

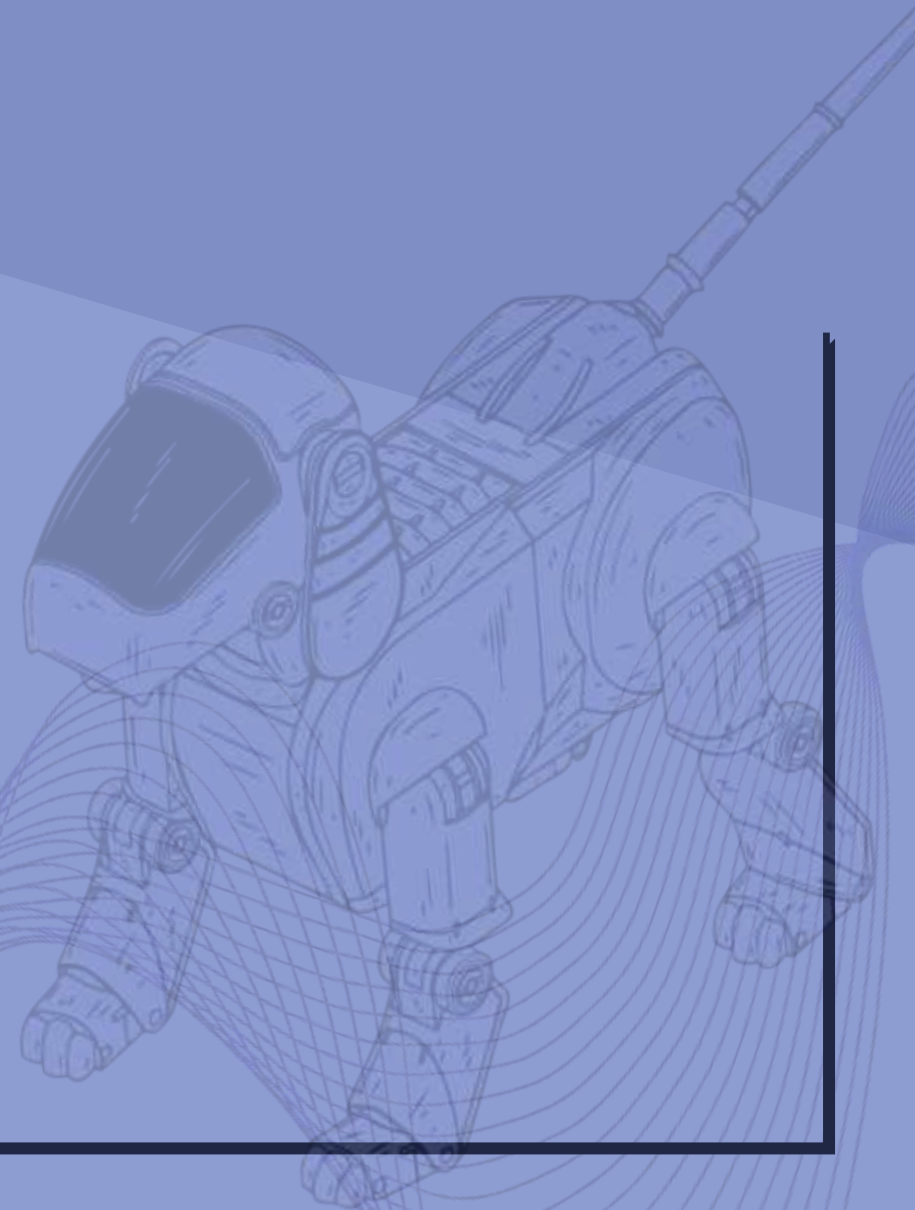
Task

- Games can be a great way of raising awareness about important issues and helping others to preserve the world that we live in!
- You have been tasked with making a **Jungle Racer Game, a two-player racer game!**

Process

Your code should

- Have two-player sprites that can be moved around the screen
- A scrolling screen
- The ability for players to collect gems
- A timer and other mechanics that allows for players to compete!



How does SCRATCH work?

The image shows the Scratch IDE interface. On the left is the 'Code' area with a block palette containing categories like Motion, Looks, Sound, Events, Control, Sensing, Operators, Variables, and My Blocks. The center is the script area with a text editor. On the right is the stage area showing a cat sprite. At the bottom right is the 'Sprite' window, which is highlighted with a dashed purple border and a purple arrow pointing to it from the text below. The 'Sprite' window shows 'Sprite1' with its x and y coordinates, size, and direction.

In Scratch, the way you interact with your program is through **sprites**

Sprites are objects within your program that you can interact with - you can change the way they act using **scripts**

In the template you have been given, many of the sprites have already been given basic scripts, such as the buttons

Sprites can be found in the sprite window here!

How does SCRATCH work?

The image shows the Scratch web interface with several annotations:

- Code Section:** A black arrow points to the left sidebar where code blocks are organized by color-coded categories: Motion (blue), Looks (purple), Sound (pink), Events (yellow), Control (orange), Sensing (light blue), Operators (green), Variables (red), and My Blocks (dark blue). A text box explains: "This is the code section! Here is where we can drag and drop blocks of code. Blocks are colour coded depending on their function".
- Display Window:** A purple arrow points to the large white area in the center-right where the stage is. A text box explains: "This is our display window, where our sprites will appear and interact with the user". The Scratch cat sprite is visible on the stage.
- Sprite Window:** A purple arrow points to the bottom-right panel where sprites are managed. A text box explains: "Sprites can be found in the sprite window here!". This panel shows the selected sprite (Sprite1) and its properties like x, y, size, and direction.

How does SCRATCH work?

The image shows the Scratch web interface. At the top, there is a blue navigation bar with the Scratch logo, a globe icon, and menu items: File, Edit, Tutorials, and an 'Untitled' tab. To the right of the menu are 'Share' and 'See Project Page' buttons. In the top right corner, there is a user profile icon and the name 'itsgracebennett'.

Below the navigation bar is a workspace with three tabs: 'Code', 'Costumes', and 'Sounds'. The 'Code' tab is active, showing a left-hand sidebar with various block categories: Motion, Looks, Sound, Events, Control, Sensing, Operators, Variables, and My Blocks. The 'Motion' category is expanded, showing several blue blocks like 'move 10 steps', 'turn 15 degrees', 'go to random position', 'go to x: 0 y: 0', 'glide 1 secs to random position', 'glide 1 secs to x: 0 y: 0', 'point in direction 90', 'point towards mouse-pointer', 'change x by 10', 'set x to 0', and 'change y by 10'. A black arrow points to the 'Code' tab.

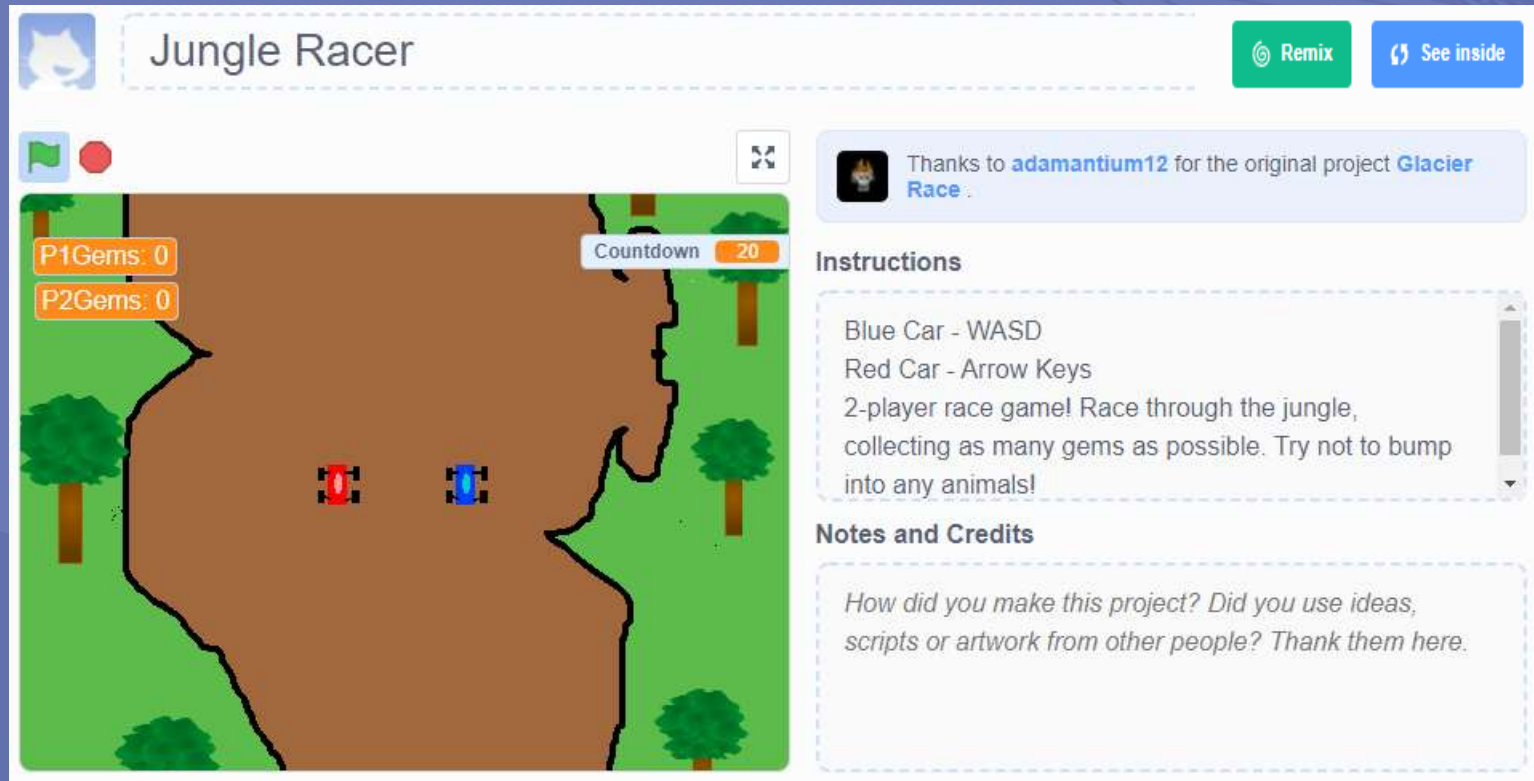
The main workspace is a large white area with a grid. In the center, there is a semi-transparent text box that says: "Use these tabs to switch between code, costumes and backdrops, and sound." Below this, there is another semi-transparent text box that says: "Here we can interact with and customise the backdrops of our game".

On the right side of the workspace is a stage area. At the top of the stage is a small Scratch cat icon. Below it is a larger Scratch cat icon. At the bottom of the stage is a control panel with a 'Sprite' dropdown set to 'Sprite1', 'x' and 'y' coordinates set to 0, 'Show' buttons, 'Size' set to 100, and 'Direction' set to 90. Below the control panel is a 'Backdrops' list with one item labeled '1'. A purple arrow points from the 'Backdrops' list to the text box below it.

At the bottom of the interface, there is a 'Backpack' button.

What our game will look like...





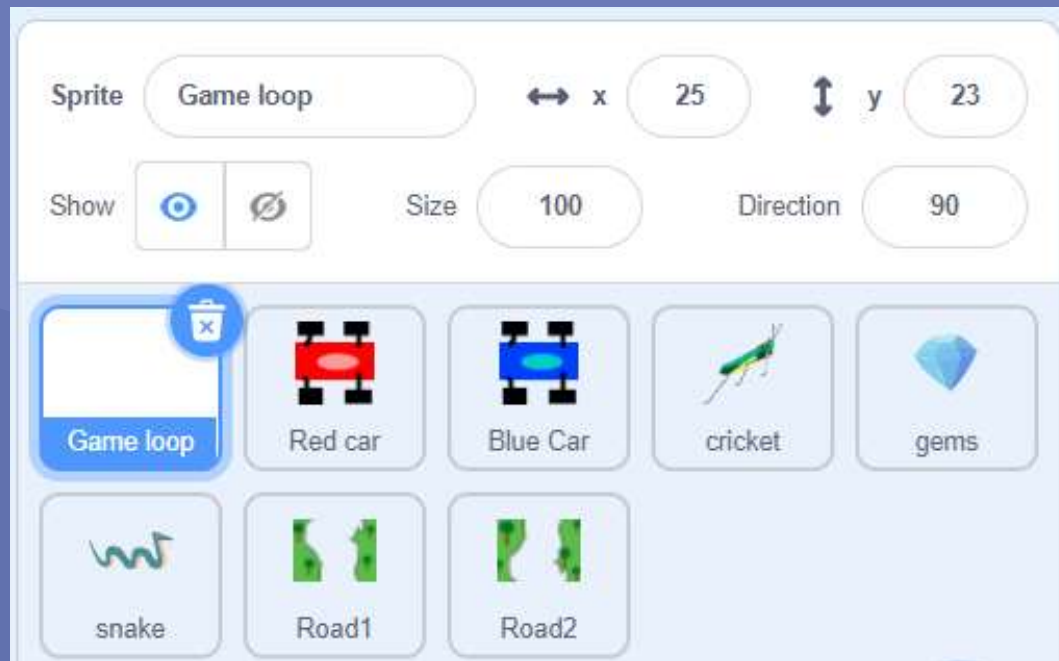
Once you have logged into Scratch...

Click [here](#) to access the template for our project!

Click [Remix](#) to get started with your project

By the end of this resource, your project should look like [this!](#) (Use this if you get stuck)

The following project has lots of different sprites. You can see them all in the sprite menu on the right-hand side!



For this project, you will only be using the two following sprites!



Step 1

Setting up Player 1



In this project, there is an invisible 'Game Loop' sprite that keeps the game moving. For most of the code, we use messages from this sprite to get our code started.

This script will run when the "Setup" message is sent by the Game Loop at the start of the game.

This script sets up the sprite's size, position and direction. We also make sure they don't spin by setting the 'spinning' variable to 0.

```

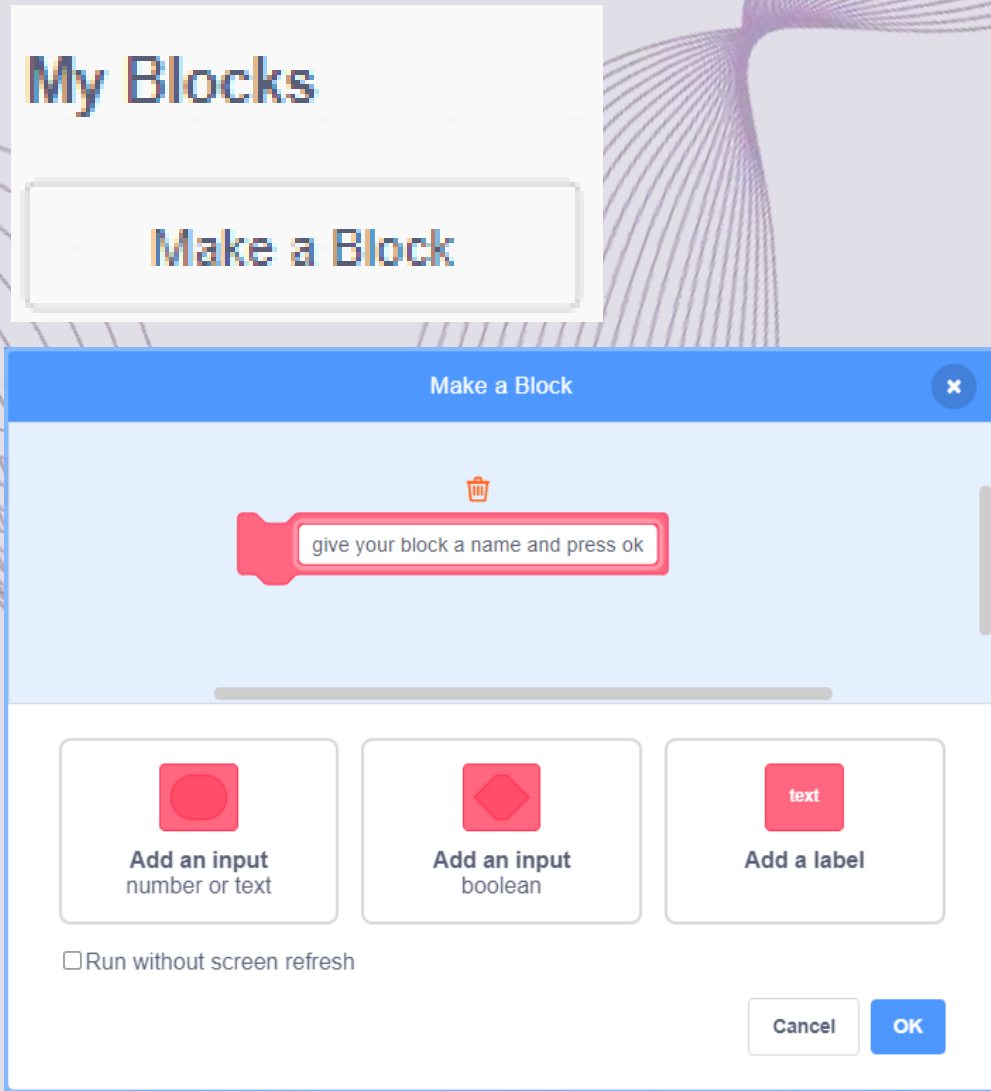
when I receive set up
  set size to 30 %
  show
  go to front layer
  go to x: -40 y: 0
  point in direction 0
  set spinning to 0
  
```

Blocks

Sometimes it's useful to organize our code into 'blocks'. When we have code that we will probably use more than one, by writing it in a block, we can easily access and reuse it at any time!

To make a block, click the 'My Blocks' menu on the left-hand side and click the 'Make a Block' button.

It should come up with a menu like this. Enter a name for your block and press OK!



Step 2

Player 1's Movement



Make a Block called car controls inside of your Red Car sprite called 'car controls'. Add the following code.

Each IF block reacts to the keys that the user presses. For example, if the user pushes the 'W' key, their Y coordinate will be increased (using the car's speed).

This will make it look like the car is driving upwards!

```

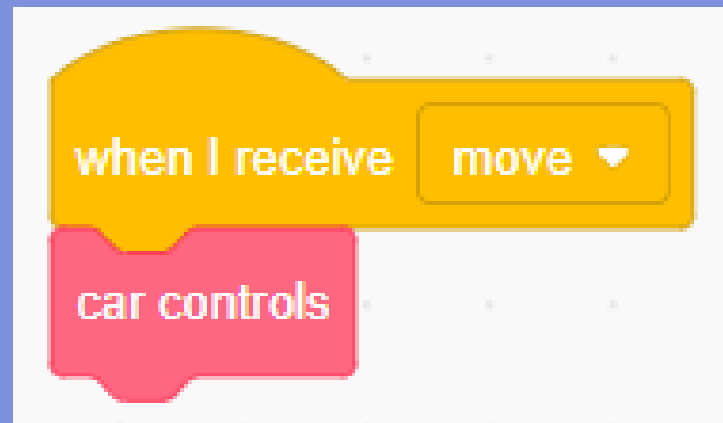
define car controls
  point in direction 0
  if key d pressed? then
    point in direction 30
    change x by CarSpeed
  if key a pressed? then
    point in direction -30
    change x by 0 - CarSpeed
  if key w pressed? then
    change y by CarSpeed
  if key s pressed? then
    change y by RoadSpeed
  
```

Step 2

Player 1's Movement



Before you test it, add this code to your sprite.
This script will run our 'car controls' block many times per second and will allow you to steer your car along the road.



Step 3

Collisions and Spins!



To make the game challenging, let's make our car spin out of control if they come off the road.

For this, we'll make a new block and call it 'check collisions'.

```

define check collisions
  if touching cricket ? or touching Road1 ? or touching Road2 ? then
    set spinning to 70
  if touching Blue Car ? then
    broadcast bounce
  
```

This code checks whether the car is touching a cricket (a bug that will randomly appear in your way) or the sides of the road. If you are, we set the spinning variable to 70 to make the player spin!

This IF block will broadcast a message that we will use later!

Step 3

Collisions with the Blue Car!



This script will move the cars away from each other when they collide. This code will make them bounce away from each other!

Step 4

Spins!



```
define spin
  turn 30 degrees
  change spinning by -1
  change y by RoadSpeed
  if spinning = 0 then
    go to x: -40 y: -180
    point in direction 0
```

To make our car spin, make a new block and call it 'spin'.

It turns the car around and reduces the 'spinning' variable by one. When the variable reaches zero, the spin ends and the car is reset at the bottom of the screen.

Step 5

Updating our screen



Finally, change the existing script triggered by the “move” message to look like this.

This should make your car spin only when you hit obstacles!

```
when I receive move
if spinning = 0 then
  car controls
  check collisions
else
  spin
```


Step 6

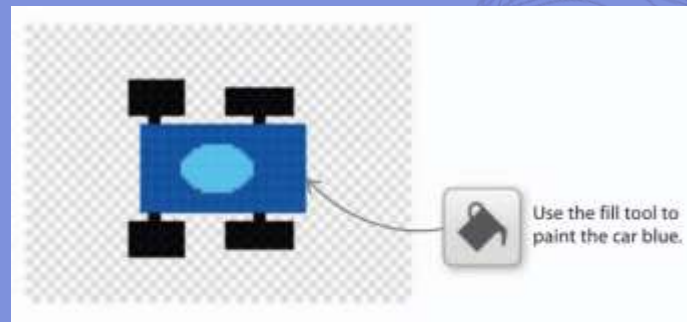
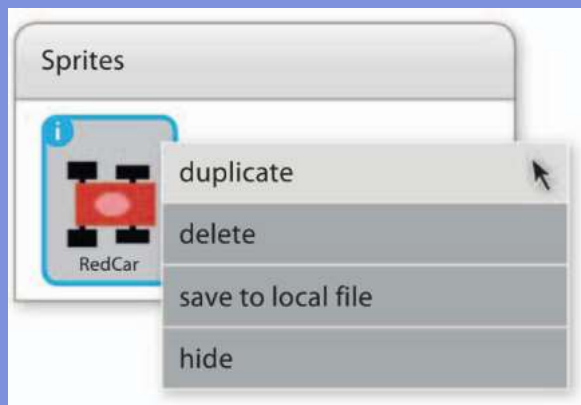
Player 2!



For the Blue Car, you want to repeat steps 1-5 but inside the Blue Car sprite so that they have the same code.

You can do this manually by rewriting the code, or you can duplicate the sprite.

To do this, right-click on the sprite and click 'Duplicate'. Make sure you rename it to 'Blue Car' and change its costume in the 'Costumes' tab. Delete the empty sprite.



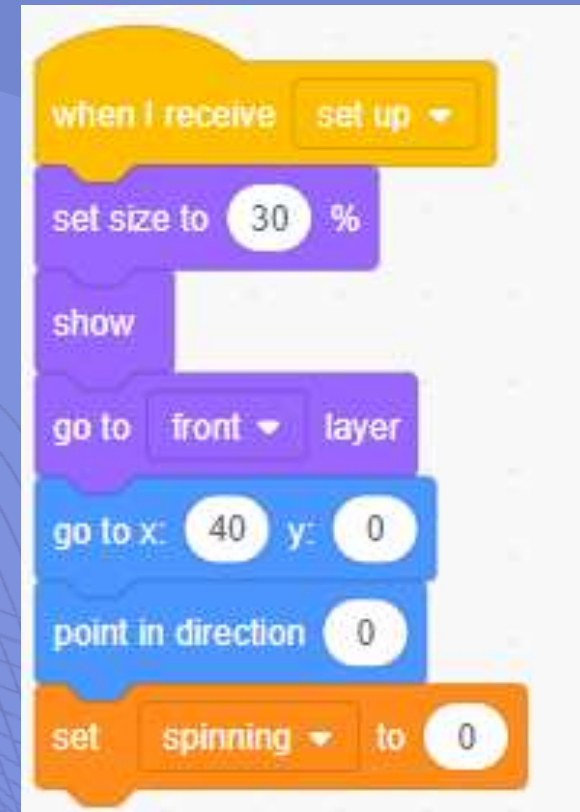
Step 7

Player 2 Setup



Tweak this script so that the positioning is slightly different for the blue car sprite.

Make sure the code in your blue car sprite looks like this.



Step 8

Player 2's Spin



```
define spin
  turn 30 degrees
  change spinning by -1
  change y by RoadSpeed
  if spinning = 0 then
    go to x: 40 y: -180
    point in direction 0
```

You also need to alter the 'spin' block – change that code to the following inside of your blue car sprite.

Step 9

Player 2's Movement



The 'car controls' block code needs to be changed so that the 'key pressed' blocks are now using the arrow keys instead of WASD.

```

define car controls
  point in direction 0
  if key right arrow pressed? then
    point in direction 30
    change x by CarSpeed
  if key left arrow pressed? then
    point in direction -30
    change x by 0 - CarSpeed
  if key up arrow pressed? then
    change y by CarSpeed
  if key down arrow pressed? then
    change y by RoadSpeed
  
```

Step 10

Collisions with the Red Car!



```
define check collisions
  if touching Road1 ?
    set spinning to 70
  if touching Red car ?
    broadcast bounce
```

Step 10

Collisions with the Red Car!



```
define check collisions
  if touching Road1 ? or touching
  set spinning to 70
  if touching Red car ? then
    broadcast bounce
```

```
when I receive bounce
  point towards Red car
  turn 180 degrees
  move 20 steps
  point in direction 0
```

Change the 'check collisions' block inside of the blue car sprite.
Add a bounce script that will move the cars away from each other if they collide.

' block inside of ite.
It move the cars they collide.

```
turn 180 degrees
move 20 steps
point in direction 0
```