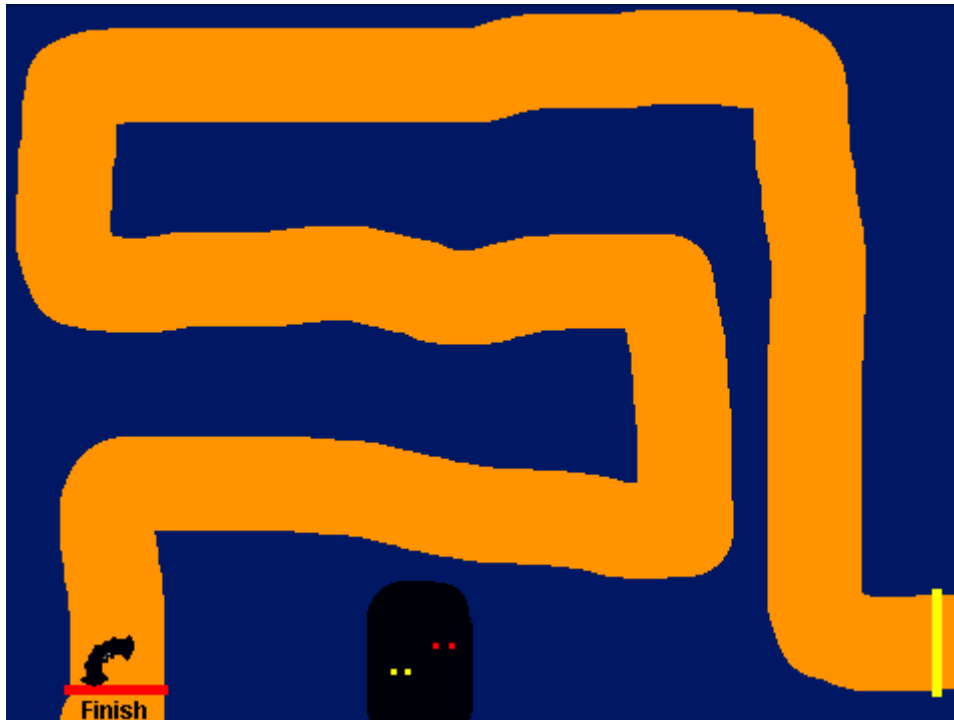# Introducing Scratch

Exercise 1. Bat Cave

In this exercise we'll create a maze game like the one shown below. The bat will start one end of the tunnel and the player will use their mouse to guide the bat to the other end. If the bat touches the sides of the tunnel the player will have to start again.

1. First start a new scratch project and remove the default sprite (You can use the poor cat another time).

2. Before we create any sprites we will need to create a background for the maze. Click on the stage in the sprite list (if there are no sprites it will probably be selected already).
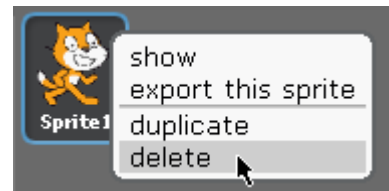
3. With the stage selected, click on the **Backgrounds** tab at the top of the **Script area**.

4. You will currently have a plain white background. Click on the **Edit** button to edit the background (you can also click the **Paint** button to paint a new background that will sit on top of the current one but there is really no need to have two backgrounds unless you want them to change in your program).
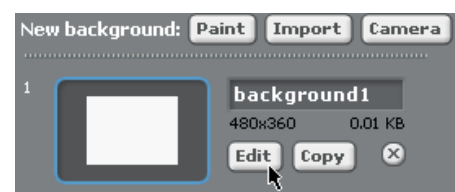
5. Use the drawing tools to create your maze. It will need to include the following:

- A tunnel/maze drawn in one colour with the area around it in a different colour. In the following example the tunnel is a light brown while the area surrounding it is blue. The player will lose the game if the bat touches the colour around the tunnel.

- A starting line in a unique colour. The example below has a yellow starting line. This is mostly for decoration to mark the starting point and won't be used in any of our script blocks.

- A finish line in a distinctive colour. In the example below the line is red. This is important as the program will end when the bat touches this colour.

- Any additional decoration you want to add to make your tunnel look good.

Note: Remember the tunnel needs to be large enough to allow the player to guide an object through it with their mouse. The narrower it is and the more turns there are, the harder the game will be.
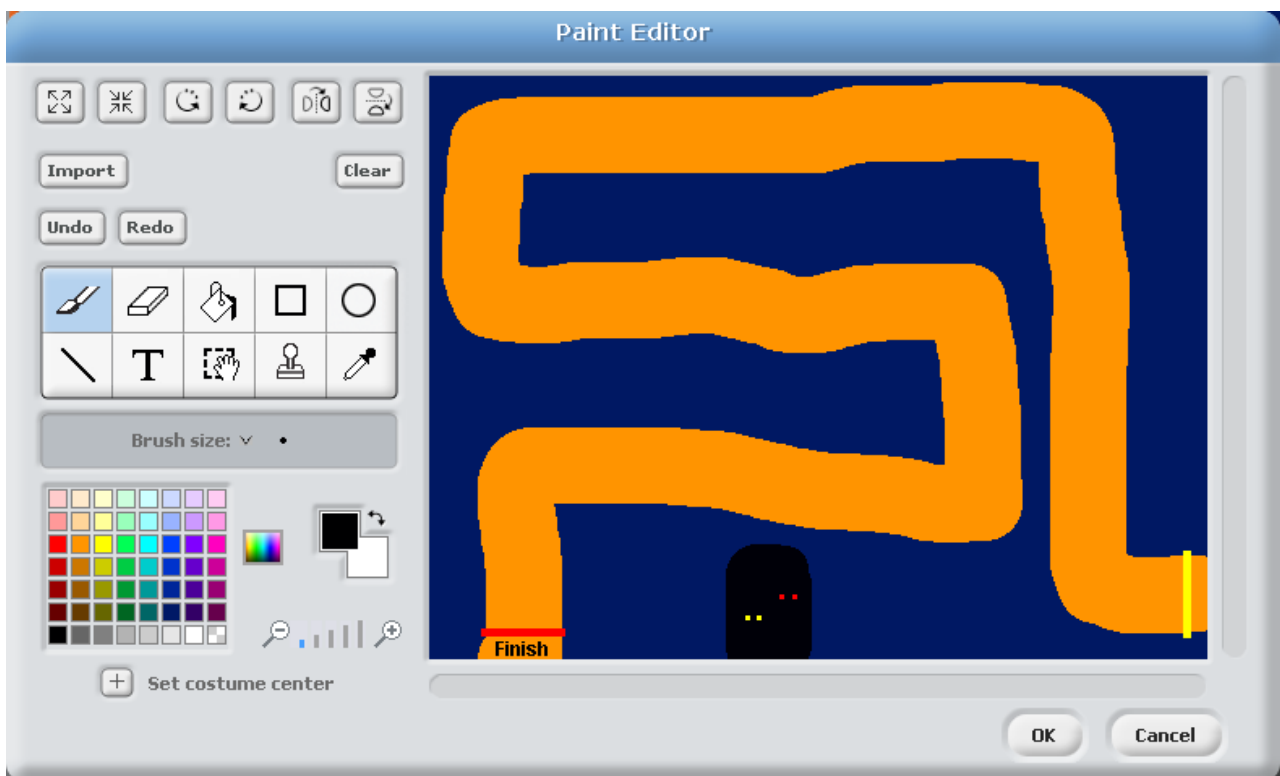
6. Click **OK** when you are satisfied with your background (you can always edit it later).

7. From the **Sprites list** area, click the **Choose new sprite from file** button.

Normally the *Costumes* folder will be the default location. If it isn't, you can find it under the folder where Scratch is installed in the *Media* folder.
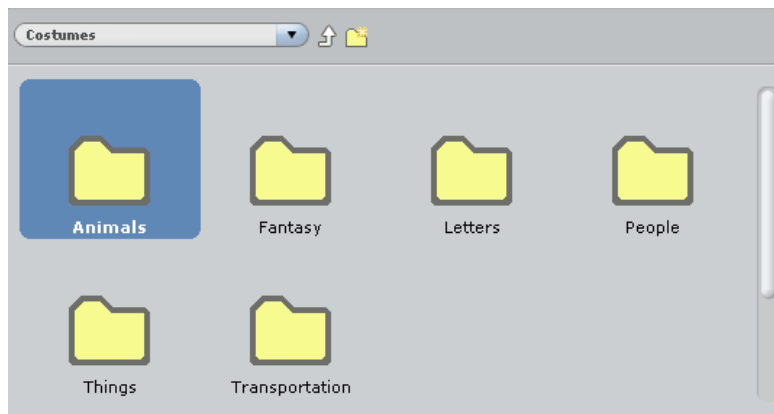
8. Double-click the *Animals* folder to open it.
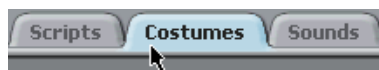
9. Double-click the file *bat2-a* to open it.

10. Rename the new *sprite1* as *Bat*.

While our program is running, we will animate the bat by making it switch between two versions of the bat picture. We need to import the other bat picture as an additional *costume*.

11. With the *Bat* sprite still selected, click the **Costumes** tab at the top of the **Scripts area**.

The existing costume will already be showing. Click the Import button to add the second costume. This time the file we will add is bat2-b. You should now see both costumes listed.
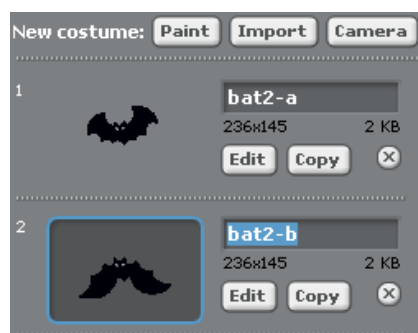
Note   You can change the names of the costumes if you want to but the current names are descriptive enough for us to know which is which.

12. Click the **Scripts** tab in the **Scripts area**. It's time to add some script blocks to tell the bat what to do.
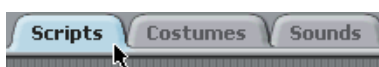
First we'll add some instructions that will position and re-size the bat when the program starts.

13. From the **Control** category add a **When** 🚩 **clicked** block to the **Scripts** area.

14. From the **Looks** category add a **Set size to 100%** block under the green flag block.

15. Change the size % to a size that will make the bat fit comfortably in your tunnel (you may need to click the 🚩 button a few times to test that the size is alright.

16. From the **Motion** category, add a **go to x: y:** block with coordinates that will position the bat on the starting line.

Tip   Drag the bat to where you want it to be first. Then when you select the go to x: y: block the current position of the sprite will already be filled in. You can also find out the coordinates of a position on the stage by moving your mouse over it and then looking at the bottom right corner of the stage.   `x: 190    y: -142`

17. Add a **Point in direction** block that will face the bat in the direction you want it to be facing at the start. The example below shows all of these blocks together with the result.
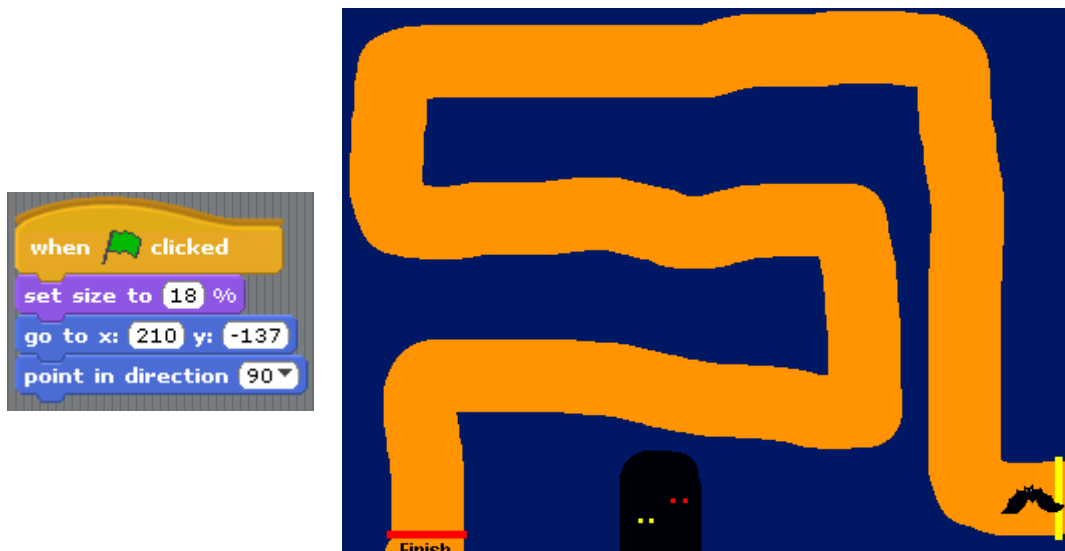
Next we will add a group of blocks that will make the bat animate.

18. From the **Control** category add a **forever** block.

19. From the **Looks** category add a **switch to costume** block inside the **forever** block. Make sure the costume is set to *bat2-b*.

20. From the **Control** block add a **wait 1 secs** block and change the number to *0.25*.

21. From the **Looks** category add a **switch to costume** block inside the **forever** block. Make sure the costume is set to *bat2-a*.

22. From the **Control** block add a **wait 1 secs** block and change the number to *0.25*.
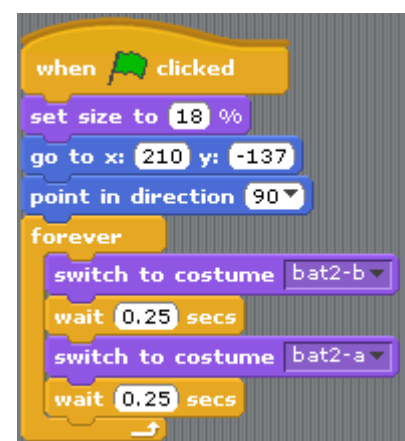
The example to the right shows how the finished blocks should look.

23. Click the **green flag** button to test your program so far. The bat should resize and move to the starting line. It should also look like its wings are flapping.

24. Click the 🛑 button to stop the program.

We will have the user start the game by clicking on the bat when they are ready.

25. From the **Control** category add a **when Bat clicked** block.

26. From the **Control** category add a **forever** block.

27. From the **Motion** category add a **point towards** block inside the **forever** block and set the option to *mouse-pointer*. If you test the program by clicking the bat now, the bat will keep on pointing towards the mouse pointer as you move the mouse.

Tip   You can change options in a block before dragging it on to the Scripts area.

28. From the **Motion** category add a **move 10 steps** block under the **point towards** block and change the number to *2*. This block will cause the bat to move in the direction it is facing. If you click the bat to test it, the bat will now move in the direction of the mouse.
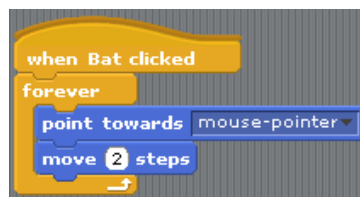
One current problem is that when the bat reaches the mouse pointer, it will start spinning rapidly since it is still trying to turn and face the mouse pointer which it is now right on top of. We will add some blocks which change the program so that it no longer tries to follow the mouse pointer when it is already close to it.

29. Drag the **point towards** block so that it is outside the **forever** block. Any blocks below it, in this case the **move** block, will also go with it. Blocks can easily be moved and rearranged in the **Script area**, just like the jigsaw pieces they are shaped like.
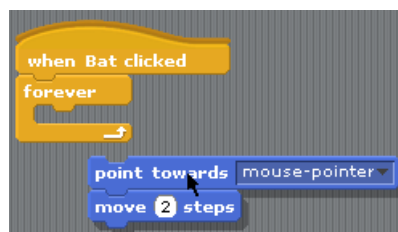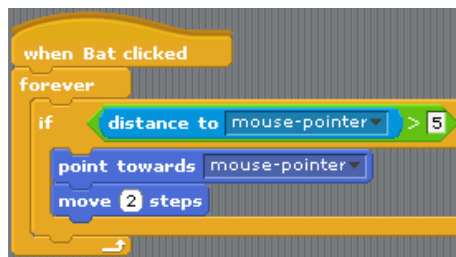

30. From the **Control** category, add an **if** block inside the forever block.

31. From the **Operators** category add a **greater than** block.

32. In the second space of the **>** block, put the number *5*.

33. In the first space, add a **distance to** block from the sensing category. Set the option to *mouse-pointer*.

34. Drag the **point towards** block (along with the **move** block) inside the **if** block. Now when you test it by clicking the bat, it will only follow the mouse pointer if it is more than *5* pixels away from the mouse pointer.

Next we will add a block that tells the program what to do if the bat touches the sides of the tunnel.
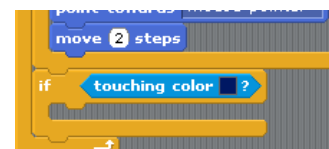
35. From the **Control** category, add an **if** block below the existing if block.

36. From the **Sensing** category locate the **touching color** block.

37. Click the coloured square to change the colour. Your mouse pointer will change to an eye dropper shape 🖊. We will use this to set the colour to the colour around your tunnel.

38. Click on your background around the tunnel to set that as the colour.

39. Now drag the **touching colour** block to the space in your new **if** block.

40. In the **Looks** category locate the **say for 2 secs** block and add it inside the **if** block.

41. Change the **Hello!** text to **Try again!**.

42. From the **Control** category find the **stop all** block (it is at the bottom) and add it under the **say for 2 secs** block.

43. Test your program by clicking the bat. Now whenever the bat touches the sides of your tunnel, you should see the try again message and the program will stop.

The last step is to add an if block that will show the message Well done when the bat touches the colour of your finishing line. Try using what you have learned to add those blocks. A completed example is shown on the following page.

44. Save the file as *Bat Cave*.
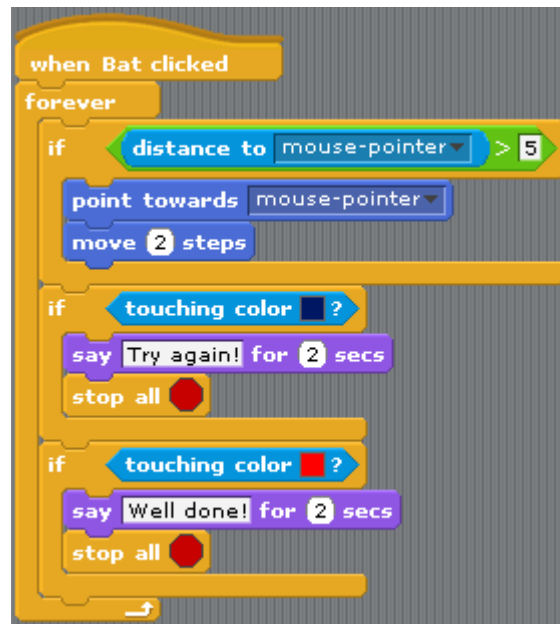
Exercise 2. Additional things to try

- You can make your game harder by changing the background picture to make the tunnel more complex. Add more twists and turns or by making it narrower.

- You can also adjust the difficulty by changing the size of the bat.

- You can change the speed the bat moves.

- Add a message that tells the user to click on the bat to begin. This message can be added to the background. It could also be added as a **say for 2 secs** block in the **When clicked block** so the bat says the message when the program is started.

Exercise 3. Adding Scoring

1. Select the **Stage** in the sprite area and add 2 variables named **Time** and **Best Time**.

2. Position them so that they aren't in the way of the tunnel the bat will be moving through.

3. From the **Variables** category find the **Set to** block. Add it inside the **Forever** block of the **When Bat Clicked** section.

4. Make sure the **Variable** is set to **Time**. In the second part add a **Timer** block (from the **Sensing** category).

We will also add a step which sets the timer to 0 when the bat is clicked.

5. From the **Sensing** category add a **reset timer** block directly under the **When Bat Clicked** block. The start of that block should look like the example shown to the right.

6. Test your program. The score timer (which won't be visible) should now start changing the value of the Time variable as soon as the bat is clicked.

We already have a section which checks for the bat touching the finish line. In that section, we will add some instructions which check to see if the time is better than the lowest recorded time and changes the best time accordingly.

Firstly you will want to some instructions that check to see if the best score is 0 (if no best score has already been recorded). If it is 0 then the score will become the new best score.

If the best score isn't 0 then we want it to check to see if the score is lower than the best score. If it is, then the score will become the new best score.

7. Modify your **When Bat Clicked** block to meet the conditions described and test it when done.