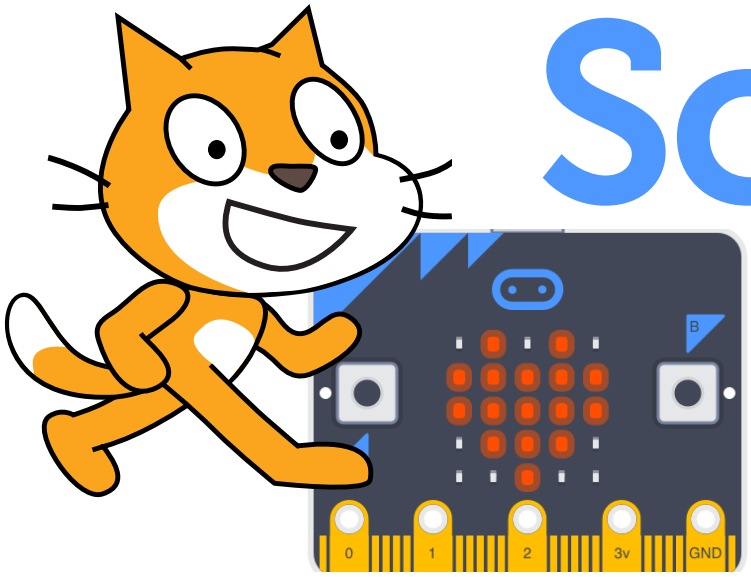


Scratch Audio Game

This game tests your focus and reaction times by giving you spoken instructions for moving the micro:bit. How many moves can you get right?

BY Sean McManus

About the Author

Sean McManus is the author of Scratch Programming in Easy Steps (now updated for Scratch 3, including a new project for micro:bit) and Cool Scratch Projects in Easy Steps.

Twitter:

@musicandwords

Web:

www.sean.co.uk/scratch

Scratch 3 introduced some great new features, including micro:bit compatibility and Text to Speech. In this project, we'll combine them to make an audio game that tells you how to move your micro:bit and times how long you take. It's a game that tests your ability to focus.

I'm assuming some experience of Scratch here. If you're a newcomer, there's a beginners' introduction on my website at www.sean.co.uk/scratch.

Step 1: Set up the micro:bit

We explained how to set up your micro:bit in micro:mag #2 (see pages 22-24). Those instructions refer to the beta version, but you can use the normal version of Scratch now. If you don't have issue 2 handy, follow the instructions on the Scratch website at https://scratch.mit.edu/microbit.

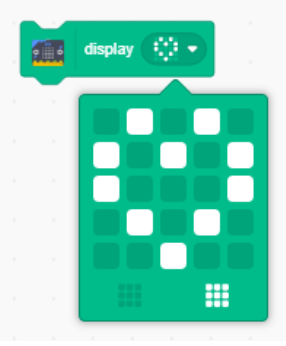
Step 2: Add the sound effects

We'll give each movement a different sound effect. Click the Sounds tab, and then click Choose a Sound in the bottom left. Delete the Meow sound and add six short sound effects. I chose Clown Honk, Siren Whistle, Big Boing, Bite, Bonk, and Pluck. Finally, add the sounds Drum Funky and Crowd Gasp as sounds 7 and 8 on the sprite.

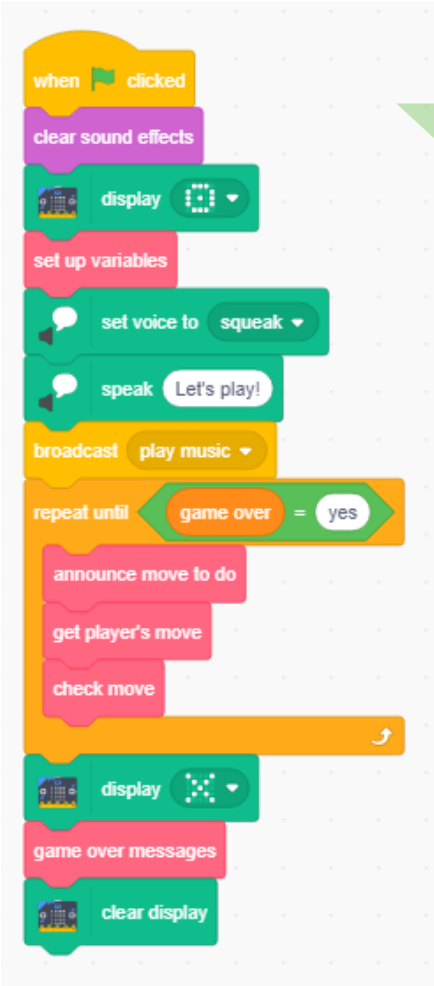
Step 3: Add the first script

We'll make our own blocks to make the project easier to read. Click the Code tab, then click My Blocks in the Blocks Palette, and click Make a Block. Create five new blocks called "set up variables", "announce move to do", "get player's move", "check move", and "game over messages". Use the Add Extension button in the bottom left of the Blocks Palette to add the Text to Speech extension. Click Variables in the Blocks Palette and make a variable called "game over".

Now you're ready to build the first listing. It shows a shape on the micro:bit LEDs using the **display** block, says "Let's play!" and begins the main game loop. You can use the **display** block to show any pattern of LEDs. Click the pattern in the block to redesign it.



Above: The micro:bit display block in scratch with the drawing dialog open



The Code

The block colours and any symbols on them help you find them in the Blocks Palette. Click the matching button to the left of the Blocks Palette to go to the right section.

Step 4: Set up variables

In the Variables, part of the Blocks Palette, use the Make a Variable button to create variables called "average response time", "move done", "move to do", "random direction", "score", and "total time". Make a list called "actions". Now add the second listing to your project. You'll need the **define** block that was created when you made the "set up variables" block in Step 3. It'll be in the Code Area. This script sets up the variables and list. The list is used to read the actions out, so beware of typos.

Step 5: Announce a move

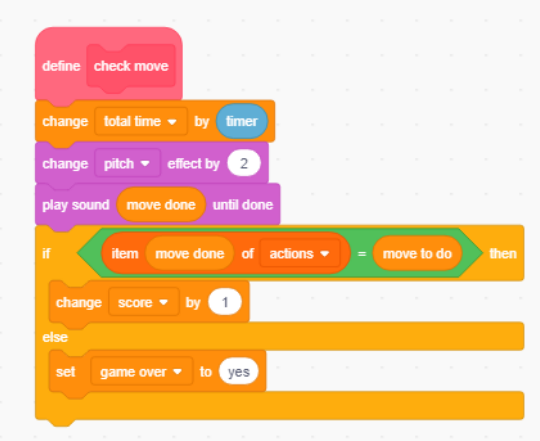
The next listing picks a random move number and then reads its name out. It also resets the timer, so the player's reaction time starts counting after the move is announced. Add this script now. Click the green flag and you should hear a rapid barrage of random instructions. Click the red stop button when you've heard enough.

Step 6: Get the player's move

The script to get the player's move checks for valid game movements until one is made. The move number the player makes is stored in the "move done" variable. It corresponds to the move name in the "actions" list (e.g. move 1 is "left button").

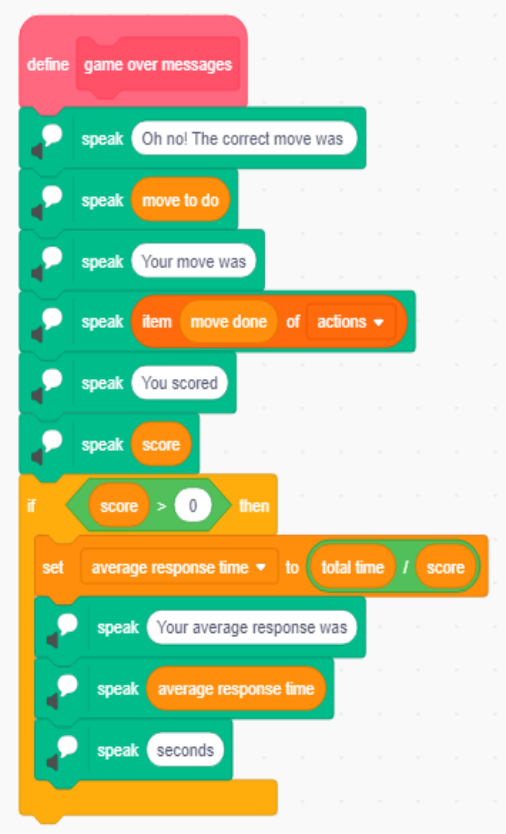
Step 7: Check the player's move

The **play sound until done** block is usually used with a sound name, but you can drop a number variable on it. I'm using it to play one of the first six sounds on the sprite, depending on the player's move, so each move has its own sound. I've used the new **change pitch** block to make the pitch higher each move, which adds tension. Each move, the time taken since the player's turn began is added to the "total time" variable. If the name of the player's move is the same as the name of the move the player needs to do, the score goes up. Otherwise, the "game over" variable is set to "yes" to exit the main game loop. Add this listing, and you can test the whole game flow now



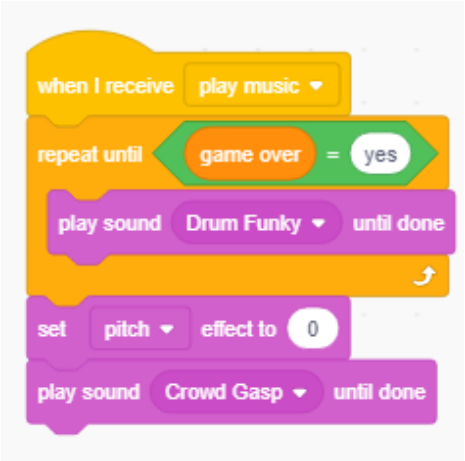
Step 8: Game over sequence

The "game over messages" script uses a number of **speak** blocks to tell the player how they did.



Step 9: Add background sound effects

The final script plays music during the game and plays the "crowd gasp" effect when the game ends. The background music is also affected by the changing pitch, so it seems to become more frantic as the game goes on.



Step 10: Refining the game

Now the game is ready to play! I set the minimum tilt angle at 30 degrees to make sure the tilt detected is intentional. When playing, you need to tilt until the move registers, then set the micro:bit flat again ready for the next move.

There are lots of things you can do to build on this game. You can change the mechanics so that players have to see how many they can get right in 30 seconds, or challenge them to memorise a sequence of moves to replay. You can add a script that displays a random arrow each move to try to confuse the player. What about rounding the average response time to two decimal places, so it's easier to understand when read out? Hint: you can treat numbers as strings and use blocks like **length of apple** and **letter 1 of apple** on them. There's a version of this game including the random arrows and the rounded result on my website.

Above:
I added a title screen to the game

You can access the project at:

go.micromag.cc/scratchaudiogame