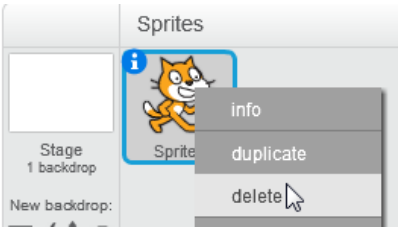
Introducing Scratch

2 – Maze Game

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.

Exercise 1. Create the Maze

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 **Backdrops** tab at the top of the **Script area**.
4. At the moment we have a blank white background. 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 a different colour. The area around the tunnel is important as we will tell the program to end the game when our main sprite touches that colour. The edge of the tunnel must all be the same colour since that is how the program will know the side of the tunnel has been touched.

A starting line in a unique colour. This is mostly for decoration to mark the starting point and won't be used in any of our script.

A finish line in a distinctive colour. This is important as we will tell the program that a player has completed the maze when the main sprite touches this colour.

Any additional decoration you want to add to make the 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.

For the time being it's probably best not to make the tunnel too complex otherwise you'll have a hard time testing the game. Once you know it's all working properly, you can come back and edit the background to make the tunnel more challenging.

5. Save the project as Bat Maze.

In the example below, the area outside the tunnel is dark blue. You can put anything you like within that area such as the cave in this example. As long as the border of the tunnel is all the same colour.

A red colour has been used for the finish line in this example.

Exercise 2. Creating the Bat Sprite

1. From the Sprites area, click the **Choose sprite from library** button.
2. The sprite library will show various categories down the side. Select **Animals**.
3. Click the sprite called **Bat2**.

A small note under the sprite tells us that this particular one comes with 2 costumes. One is a picture of a bat with its wings up and the other is a picture of the bat with its wings down. That will be pretty handy since we will make the program switch between the 2 different costumes to make it look like the bat is flapping its wings.

4. Click **OK** to finish selecting the **Bat2** sprite. If you like you can rename the imported sprite to simply **Bat**.
5. Click the **Costumes** tab at the top of the script area to see the 2 costumes for the Bat sprite.

When you select one of the costumes you can change the name of the costume and also edit the picture for the selected costume but we won't make any changes to them at the moment.

6. Click the **Scripts** tab. It's time to tell the bat what to do.
7. From the **Events** category, add a **when** clicked block to the scripts area.
8. From the **Looks** category, add a **set size to 100%** block under the previous block.
9. Change the size to an amount that will make your bat fit in the tunnel with a bit of room to move on either side. You may need to click the to test it until you get the size just right. In the example below, the size has been set to 25%

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

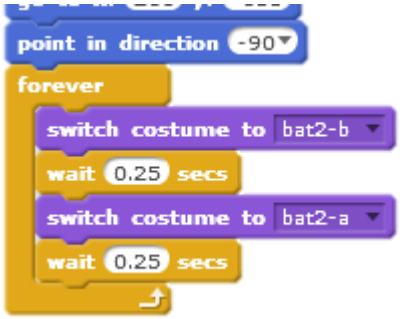
Tip Drag the bat to where you want it to be first. Then when you select the **go to x: y: block** the current position of the sprite will already be filled in. You can also find out the coordinates of a position on the stage by moving your mouse over it and then looking at the bottom right corner of the stage. x: 182 y: -122

11. 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. We will do this by adding a loop that switches between the two bat costumes so that it looks like the wings are flapping.

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

13. From the **Looks** category add a **switch to costume** block inside the forever block. Make sure the costume is set to **bat2-b** (You can check the costumes tab to see the names of the costumes. The first one should be bat2-a).
14. From the **Control** block add a **wait 1 secs** block and change the number to **0.25**.
15. From the **Looks** category add a **switch to costume** block inside the forever block. Make sure the costume is set to **bat2-a**.
16. 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.

Tip If you change the direction your bat is facing, it might end up being upside down. You can fix that by going to the costumes tab and clicking the **Flip up-down** button for both costumes.

17. Click the 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. You might want to make some additional adjustments to the **X** and **Y** positions to make sure the starting point is just right.
18. Click the button to stop the program.

We will have the user start the game by clicking on the bat when they are ready. You could also have it start when the user presses a key if you prefer.

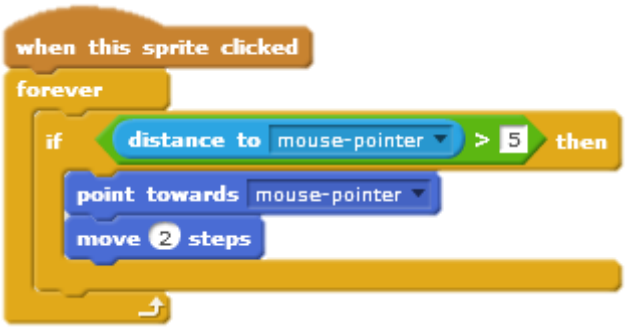
19. From the **Event** category add a **when this sprite clicked** block. You can add it to the side of your existing blocks if there is enough room.
20. From the **Control** category add a **forever** block.
21. From the **Motion** category add a **point towards** block inside the **forever** block 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.
22. 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.

When you test it, you may notice that when the bat reaches the mouse pointer, it will start spinning rapidly. 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.

23. 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.

24. From the **Control** category, add an **if** ___ **then** block inside the **forever** block.
25. From the **Operators** category add a **greater than** block inside the condition area of the if block.
26. In the second space of the **>** block, put the number **5**.

27. In the first space, add a **distance to** block from the sensing category. Set the option to **mouse-pointer**.
28. 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.



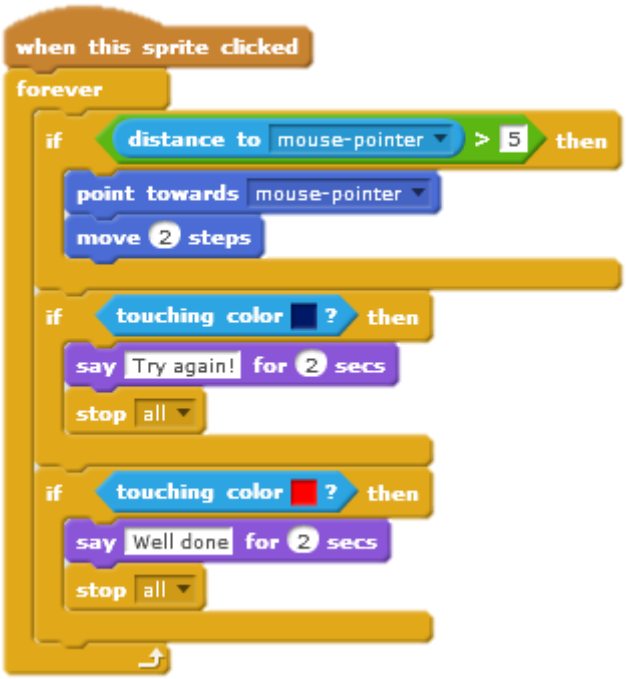
Next we will add a block that tells the program what to do if the bat touches the sides of the tunnel.

29. From the **Control** category, add an **if** ___ **then** block below the existing **if** block.
30. From the **Sensing** category locate the **touching color** block.

31. Click the coloured square to change the colour. Your mouse pointer will change to a hand shape . We will use this to set the colour to the colour around your tunnel.
32. Click on your background around the tunnel to set that as the colour.
33. Now drag the touching colour block to the condition space in your new if block.
34. In the Looks category locate the say for 2 secs block and add it inside the if block.
35. Change the **Hello!** text to **Try again!**.

36. From the **Control** category find the **stop all** block and add it under the **say for 2 secs** block.
37. 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.

38. The last step is to add an **if** ___ **then** 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 below.
39. Save the changes to your file.



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 make 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. From the **Data** category 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 **Data** 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 an instruction 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 the best score is 0 (if no best time has been recorded yet) 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 latest score is lower than the best score. If it is, then the score will become the new best score.

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

Note that we have used a **if** __ **else** block and inside the else part there is another **if** block

So these instructions are basically saying, if the bat touches the finish line colour – check to see if the best time is 0. If it is 0 then change it to whatever the time variable is.

If the best time isn't 0, then check to see if the most recent time is less than the best time. If it is less, then that will become the new best time.