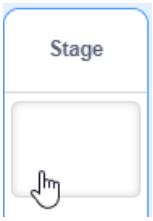
Introducing Scratch

6 – Racing Cars

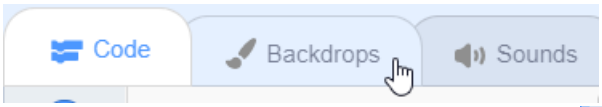
Exercise 1. Create the Background

In this exercise we will create a car racing game for 2 players. First we will create the background for the game including the racing track. This will be very similar to creating the background for the Maze Game in an earlier lesson.

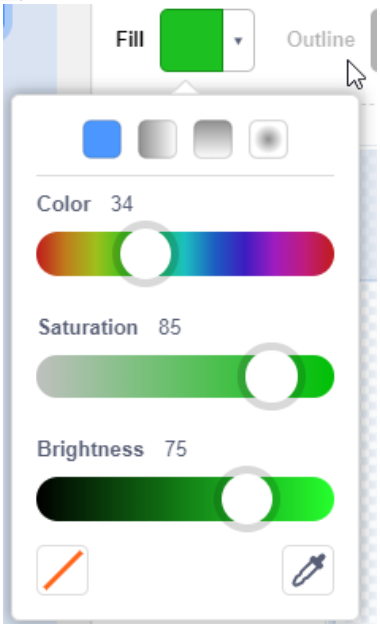
1. Start with a new project and click on the **stage** to select it.



2. Select the **Backdrops** tab



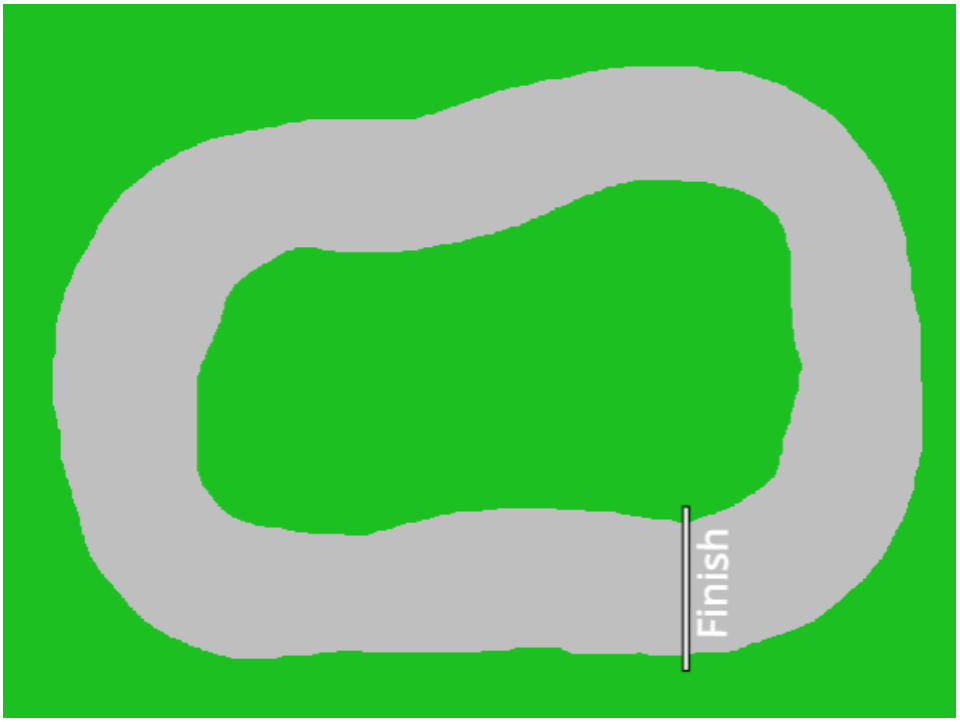
3. Pick a fill colour and then use the paint bucket tool to fill the stage with the colour you want outside your track (such as a green grass colour). You will need to convert to bitmap mode first, since in vector mode the fill tool will only fill shapes you have drawn.



4. Select the **Brush** tool and pick a colour suitable for your track (if you want grey, move the saturation all the way to the left).

5. Set the brush size to 100 pixels which is the maximum

6. Draw a racing track on your stage. Since our cars will be doing laps, make sure the start of your track joins on to the end to make a continuous track. Make sure your track doesn't go all the way to the edge of the stage. Every part of the track needs to have an edge for cars to crash in to.

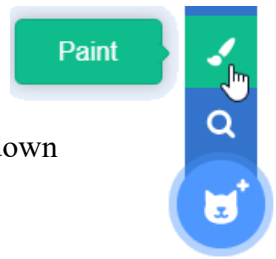


7. Add any additional decorations to your track. Just as long as you make sure that the edge of the track is all the same colour.

Tip Including more twists and turns or making the track narrower will make it more challenging for players. While you are testing the game it's best to keep it easy. You can redraw the track to make it harder once you know everything works properly.

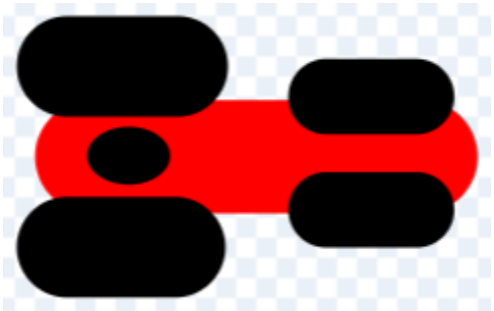
Exercise 2. Creating the sprites

1. Delete the cat sprite.
2. Click on **Paint Sprite**

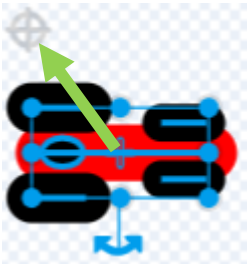


Our cars are going to be pretty small so click the Zoom icon to get a bit closer. This will make it easy to draw your small cars. You can also hold down **Ctrl** and roll your mouse wheel to zoom in and out in sprite edit mode.

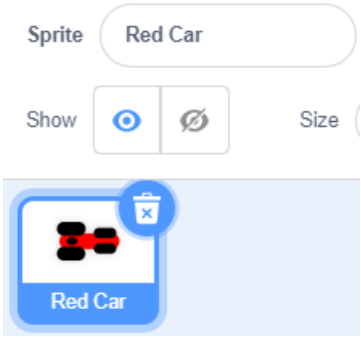
3. Use the drawing tools to draw a red car. The following is a simple example, but you can be creative and make yours however you want. If you want to draw yours the same, then refer to the end of this exercise document for some tips.



It is very important to make sure your car is in the centre of the sprite area. If it is not then make sure it is selected by dragging around it with the selection tool. Then drag it so that it's centre lines up with the small cross in the middle of the sprite editor.



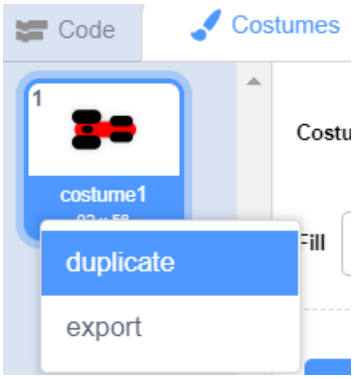
4. Change the sprite name to **Red Car**.



5. Save the game as **Racing Cars**.

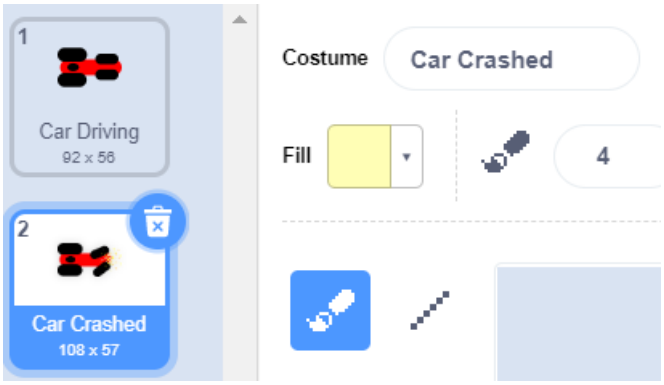
Tip Remember to save your game regularly while you are working

6. In this sprite we will have two costumes. One for the car while it's driving and another one for when the car crashes. In the costume section to the left of the drawing area, right-click on the first costume and then click **duplicate**.

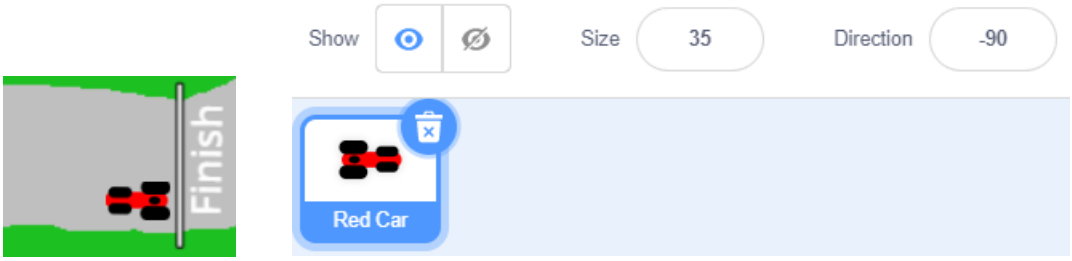


7. Modify the new copy of the sprite so that it looks like the car has crashed (in the example below, the dots were added after converting the costume to bitmap mode).

8. Change the costume names to **Car Driving** and **Car Crashed**.



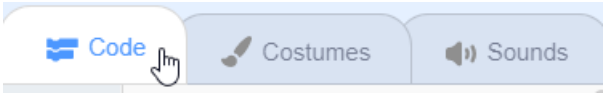
9. Click on the Car Driving costume to make sure that is the one currently showing on the stage.
10. Drag the car sprite on the stage to the place where you want it to start the race. Adjust the Size and direction of the car so that it fits on the track with plenty of room for a second car next to it.



Exercise 3. Add the Starting Code

There are several things that need to happen when the program starts so we will now add them in as code.

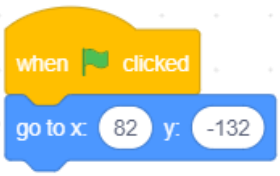
1. Click the **Code** tab.



We'll start by adding code that places the car where we want it to be at the start of each race.

2. From the **Events** category add a **when**  **clicked** block.

3. Select the **Motion** category.
4. Add a **Go to block**. The x and y coordinates are already filled in with the current location of the sprite. You might need to make some adjustments to get the starting position right.



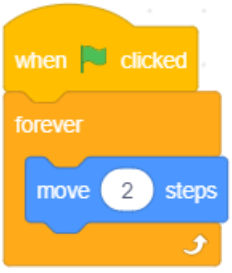
5. Add a **point in direction** block.
6. Change the angle so that the car is facing in the direction you want it to be facing when you start the race. You might need to click the green flag to test it a few times until you get the angle just right.

Exercise 4. Moving the Car

Now to add some code that will move the car once the race starts.

1. Add some additional blocks that will make the car turn when the left and right arrow keys are pressed.

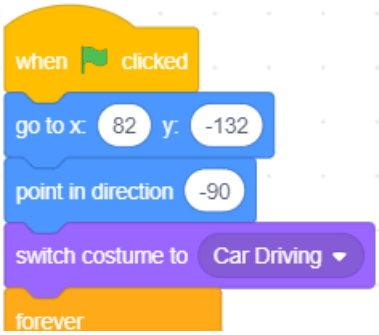
2. Click ⚑ to run the program and then press the ← and → arrows on the keyboard to practice turning the car.
3. Add a **when ⚑ clicked** block
4. From the **Control** category add a **forever** block
5. From the **Motion** category add a **move** block and change the number of steps to **2**.



6. Click ⚑ to test the program. The car will now move and you can steer it with the arrow keys.

7. Press the **stop** button once you have tested it.

Tip In the start block, add a block which sets the program to the **Car Driving** costume to make sure the game always starts with that one.



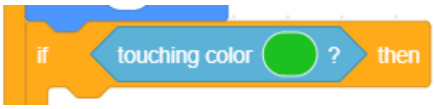
Exercise 5. Handling car crashes

Now we will add some code that tell the program what to do when the car touches the side of the track (crashes). We will tell the program that the car has crashed any time it touches the colour around the edge of the track. That is why it is essential that the border of the track has only one colour. Even if you use other colours to add decorations elsewhere on the stage.

We will add the new blocks to the last **when** **clicked** section that you added (the one that makes the car move).

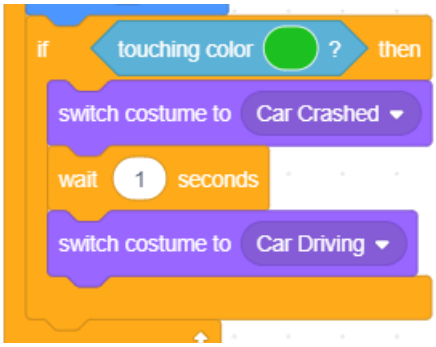
From the **Control** category add an **if then** block inside your **forever** block.

8. From the **Sensing** Category add a **touching color** block inside the **if then** criteria space.
9. Set the colour to a part of the stage that has your track edge colour to set that as the colour the program will be looking for.



Now inside the **if** block we need to tell the program what to do when the car does touch that colour. In this program we will make the car change to its crashed costume and then wait 1 second before reverting to its usual costume and moving again.

Add the blocks shown below



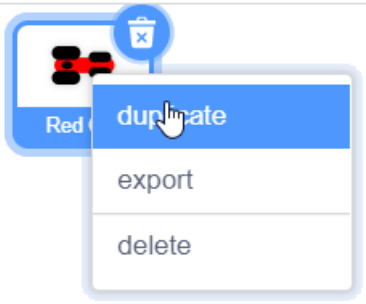
Tip if you had a suitable car crash sound you could add that in there too with a **play sound** block from the **Sounds** category

10. Run the program to test it. When the car crashes it should stop for a second but it can still be turned while it is stopped. The player can straighten it up before it starts moving again.

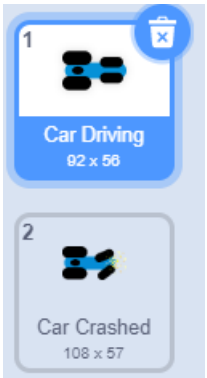
Exercise 6. Adding a Second Car

For the second car we'll make a copy of the existing car sprite and then change a few things.

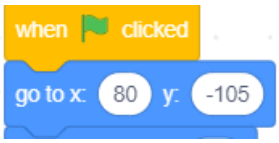
1. **Right click** on the red car sprite and select **Duplicate**.



2. Rename the sprite as **Blue Car**.
3. Edit the sprite costumes so that the red parts are changed to blue.



4. Adjust the **when**  **clicked** block so that the blue car starts next to the red car.



5. Change the left and right arrow keys to z and x so that the second player can use the other side of the keyboard.

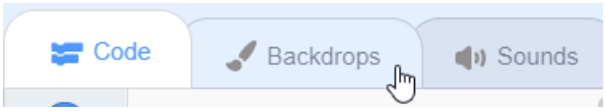
6. Test the program to make sure the blue car drives correctly and changes costume correctly when it crashes in to the side of the track.

Exercise 7. Ending the Race

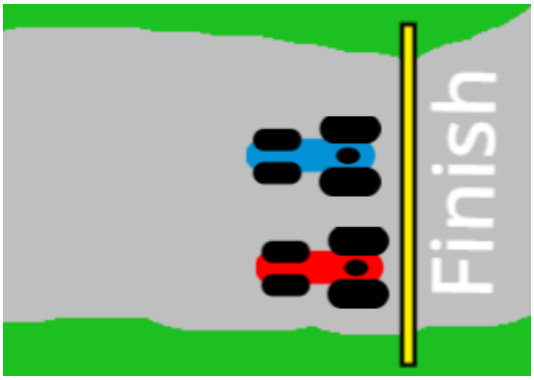
Now we'll add a finish line and add some code that will make the race end when one of the cars touches the colour that the finish line is made of.

1. Click the **Stage** in the sprites area to select it.

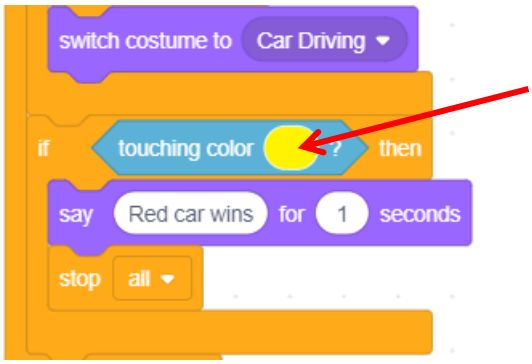
2. Click the **Backdrops** tab



3. Select the Line tool and pick a colour for your finish line. Make sure it is different from the edge of the track. Also make sure it isn't a colour that is used in your cars.
4. Draw a finish line across your track right behind where the cars start. If you had already drawn a line then make sure it is a colour that can be referred to in your code.



5. Select the red car sprite and add the following blocks inside an if block, right beneath the crash blocks.



This should be set to the colour of your finish line

Add similar blocks for your blue car sprite.

Tip You can right click a block you want to copy and select duplicate. E.g. in this situation you can right click the **if** block in the red car sprite and select duplicate to pick up a copy of this block along with the blocks inside it. You can then click the blue car sprite to drop it in there. Then switch to the blue sprite and copy the blocks to where they need to go. All that would be left is editing the text to say blue car wins.

You could also simply drag the blocks you want to copy right on to the sprite in that you want to copy it to in the sprite area.

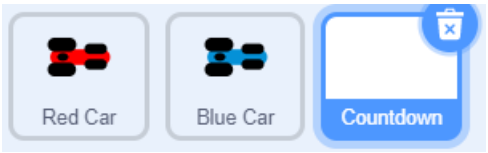
6. Test the program to make sure it shows the win message correctly when one of the cars crosses the finish line.

Tip If you don't feel like driving all the way around the track to test it, while the game is playing you can drag one of the cars so that it is right before the finish line. Then it will drive right over it. Cheating is ok when we're just testing it.

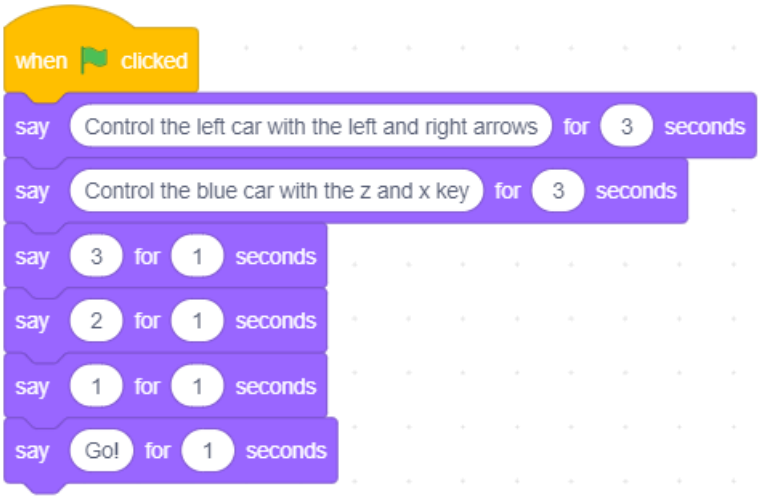
Exercise 8. Adding a Start Count Down

We'll add a count-down so that the players have time to get ready when the race starts.

1. Paint a new sprite and call it **Countdown.** We won't really be painting anything as we will leave it blanks. It will provide us with a sprite to add some messages to though.



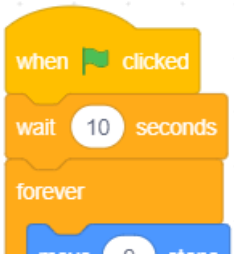
2. Add the following code blocks inside the new sprite.



These blocks will make a series of messages appear one after another.

It will take 10 seconds for all of these messages to appear and we don't want the cars to start moving until we get to the **"Go!"** message.

3. Select the **Red Car** sprite.
4. Add a **wait 10 secs** block before the **move 2 steps** block (before the **forever** block).

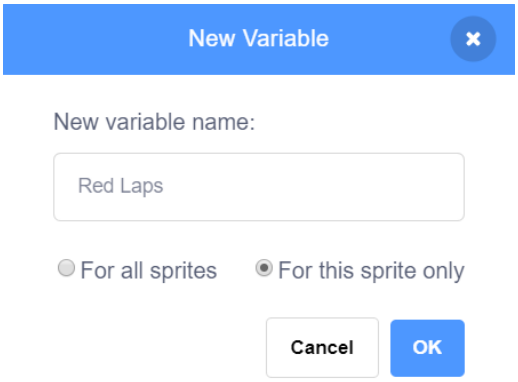


5. Do the same for the **Blue Car** sprite.
6. Test the program to see the messages. The cars should begin moving as soon as the last message (Go!) has displayed.

Exercise 9. Adding Laps

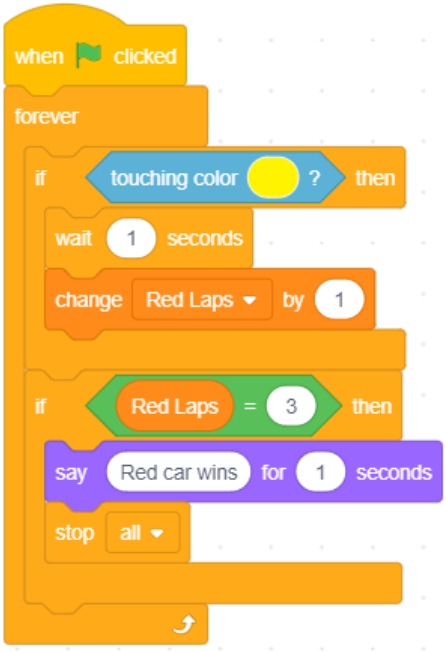
What if we want the race to end after a certain number of laps instead of only 1 lap? We can use a variable to count how many times a car passes the finish line and then finish the game once the variable reaches a certain number. We'll test it with 3 laps but you can set it to any amount you want.

1. Select the **Red Car** sprite.
2. Select the **Variables** category and click **Make a Variable**.
3. Call the new sprite **Red Laps** and set it to **this sprite only**. Click **OK** to create the variable.



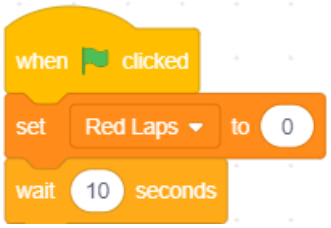
4. Remove the following blocks.

5. Add a set of blocks like the following.



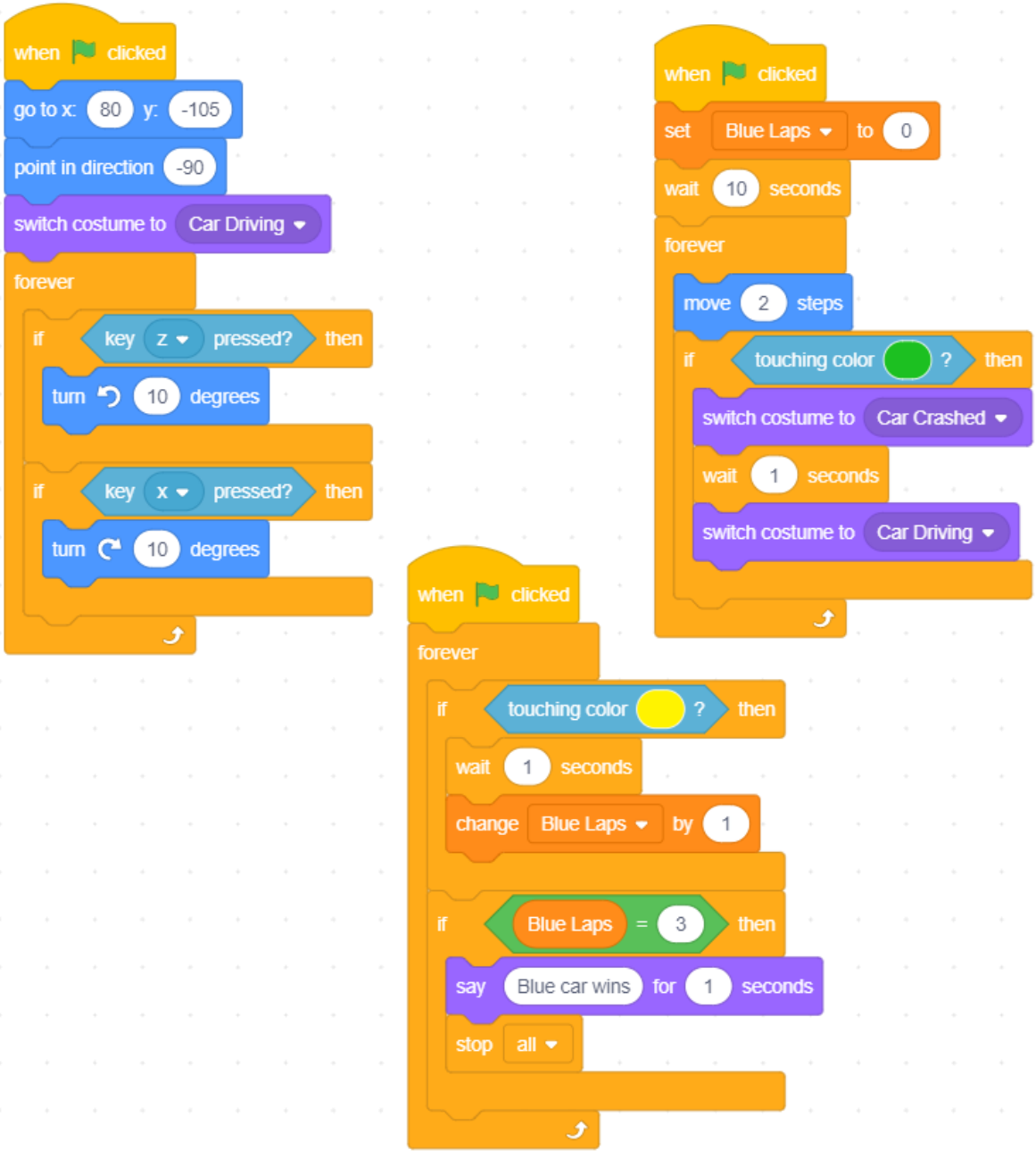
Note The wait 1 secs block is there because it takes time for the car to pass over the line. If we didn't have this block then it would keep on counting laps for as long as the car's touching the line. This makes sure it only counts one lap while it's passing the line.

6. Before the wait 10 secs block add a set Red Laps to 0 block so that the number of laps will always be 0 when the race starts.



7. Repeat these steps to add a lap counter for the Blue car.

The finished code for the Blue Car should look similar to the following.



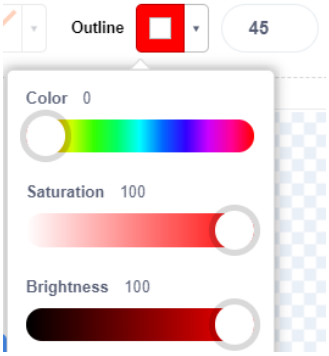
Exercise 10. Extra Things to Try

8. Modify the speed of the cars or how much they turn when you steer them.
9. Add a timer to track how long the race has gone.
10. Add a timer to keep track of the best race time.

Exercise 11. Car Drawing Tips

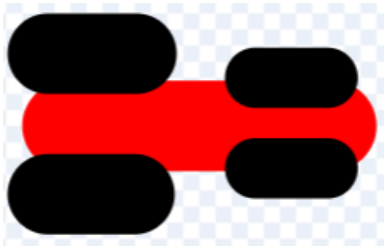
The following steps were used to draw the car shown in these exercises.

1. Make sure you are in vector drawing mode.
2. Click on the **Line** drawing tool.
3. Select the colour for the middle of the car (the options to the right show red)
4. Select a large line thickness. The example to the right shows 45 pixels.
5. Drawn a line for the middle of the car. Holding down **Shift** while you draw will keep the line horizontal.



6. Use the **Select** tool to position it so that it is on the centre of the sprite editing area.

7. Draw short (but thick) lines for the wheels. In the following example the front wheels had 30 pixel thickness while the rear wheels had 40 pixel thickness. The benefit of working in vector mode is that if you don't draw them quite the right length, you can adjust them after you have drawn them by using the Select tool.



8. Add a shape to show where the driver would be sitting in the car. A small oval with a fill colour and no outline colour was used in this example.