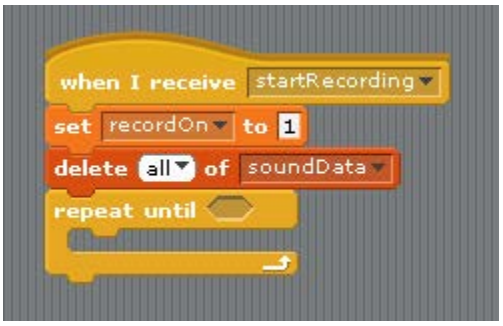
7. We will now build the Scripts for the Microphone. For this project we will use the ‘Broadcast’ and ‘when I receive’ as message events through which the Sprites will pass directives and control the flow of the program across objects. Drag a ‘when I receive’ into the Scripts area of the Microphone Sprite.

8. Select ‘new’ from the ‘when I receive’ block and type ‘StartRecording’ and click OK.

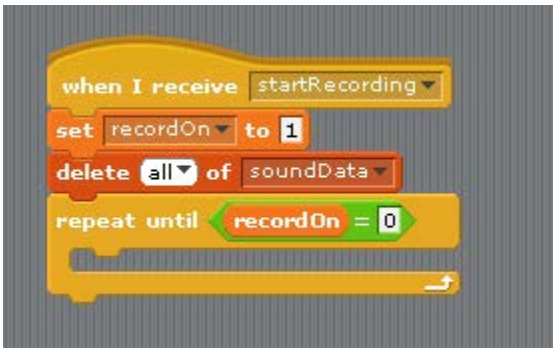
9. Next we will initialize the recordOn variable with a value. Drag a set ‘recordOn’ to 1 block and connect it to the ‘when I receive startRecording’

10. We also need to initialize the soundData list object by clearing all values. Drag and connect a ‘delete all of soundData’ block to the stack.

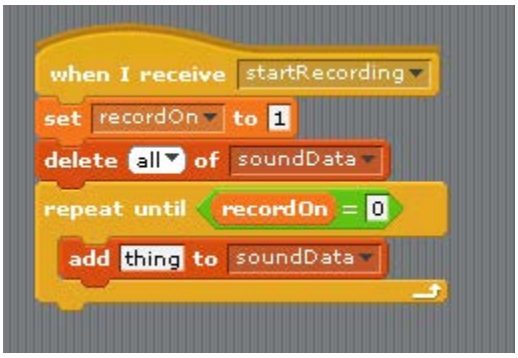
11. A ‘repeat until’ block follows setting up a loop that will place the reading from the PicoBoard Sound Sensor (or loudness from computer microphone) into the soundData list. Place a repeat until block on the stack.



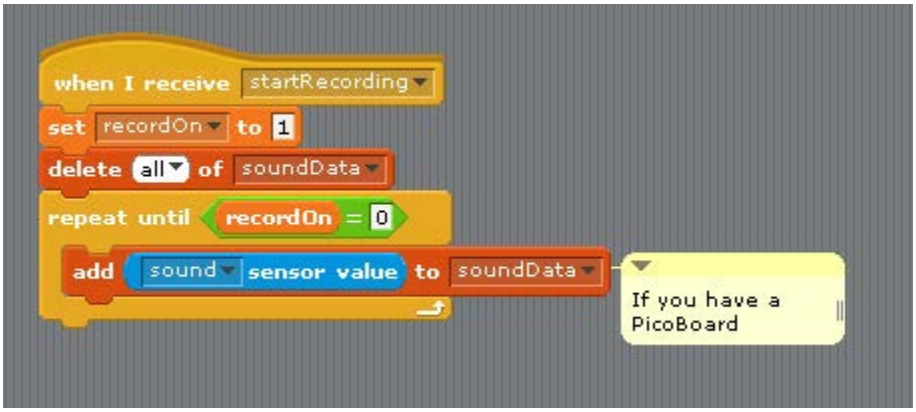
12. The repeat until block needs a stopping condition. Place the expression ‘recordOn = 0’ in the repeat until hexagon.



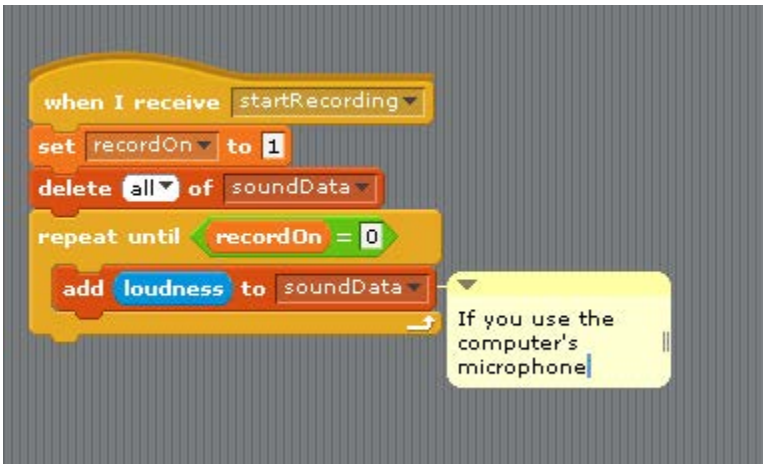
13. The core function of this script is to place the readings from the loudness or sound sensor into the soundData list. Place a ‘add thing to soundData’ block inside the ‘repeat until’ loop.



14. If you have a PicoBoard, place a ‘sound sensor value’ block in the ‘add thing’ block.

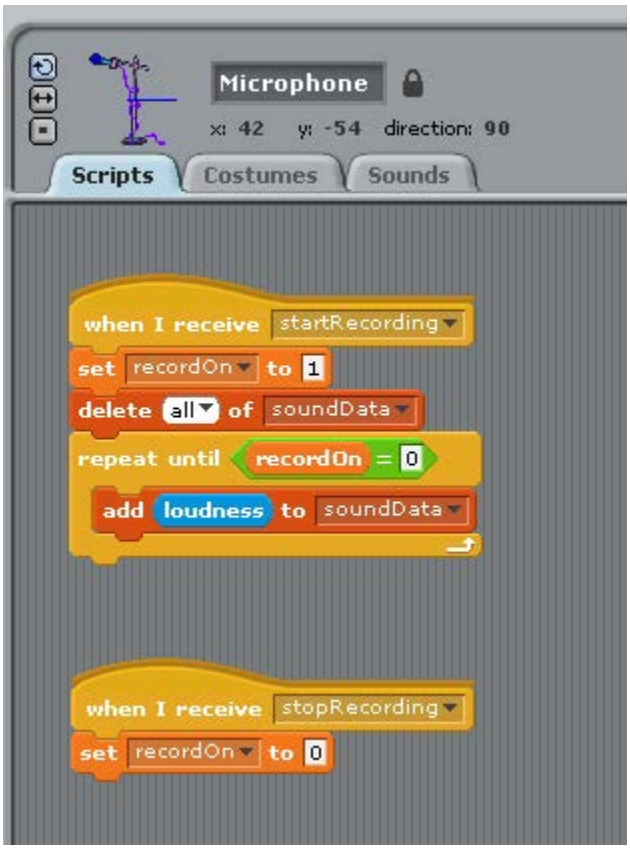


15. If you do not have a PicoBoard and want to use the computer’s microphone, use a loudness block.

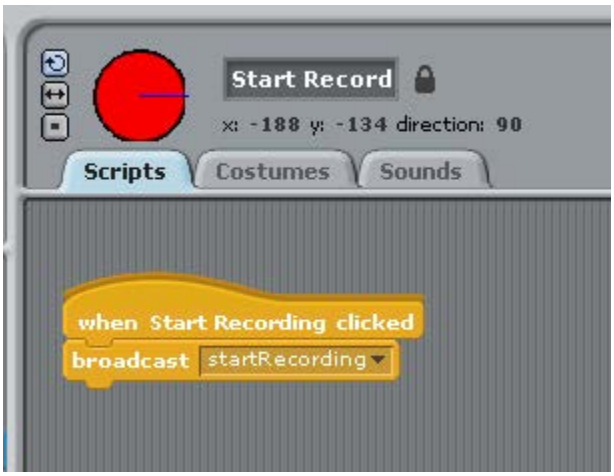


16. We need a way to set the ‘recordOn’ variable to 0 in order to exit the ‘repeat until’ loop. Add a ‘when I receive’ and ‘set recordOn to 0’ scripts. The ‘stopRecording’ will be called by a different Sprite in the program.

17. The entire Script for the Microphone should look like this:



18. Go to the Start Recording Sprite. This Sprite takes a click event and then broadcasts the message to ‘startRecording’ to the Microphone Sprite. Place a ‘when Start Recording clicked’ and a ‘broadcast startRecording’ block into the Start Recording Sprite Scripts.



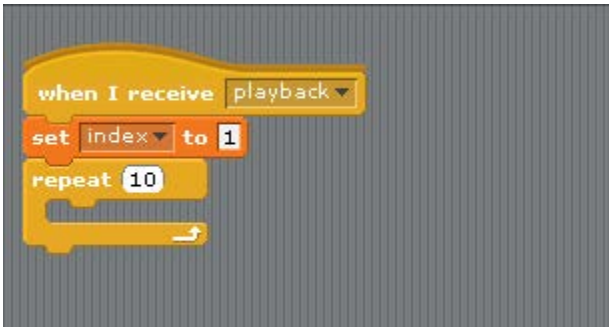
19. In a similar manner, go to the Stop Recording Sprite and place a ‘when Stop Recording clicked’ and ‘broadcast stopRecording’ block into the Scripts area.

20. The Playback Sprite will broadcast the message to start the playback.

21. Go to the PlaybackBox Sprite. This sprite will iterate through the soundData list and use each value to play a drum sound. To iterate or 'count through' a list, we will need an index to track each location of the items in the soundData list. Create a variable 'index' on PlaybackBox and make sure it is for 'this sprite only.'

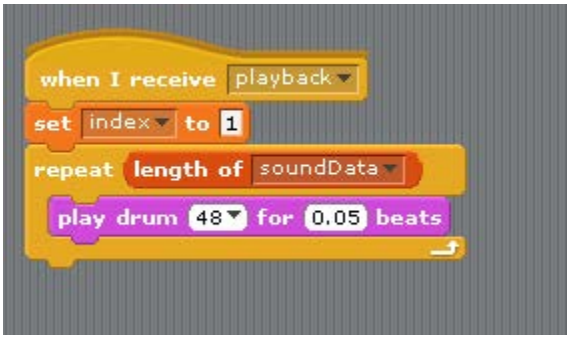
22. Place a 'when I receive playback' and a 'set index to 1' to start the playback process. The 'set index to 1' acts to initialize the index variable to the value 1. (Corresponding to the first value in the list.)

23. Place a repeat block under the ‘set index to 1’.

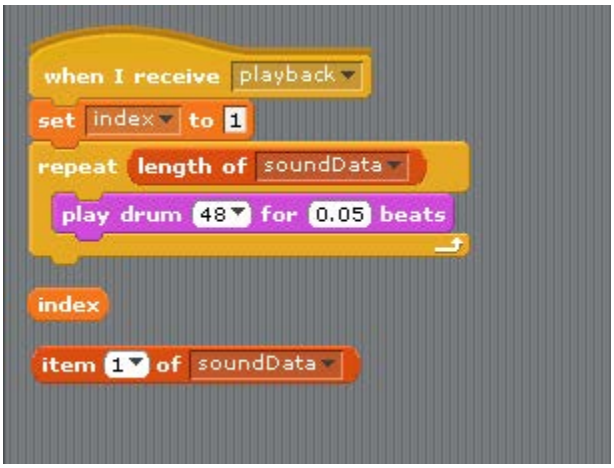


24. Since we have to move through each sound in the soundData list, place a ‘length of soundData’ block in the ‘repeat’ block parameter.

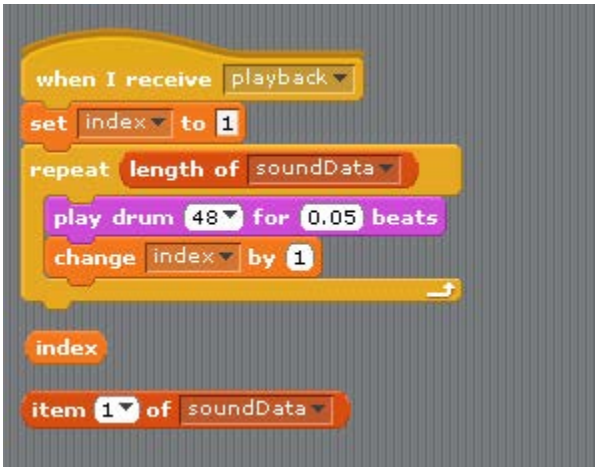
25. We will use the soundData values to play drum sounds. Place a play drum block inside the repeat and set the beats to 0.05 beats.



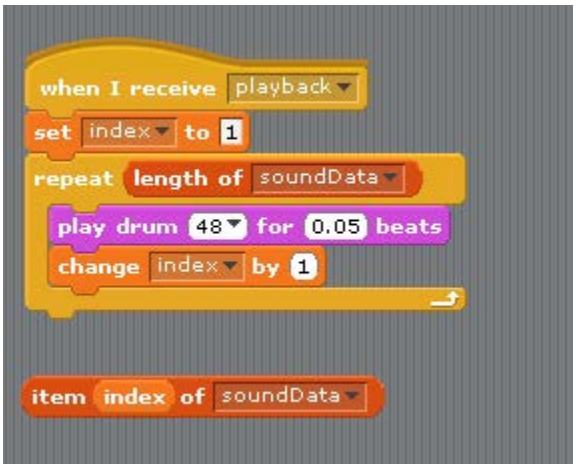
26. Here is where the awesomeness of the index variable appears! Drag an index block and an ‘item 1 of soundData’ block under the repeat block.



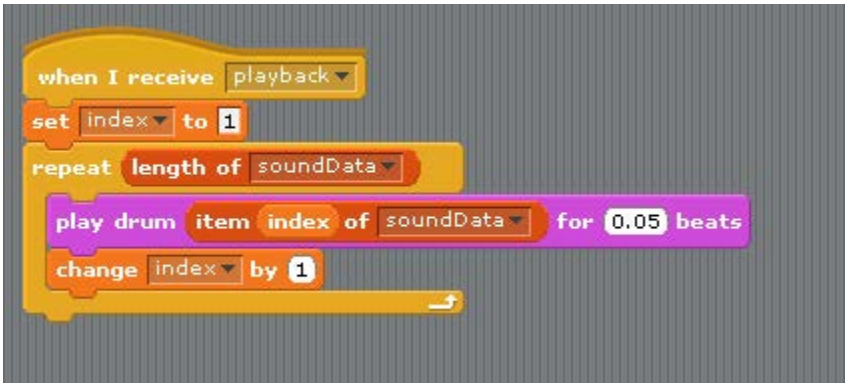
27. Place a ‘change index by 1’ block into the repeat.



28. Because the index variable will increase by 1 each time the repeat counts, we will use this increasing value to call out numbers from the soundData list in order. Place the 'index' block in the 'item 1 of soundData' block.



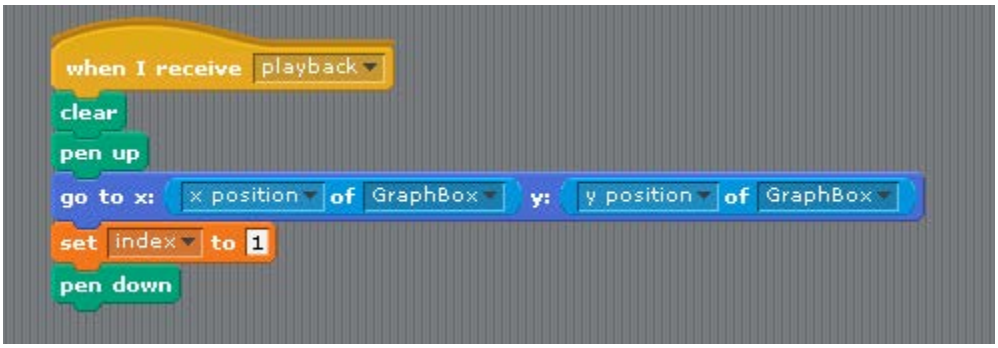
29. Place the 'item index of soundData' into the play drum block.



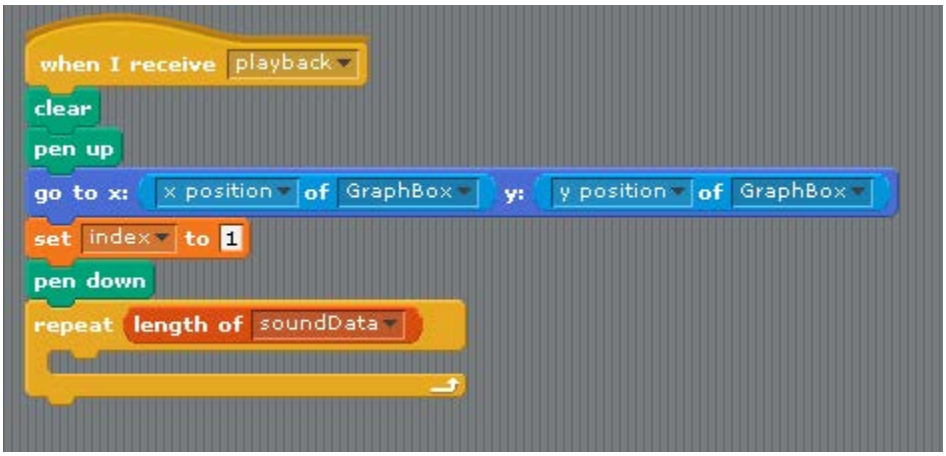
30. At this point the program will work. Click the Record Sprite and the list will increase in size as the sound values are added to the soundData list. Click the Stop Sprite to stop the recording. Click the Playback button to 'listen' to the volume data recorded in the soundData list.

31. For the Grapher Sprite, begin by creating three variables. Make sure these variables are for ‘This sprite only’.
    a. Index
    b. xPos
    c. yPos

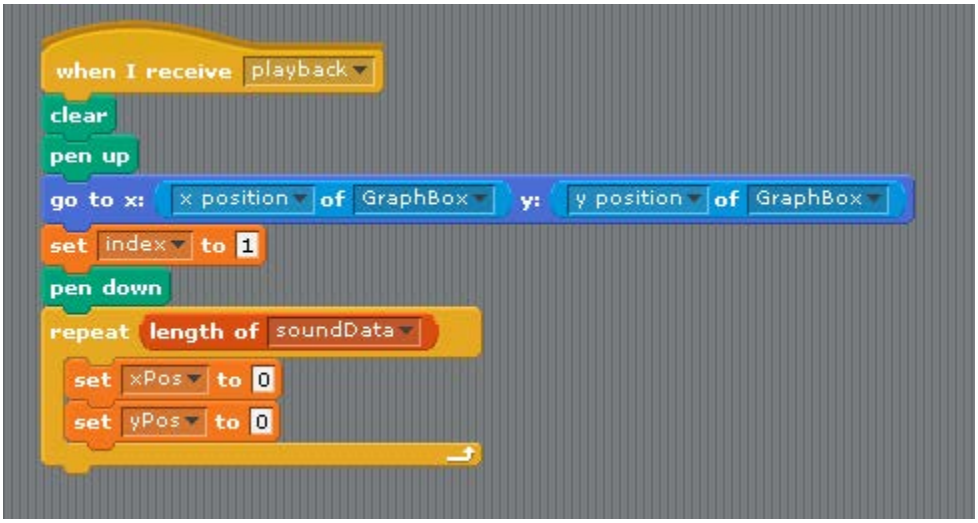
32. Put a ‘when I receive playback’ and then the blocks to initialize the Grapher Sprite.
    a. Clear any Pen
    b. Call pen up
    c. Go to the x and y position of the GraphBox
    d. Set the index variable to 1
    e. Place the Pen down



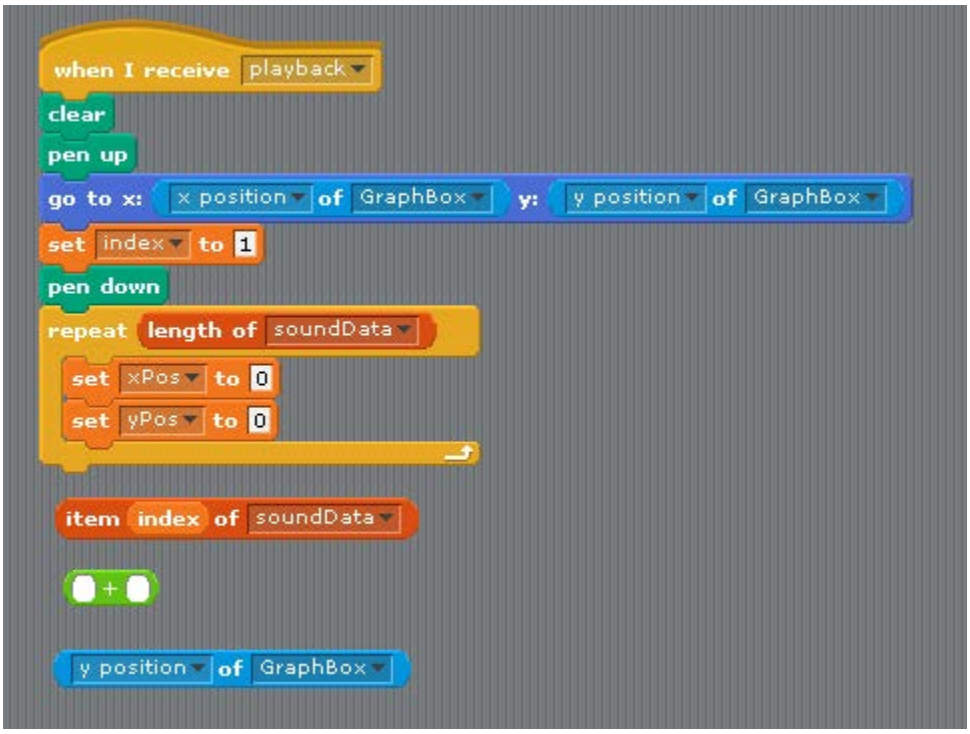
33. Place a Repeat ‘length of soundData’ under this stack. Similar to playing the sounds, we will use the soundData values to make a line graph.



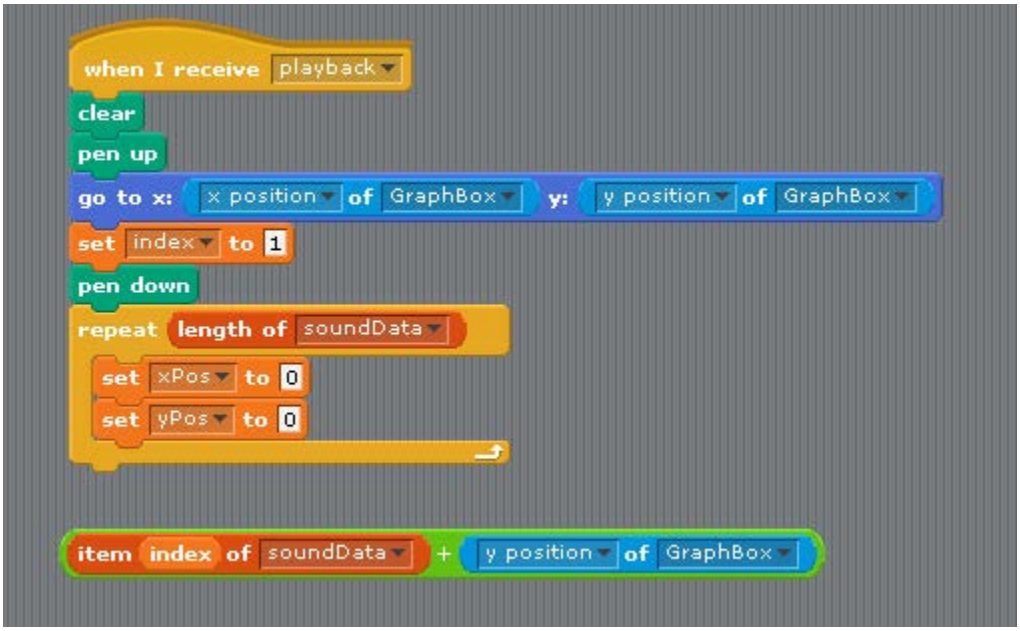
34. We now need to calculate the xPos and yPos variables that represent the x and y positions of the Grapher. Place a set xPos and a set yPos blocks into the repeat.



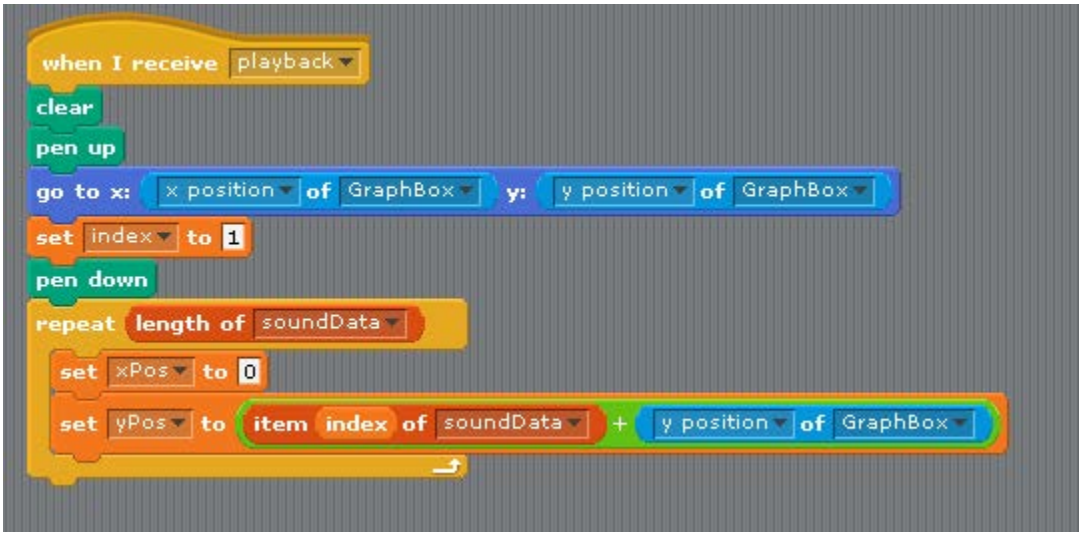
35. To calculate the yPos, place an ‘item index of soundData’, a plus operator, and a ‘y position of GraphBox’ under the repeat block.



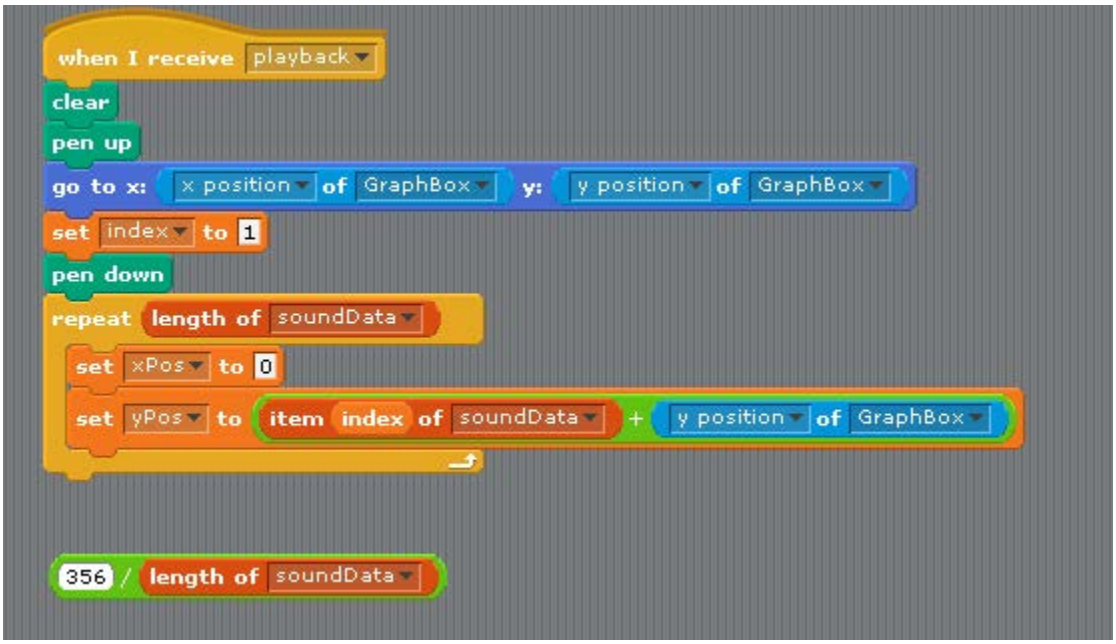
36. Place the ‘item index’ block and the ‘y position of GraphBox’ block into the plus operator.



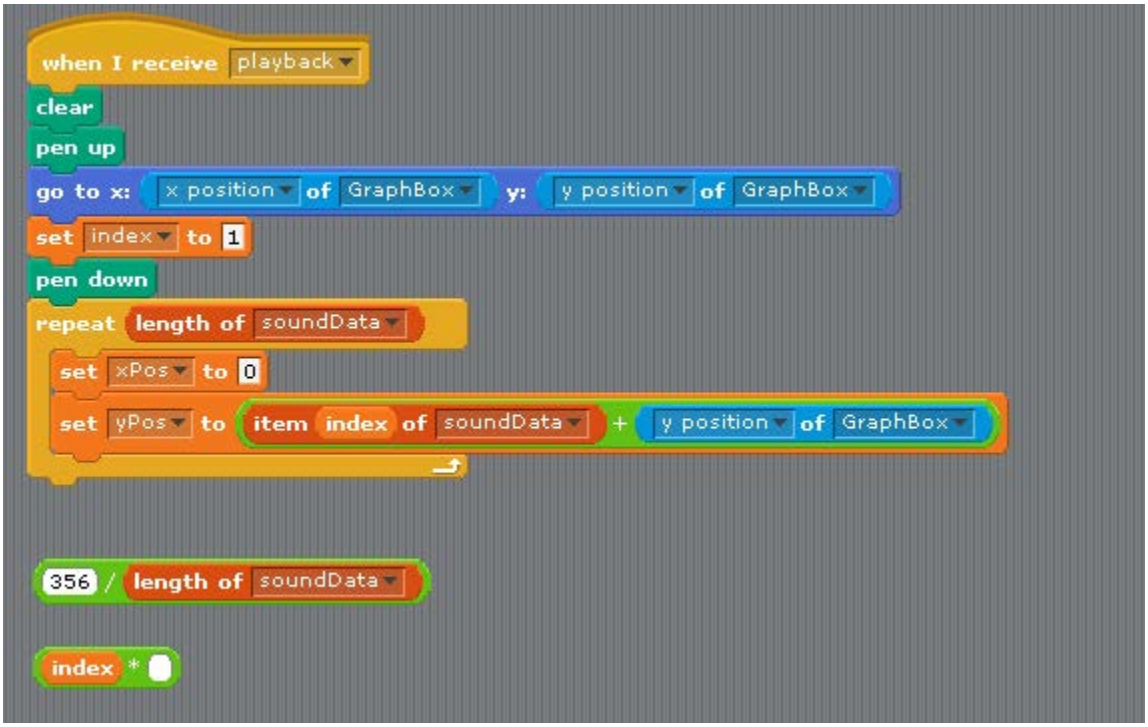
37. Place this block into the ‘set yPos’ block in the repeat.



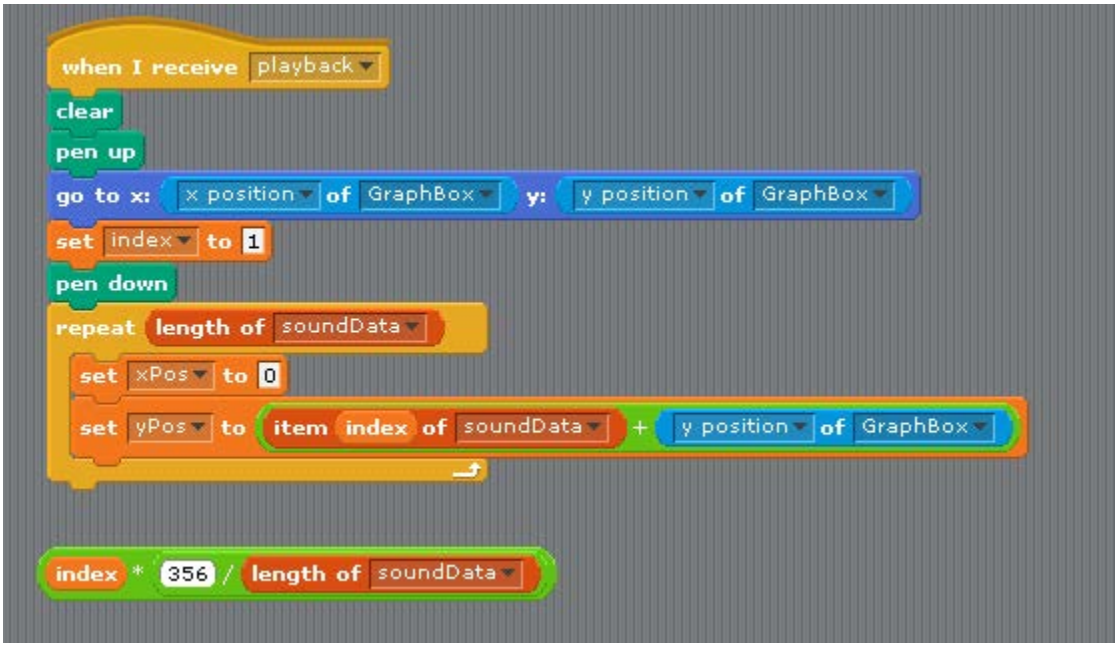
38. The above code will put the Grapher Sprite at the Y position according to the values in the soundData list. We now need to calculate the X position by advancing the Graph Sprite in reference to the length of the GraphBox. In our example the GraphBox is aobut 356 pixels long. So we will set up a ratio between the soundData List and the GraphBox. Place a divide operator and build the expression ‘356 / length of soundData’



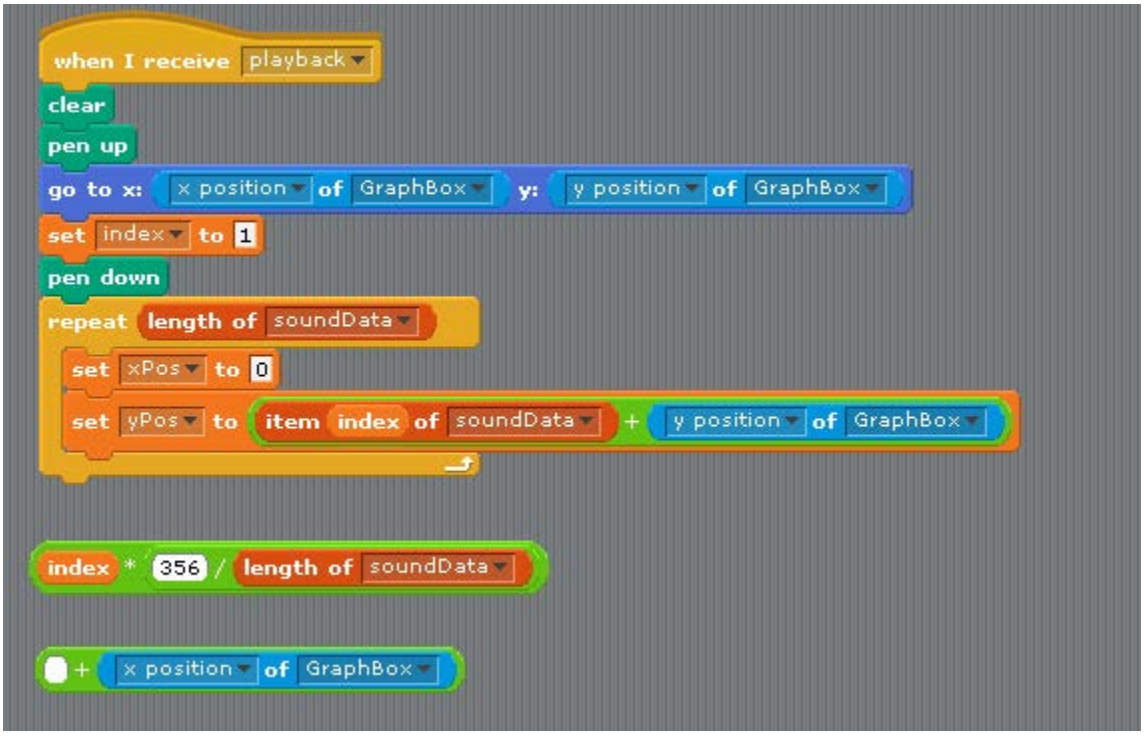
39. Place an ‘index’ variable and a multiply below the stack.



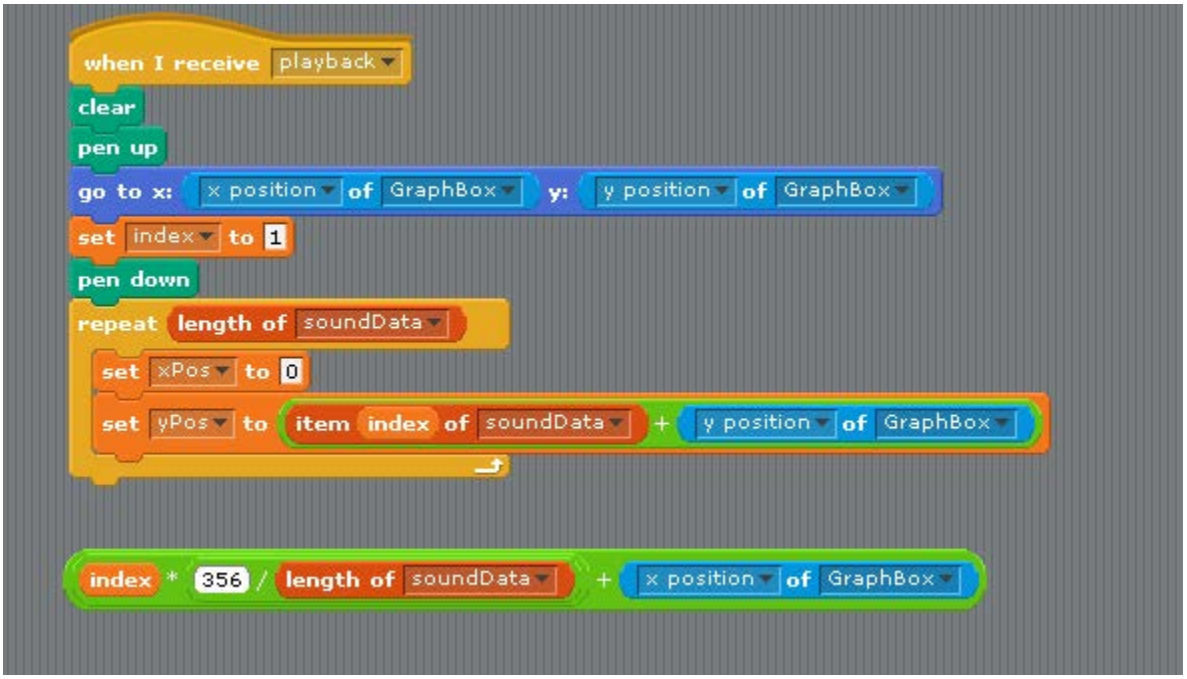
40. Place the ‘356 / length of soundData’ block into the multiply block.



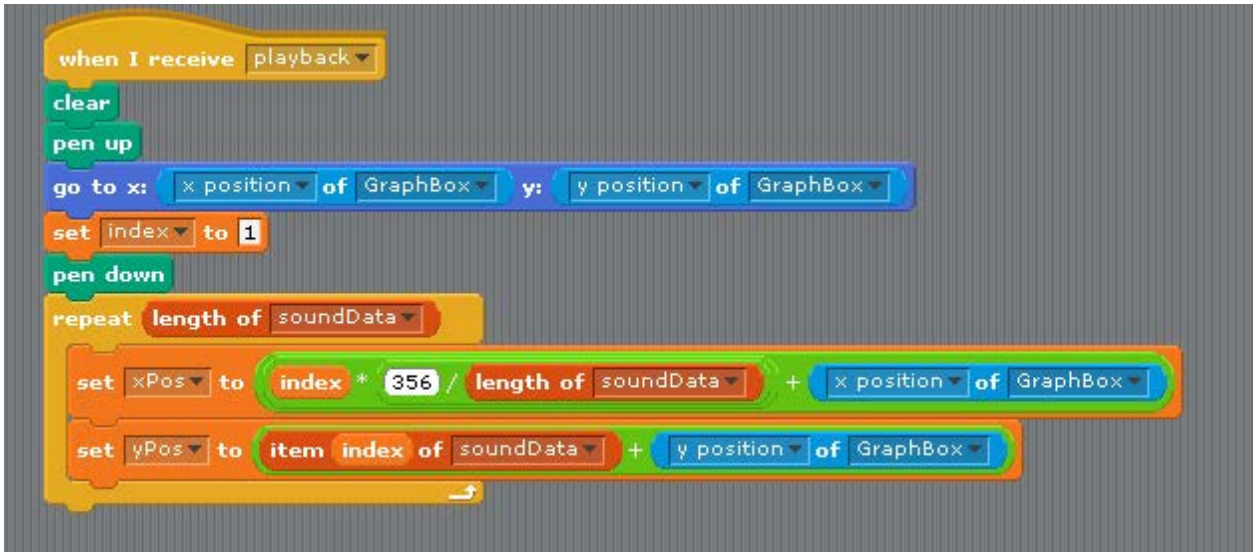
41. Place an addition operator and an ‘x position of GraphBox’ below the stack.



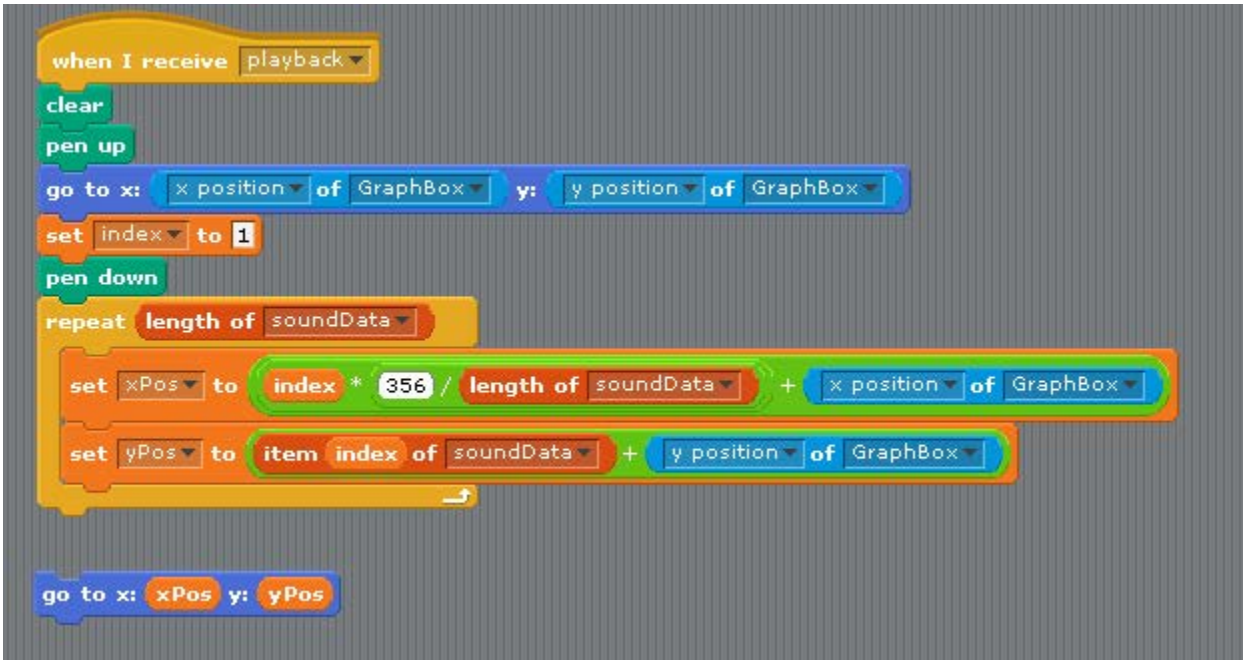
42. Place the index expression into the addition expression.



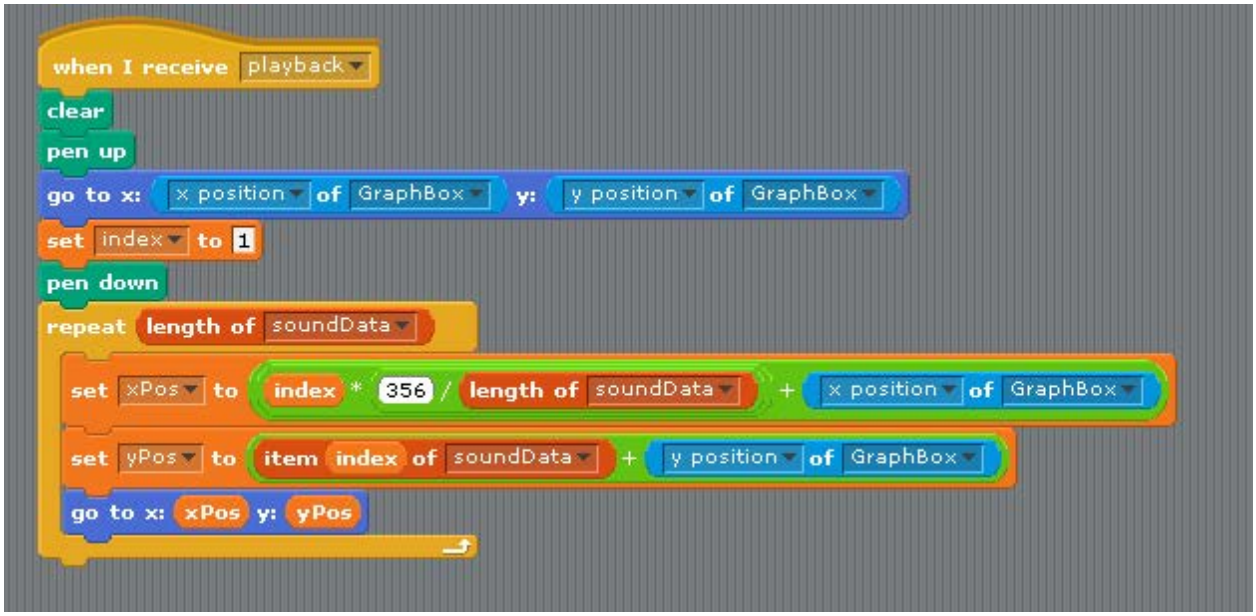
43. Place this complete expression into the set xPos block.



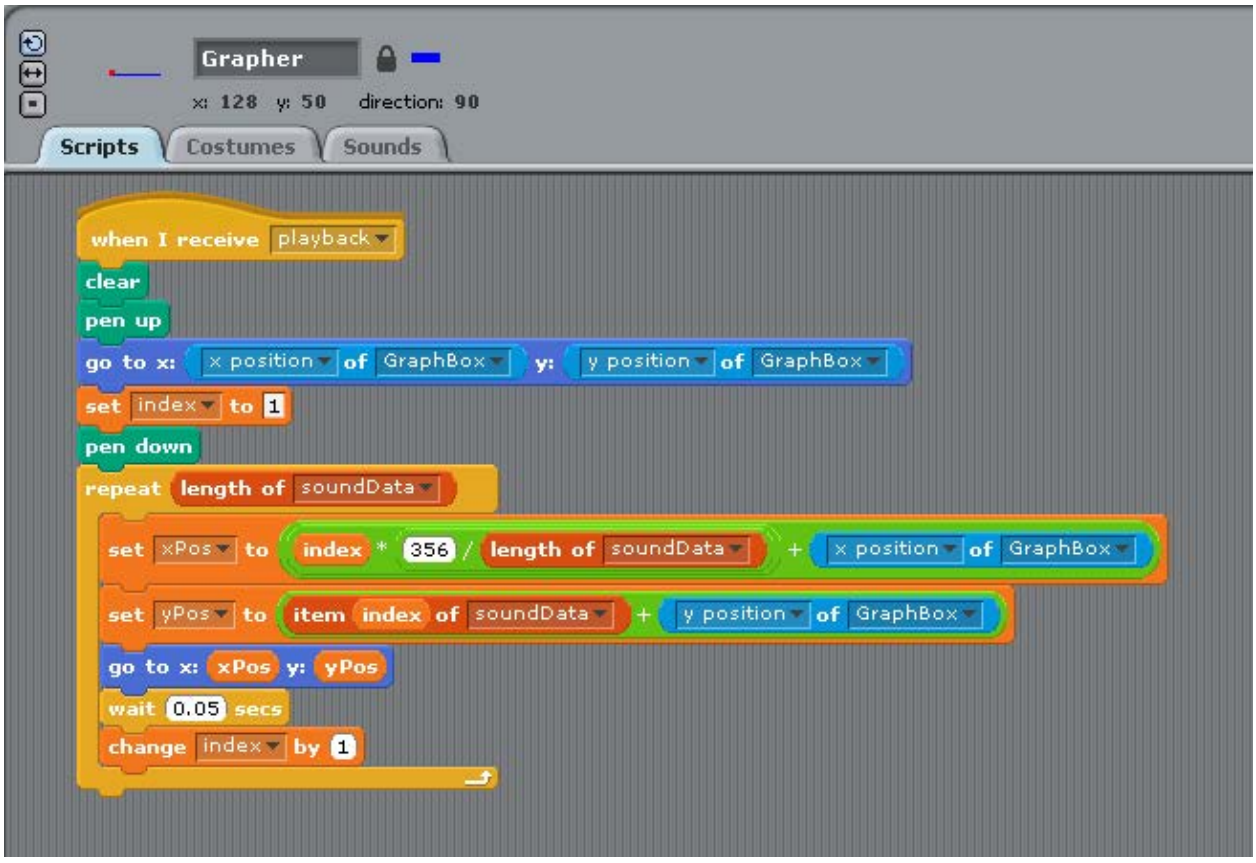
44. This represents the model of the Grapher movement. To draw this to the screen we need a ‘go to x: y: ‘ block with xPos and yPos



45. Place the ‘go to’ block into the repeat.



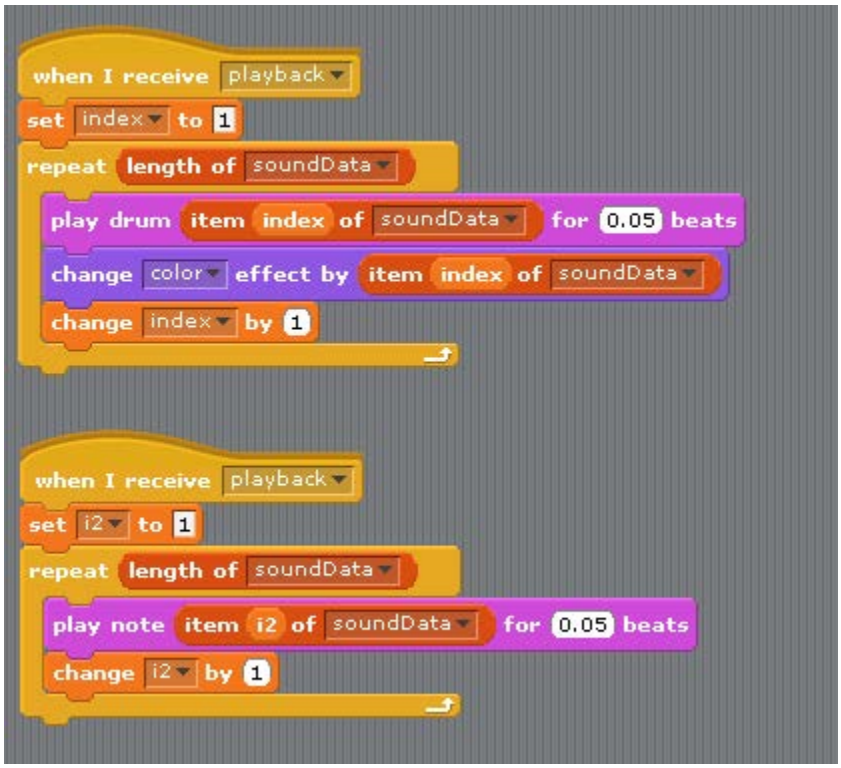
46. To keep everything timed together, place a wait 0.05 seconds and a change index by 1 into the loop. This completes the script for the Grapher

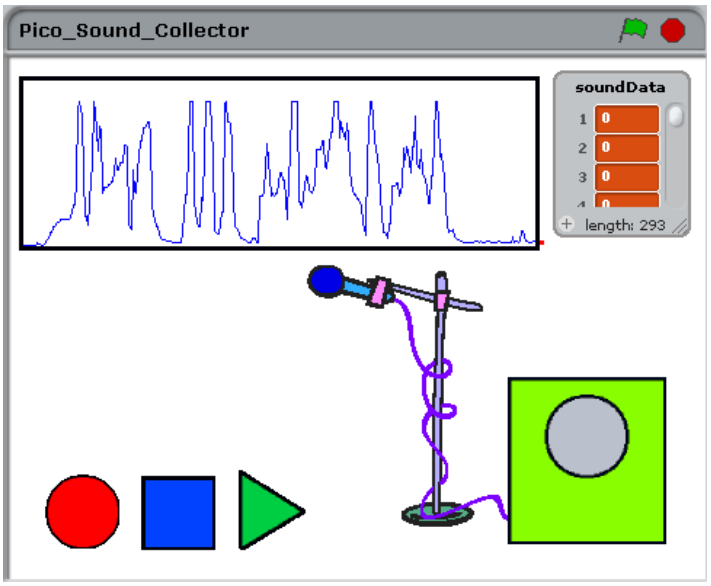


47. You are done! Run and Test the program. It sound record sound loudness and then map these values to a like graph and percussion sounds.

48. Try to map the soundData to different properties:

    a. Costume Color
    b. Costume Size
    c. Different Pitches or Durations




Pico_Sound_Collector
soundData
1 0
2 0
3 0
length: 293