Virgin media

Switched On Futures

Powered by

1 CODERDOJO SCOTLAND

Make a Mobot

Get ready.

We've designed our game with beginners in mind and reckon anyone can Make a Mobot, but the age range we think this works best for is 9-14.

To get the most out of it you'll probably want someone around who's got a bit of technical know-how in case you get stuck. New to Scratch? Don't worry, there are loads of tutorials available on the Scratch-Ed website to help you get going: **scratched.media.mit.edu/**

You can play this game on your own or with friends. If you want to find out a more about setting up a coding club for your friends or have any questions about the game, you can get in touch with us at **switchedonfutures@Virginmedia.co.uk.**

Set?

If you want to Make a Mobot there a couple of things you'll need to set up first:
- a Kinect controller
- a computer with Windows installed

You'll also need to download and install a couple of programs:
- Scratch 1.4 for windows
  You can download the page with instructions here: **scratch.mit.edu/scratch_1.4/**

- Kinect2Scratch
  You can download the page here: **scratch.saorog.com/setup**
  And the instructions are here: **scratch.saorog.com/setup.pdf**

GO!

All set up? Now you can get gaming! We've written out a step-by-step guide to getting your head around the basics of Scratch so you can Make a Mobot. You'll be able to customise what your robot wears and we'll show you how to use the Kinect to control what you build. We've de-signed the program to recognise certain signature moves like Mo's Mobot and you can even design some smooth moves of your own.

Part 1: Let's get started

Scratch is a visual programming language that allows you to give the computer instructions by arranging a series of blocks. This means you don't have to remember lots of different commands and type them in exactly right, something that even experienced programmers often make mistakes with!

It makes it easier to concentrate on working out exactly what it is you need to do to solve a problem - in our case making our robots come to life and recognise our 'signature moves.'

Introduction to Programming with Scratch

Scratch programs usually involve moving images called 'sprites' around the screen: the cat cartoon (Fig.1) is a sprite.

The code blocks in (Fig.2) will move our cat sprite 10 steps to the right and then wait for 1 second. The way we've coded it means the sprite will do this 10 times.

fig.1

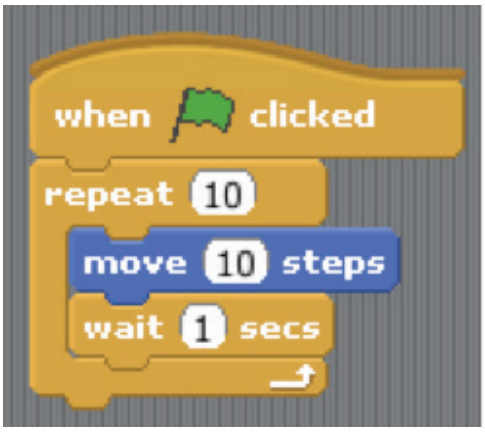

fig.2

Variables

A variable is like a box where we can store a number. For example we could have a variable called number_of_medals to keep track of how many medals an athlete has won so far in the games (Fig. 3).

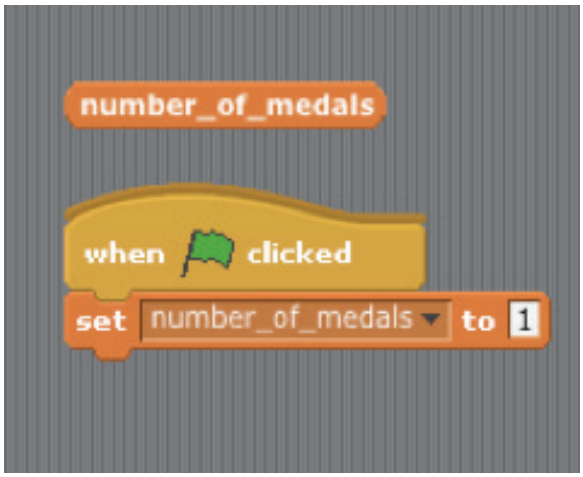

fig.3

Conditional Statement

Sometimes we only want something to happen if something else happens first. This is called a condition. You do this every day in real life – like if it's raining then you take an umbrella. In this case, if our athlete has won an event, the condition is to add 1 to the number of medals they have (Fig. 4).

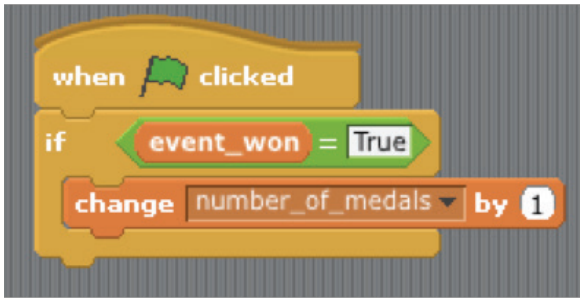

fig.4

Loops

Sometimes you need to repeat an action several times, which for can be pretty boring for a person. Luckily, computers don't get bored, so we can get them to do the same action over and over again. We do this using a loop - it also avoids us having to drag several copies of the same code into our program many times.

Suppose the total number of events an athlete can win is 10. Here we can use a loop to keep adding 1 to the number of medals won until it reaches 10 (Fig. 5).

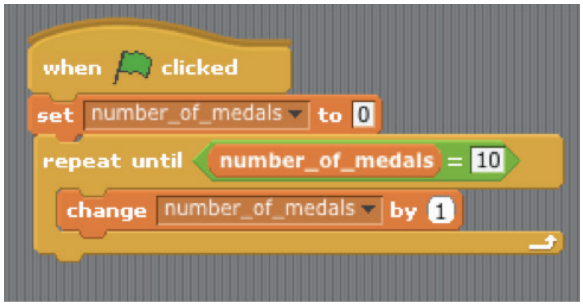

fig.5

All making sense?
Now it's time to make your robot...

Part 2: Make your Robot

How the Kinect works

The Kinect is able to detect humans in a room from the shape of their skeleton. The data is sent to Scratch, which uses a piece of code called Kinect2Scratch to set up some variables representing a number of points on the user's skeleton. These points are then used to draw a skele-ton figure on the Scratch stage.

Here's what the basic skeleton outline looks like before we turn it into a robot (Fig. 6). Each red dot is a Scratch sprite representing a different part of your body.

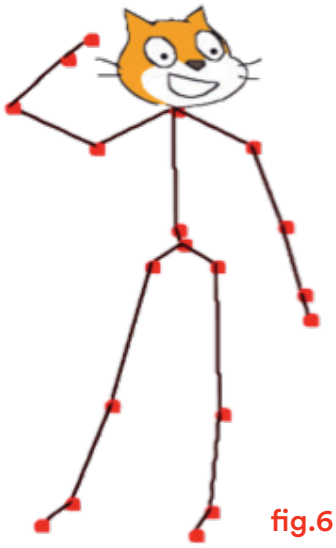

fig.6

To turn our skeleton into a robot we need to select a 'costume' for each part of its body. This can be done either by just clicking on them in the 'Costumes' tab of the sprite and selecting the costume you want (Fig. 7). Or you can put a tiny script into each sprite that changes to the correct costume when 'Go' is clicked (Fig. 8).

fig.7

fig.8

Part 3: Make you robot do the Mobot!

To bring your robot to life you need to link up the Kinect and the Scratch program.
To do this you need to follow a few easy steps:

1. **Connect the Kinect controller to your computer**
2. **Start up the Kinect To Scratch program**
3. **Click on 'Launch Kinect'**
4. **Click on 'Connect to Scratch'**
5. **Open the mobot.sb program in Scratch**
6. **Click the green arrow to start the program running**

Now you should see the Kinect picking up your movements and the robot copying them. If you can, use the Kinect in a clear space where there are no objects blocking the view of the camera. If your robot looks a bit scrambled it could be because the Kinect is picking up other objects in the room and thinks that you are a human/sofa hybrid! If this happens, close all the programs and start again, making sure that in the left hand window of the Kinect2Scratch program only your body shows up in red.

How Kinect2Scratch lets us draw a skeleton

This is the code that takes the data from the Kinect and uses it to draw the skeleton representing your body (Fig. 9).

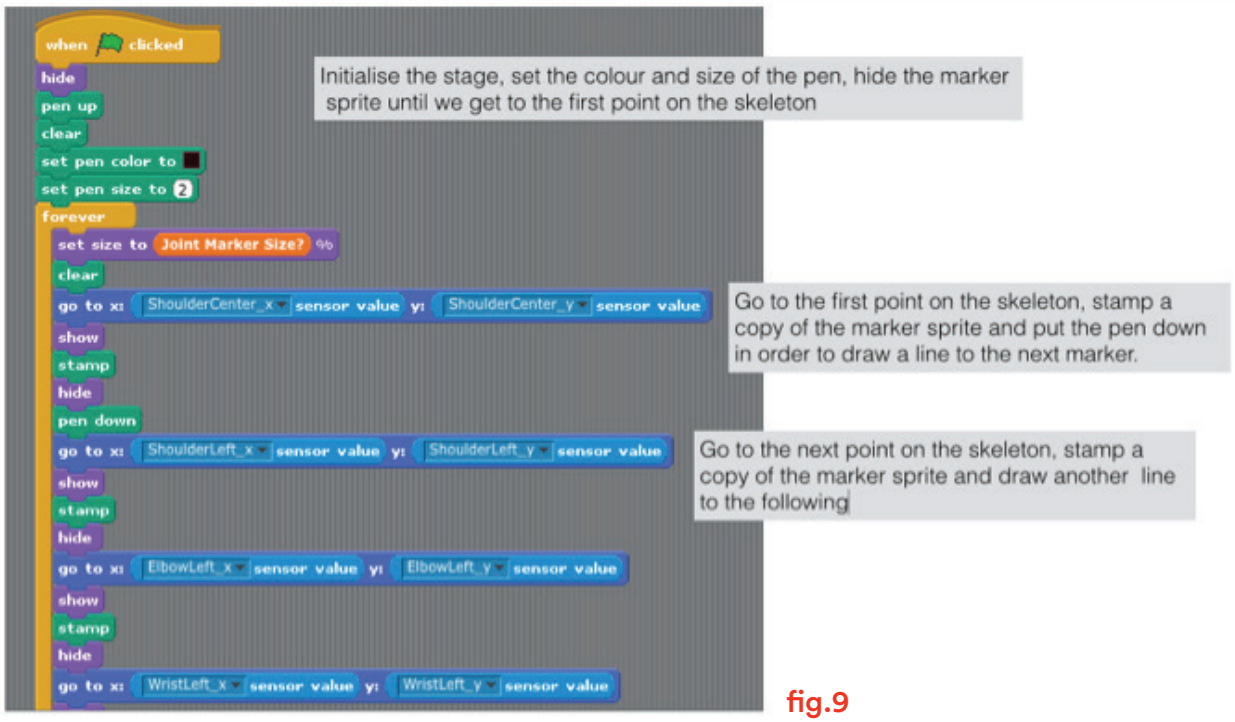

fig.9

Part of the script that draws the basic skeleton of the user from data Scratch receives from the Kinect

Each point on your body is defined as an x value (i.e. left to right) and a y value (up and down) (Fig. 10)

So the position of the sprite representing your right wrist is defined as its x-value: WristRight_x and its y-value: WristRight_y (Fig. 11)

Try out the robot's moves by moving your body and seeing what happens on screen.

You'll notice that your Mobot acts like a reflection of you - so when you move your right arm the Mobot moves its left arm, just as would happen if you were looking in a mirror.

To make it easier to work out how to describe signature moves, the sprites' names all refers to your body not the robot. For example, the robots left wrist is called Wrist_Right as it copies the movements of your right wrist.

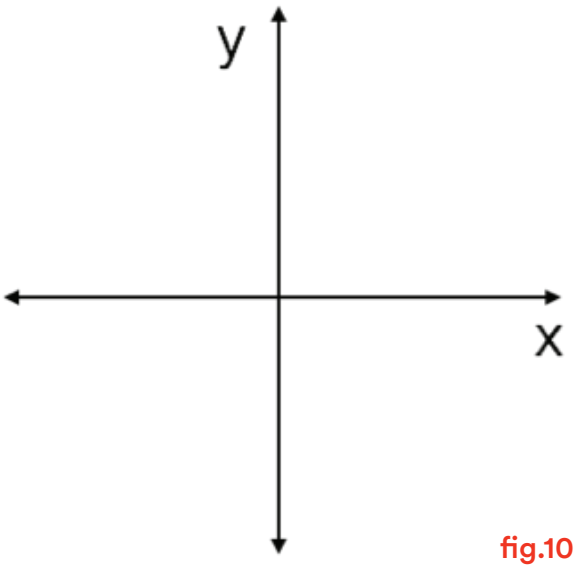

fig.10

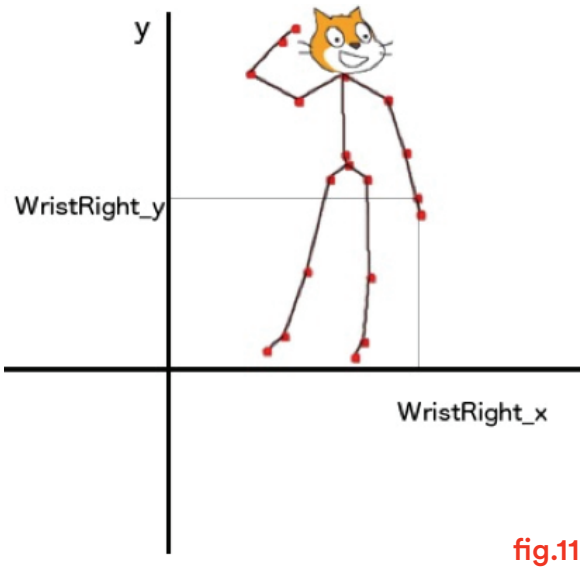

fig.11

Get your robot doing the Mobot!

Now that your robot is copying your movements we can get it to recognise some of our cool moves like the 'Mobot' and the 'Lightning Bolt.' When it recognises one of these moves, it displays the name on the screen and adds points to the user's score. Here it's displaying 'Mobot' and the current score (Fig. 12)

fig.12

Let's have a look at the code that makes this happen. In the next few diagrams the bits of code that we aren't talking about at each point are blurred out, so that we can concentrate on specific blocks.

The code that recognises and reacts to signature moves is in the script attached to an invisible sprite called 'Kinect Code' (you can see it highlighted in pale blue in the list of sprites in (Fig. 13). Any sprite can be made invisible, but usually we want to be able to see them. In this case though, our invisible sprite is a bit like the 'brains' of the Mobot, doing all the thinking and controlling. And as you can't normally see someone's brain, we've made it invisible here.

fig.13

The program (Fig. 14) has two variables that hold values used during the program: 'pose' is used to store the most recently recognised pose and 'score' holds the user's score in the game. The first thing we have to do is define our 'signature moves' in terms of the position your body. We use the 'marker points' that correspond to joints and other main parts of the robot's 'skeleton', like its head, feet and body.

Variables to hold values for current "pose" and "score". Because the box beside them is checked they will be displayed on the screen

When start is clicked we set pose to nothing and score to 0 so that we start the game with a "clean slate"

fig.14

(Fig. 15) highlights the block of code that defines what a Mobot is in terms of your body's position. It's a 'conditional', which means that the Mobot is only recognised if certain conditions are true. In this case we need both the Mobot's elbows to be higher than its head. We put 'both elbows higher than head' formally as 'y-value of left elbow is greater than y-value of head' and 'y-value of right elbow is greater than y-value of head.'

This condition is used in a larger block of code (Fig. 15), which checks to see if you are currently standing in a pose that meets the definition of a 'Mobot'. If they are it adds 10 points to your score and sets the value of the variable 'pose' to 'Mobot'.

fig.15

With the code as it is you could run up your score just by doing the 'Mobot' over and over again. But we think that's kind of cheating. To stop this happening we added a bit of code that checks to see what the current pose is. If 'pose' is set to 'Mobot' - because the last pose we did was a 'Mobot' - it tells us to 'Mix it up!' and doesn't add any points to our score. If 'pose' is set to another value it recognises that we've changed position, adds to our score, and sets "pose" to be 'Mobot' (Fig. 16).

fig.16

The check for the Mobot Signature move is enclosed inside a 'Forever' block (Fig. 17). This means that the program keeps checking for the move over and over again as long as it's running.

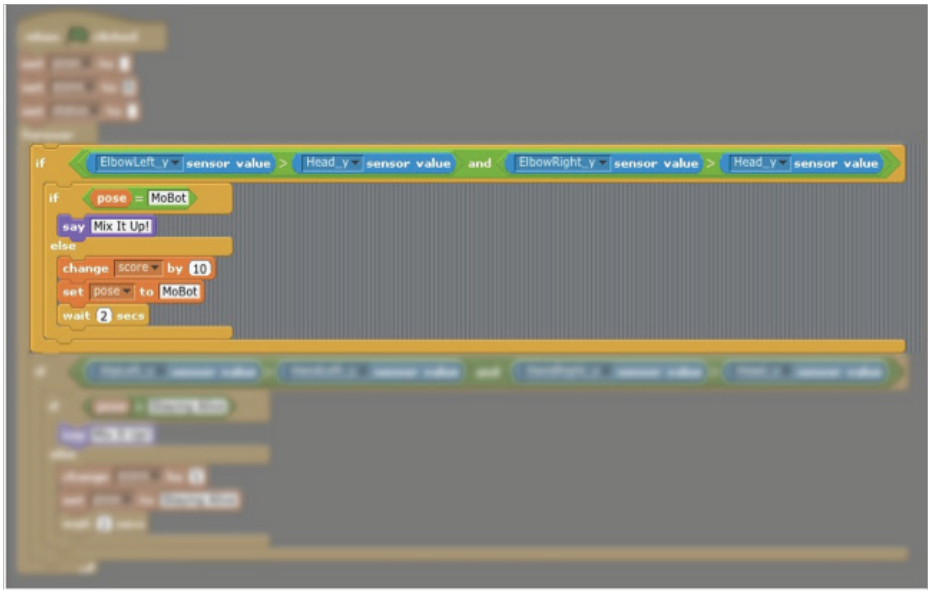

fig.17

We can also add more blocks of code that check for other signature moves to the forever loop. The second block in Fig. 18 is very like the 'Mobot' block, but it checks for the 'Staying Alive' move.

fig.18

Part 4: Now it's time to make your own moves!

Now that you know how the program reads the moves we've put in, you can invent your own smooth moves and get Scratch to recognise them too. Let's break down this task into some smaller tasks by looking at how we added the 'Staying Alive' move (Fig. 19).

- Work out your move - it should have a strong shape and not be too complicated.
- Work out how to describe it in words: Left hand lower than left hip, right hand higher than head. It might help to draw your move out on a bit of paper or a whiteboard for this bit.
- Now work out how to say that in terms of sensors: HipLeft_y > HandLeft_y and HandRight_y > Head_y.
- Copy the block of code that identifies the MoBot by right-clicking on it and selecting 'Duplcate', then edit it so that it recognises your move instead (Fig. 20).

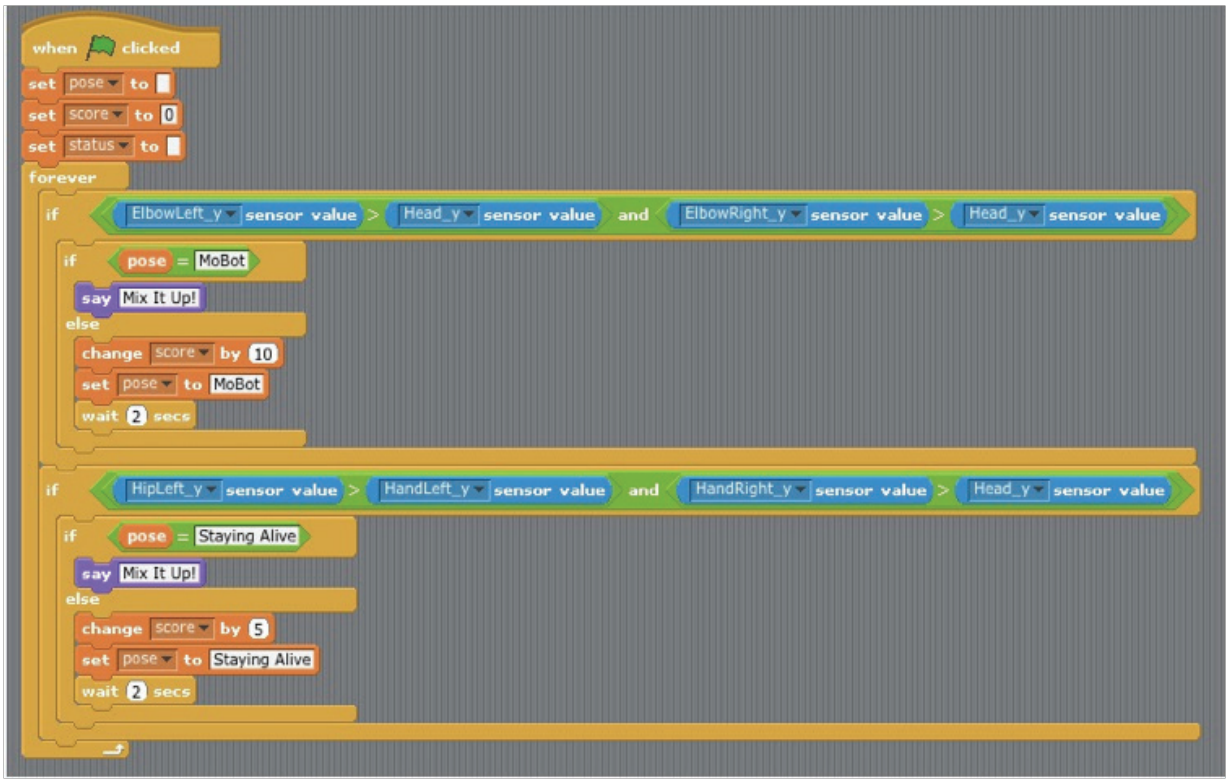

fig.19

Here we've changed the condition at the start of the if block to be the definition of the "Staying Alive" and the value of "pose" to be "Staying Alive" too.

You can add as many moves as you like by duplicating the "MoBot" block, putting the new block inside the Forever loop too, and editing it to recognise your new move.

fig.20