Raspberry Pi
Scratch Maze Game

Tutorial by Andrew Oakley
Public Domain 24 April 2016
www.cotswoldjam.org

Getting started

This tutorial uses Scratch version 1.4. It'll work on the Raspberry Pi, PC or Mac.

```
Words you will see on the screen, or that you need to type in, are highlighted like this
```

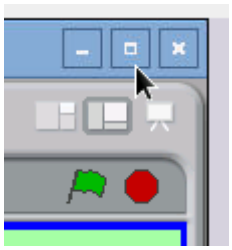
At the end of typing something, you will usually have to press the Enter key. However there is not much typing in Scratch - it's mostly using the mouse.

Your tutor should have already downloaded the files for this tutorial from: http://www.cotswoldjam.org/downloads/2016-04

Your tutor should have unzipped the files and saved them in the /home/pi/scratch/maze folder. Your tutor may have needed to create the scratch and maze folders.

Starting Scratch

Start Scratch by clicking on Menu - Programming - Scratch.



Maximise the window by clicking the big square icon in the top-right corner of the window. It should now fill the whole screen.

You can also see the green flag, which starts the program, and the red stop button, which can stop a program if it goes wrong.

Scratch uses the word "stage" to describe the screen background, and "sprites" to describe people, objects or monsters than inhabit the game.

We've created a stage and sprites for you. Load them in by clicking File menu - Open - click Pi - Scratch - maze - maze1 . If you're asked if you want to save the current project, click No.

Your first program

Click on "Person". **Make sure you have "Person" selected and not "Berry".**

Programming in Scratch is done with blocks. There are lots of different blocks which are organised by category.

Next, click "Control". You will see the Control blocks.

Drag "when Space key pressed" into the Scripts area in the middle of the screen.

(To drag something, point at it with the mouse, hold down the left mouse button, keep the button held down whilst moving the mouse to where you want it, and then let go of the mouse button)

We want to move the person in the game, when the right arrow key is pressed.

Click the black triangle next to Space in "when space key pressed. Change "space" to "right arrow".

Click the Motion section.

Drag "change x by 10" so it locks on to the bottom of "when right arrow key pressed".

Create another set of scripts for "left arrow".

In "change x by 10" for the left arrow script, click the black triangle and change 10 to -10 (minus ten). Use the delete and backspace keys to rub out 10 and then type -10

Click the green flag above the stage.

Press the left and right arrow keys. The person should move left and right.

Notice how the person can move through the blue maze wall on the left.

Is the raspberry moving instead of the person? As we said at the start of this chapter, **make sure you have "Person" selected and not "Berry".**

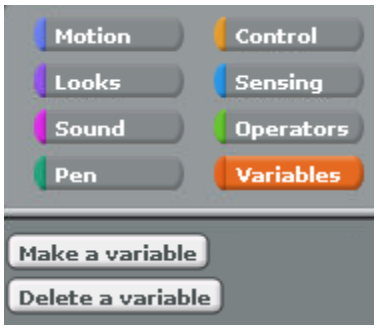
If you have difficulty setting up the program, you can load the file "maze2" instead.

Variables and Subroutines

We are going to use variables to test whether the person has hit a wall.

A variable is like a box that holds something, usually numbers or words. We are going to create a box called movex that holds a number. We'll then check whether moving the person by that many places, hits a wall. If they do hit a wall, we'll move the person back again.

We are also going to create a subroutine to do the wall checking. A subroutine is a script that can be called by other scripts. In Scratch we use the commands "broadcast" to call a subroutine, and "receive" to define what a subroutine does.



Go to Variables and click "Make a varaible".

Type:

```
movex
```

...and click Ok. Make sure there are no spaces in "movex".

We need to delete "change x by 10". Grab "change x by 10" and move it out of the Scripts box - for instance, move it back to the controls area. It will disappear.

In Variables, grab "set movex to 0" and stick it under "when right arrow key pressed".

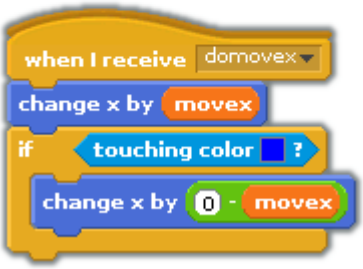
Click the value 0 and change "set movex to 0" to "set movex to 10" (ten).

Now go to Control and grab the "broadcast and wait" block. Stick this under "set movex to 10".

Use the black triangle to create a new broadcast named:

```
domovex
```

Repeat the above steps for left arrow. Remember to use -10 instead of 10.



Now create a new script by dragging "when I receive domovex" from the Control section to the Scripts section.

From the Motion section, grab "change x by" and stick it under "when I receive".

From the Variables section, grab "movex" and stick it onto "change x by". It should now read "change x by movex".

From the Control section, grab "if" and stick it under "change x by".

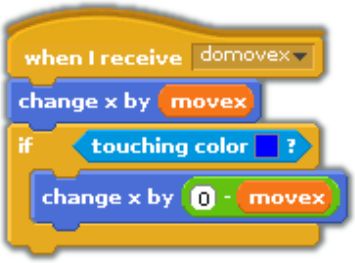
From the Sensing section, grab "touching color" and stick it onto "if".

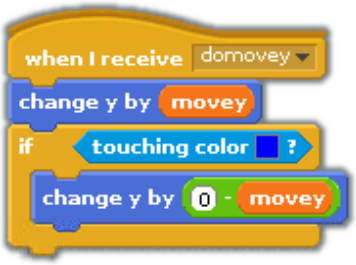
Click the navy blue box in "touching colour", and then select a royal blue wall colour from the maze- the mouse pointer will change to an eyedropper; point it at a blue wall and click.

From the Motion section, grab "change x by" and stick it inside "if".

From the Operators section, grab "[] - []" (minus, take-away) and stick it onto "change x by".

Change []-[] so that it reads 0-movex (grab movex from the Variables section). 0 is zero.



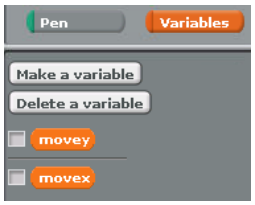


If you have difficulty setting up the program, you can load the file "maze3" instead.

What we are doing is moving the person, checking whether they have hit the wall, and if they have hit the wall, moving the person back again.

Run the program with the green flag. Use the left and right arrows. The person should stop when they come to the blue wall on the left.

Now create the same scripts for up and down, using y instead of x. For example create a new variable called movey, a broadcast called domovey and use "change y by".Run the program with the green flag. You should now be able to move the person all around the maze.



The variables movex and movey may be displayed on the maze. You can hide them by unticking their boxes in the Variables section.

Detecting a Win

At the end of each domove script, broadcast a new message called "checkwin". Create a new script called "checkwin" and check whether the person is touching the red raspberry colour.

From the Looks controls, use the Say command to tell the player they've won. Remember to reset the person to their original position after they've won.

You could also reset the person's position when the green flag is clicked, in case the program has been stopped half-way.

If you have difficulty setting up the program, you can load the file "maze4" instead.

Counting Moves

You can make the game better by adding a counter that counts the number of moves. You'll need to make a variable called "count".

You'll also need an "if...else" control rather than a normal "if "control. You may need to scroll down the list of controls to find "if...else". "If...else" allows you to do one thing if the condition is true (person has touched blue) and another thing if the condition is false (person has NOT touched blue).

Change the domovex and domovey "if" commands to "if...else" commands. If the person moves but doesn't hit a wall (the "else" section), then "change count by 1" (this is in the Variables controls)

Remember to make sure the box is ticked next to the variable "count", so you can see how many moves the player has made. **See the program "maze5" for an example.**