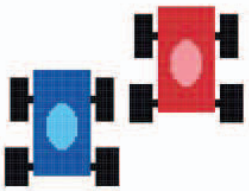
How to build Glacier Race

Glacier Race is a two-player game in which you race up the screen, swerving around obstacles and collecting gems as you go. There's no finish line in this race—the winner is simply the person with the most gems when the time runs out.

AIM OF THE GAME

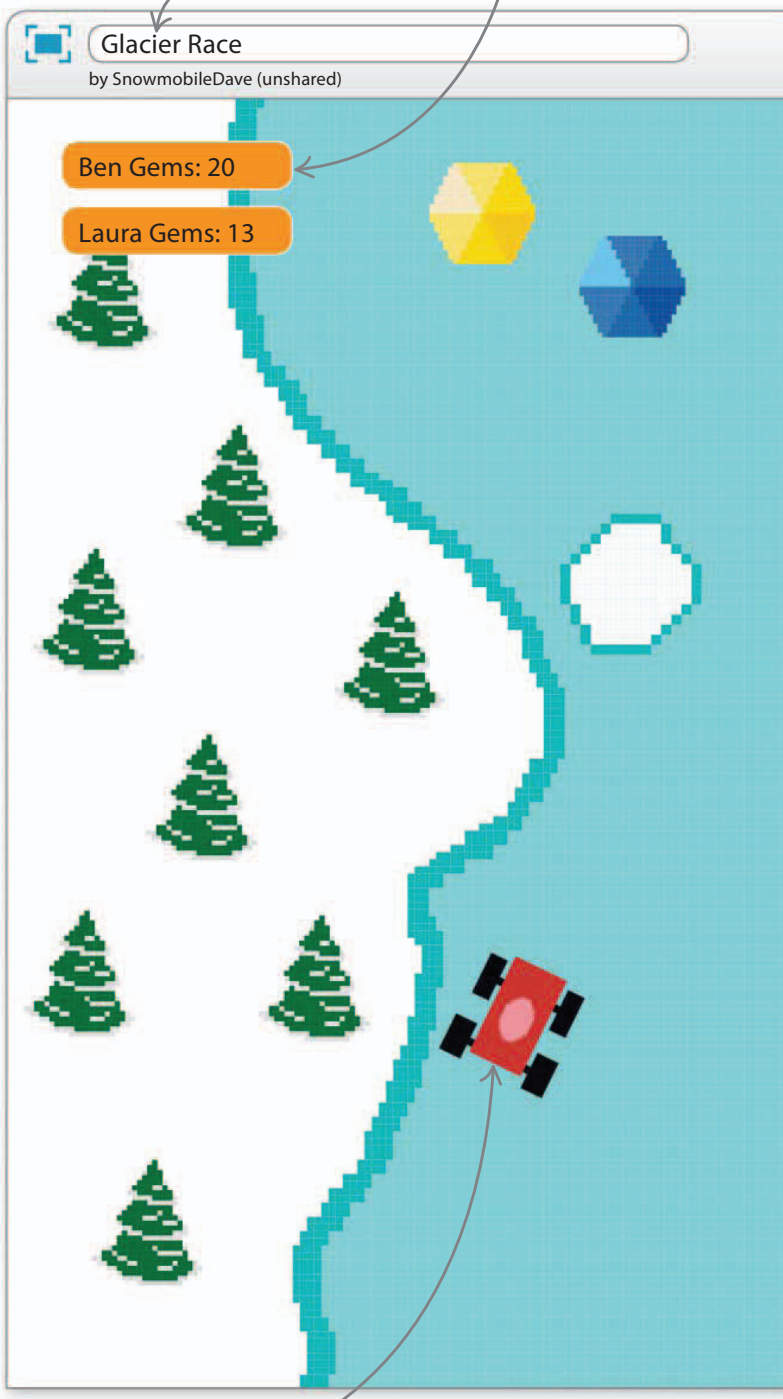
It's red car versus blue car in a race against the clock. Win by collecting more gems than your opponent before the countdown ends. Every gem you grab adds an extra second to the race countdown, but stay clear of the snow or you'll end up in a spin.



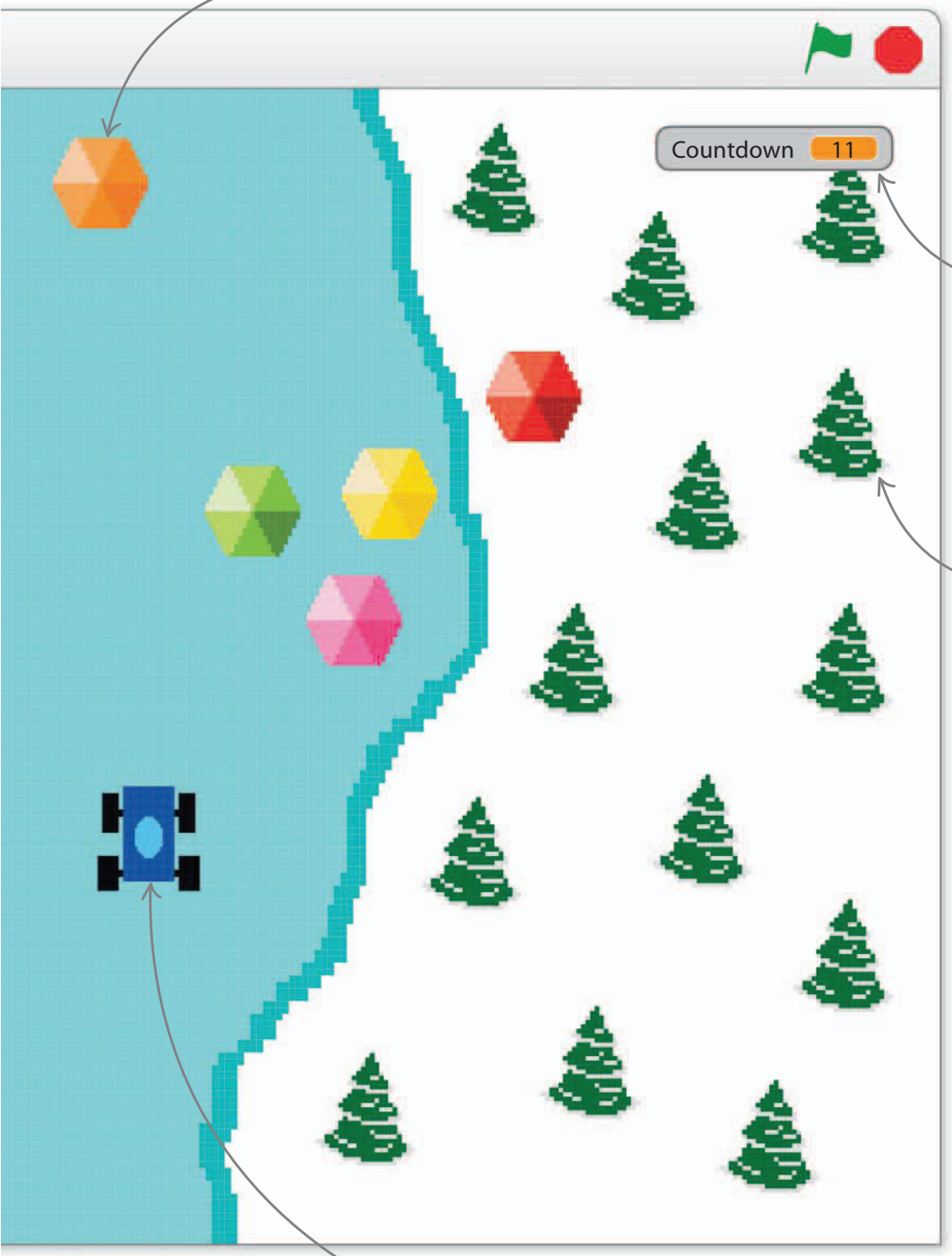
◁ **Cars**
Use the game controls to keep your car on the ice and collect gems. You can also push the other car off the road to gain an advantage.

◁ **Obstacles**
Avoid the giant snowballs and the edge of the road or you'll spin out of control.

◁ **Penguin**
The penguin is the master of ceremonies. He asks the players' names at the start, gives instructions, and announces the winner at the end.



Collect gems to score a point and add a second to the countdown so you can race a little longer.



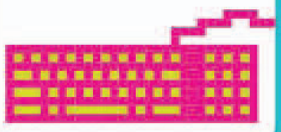
The countdown starts with 20 seconds. When it reaches zero the game ends.

Snowy hills and trees whiz past as the cars race.

The blue car starts on the right and is controlled with the arrow keys.

GAME CONTROLS

Use the arrow keys and the W, A, S, and D keys on the keyboard as game controls.

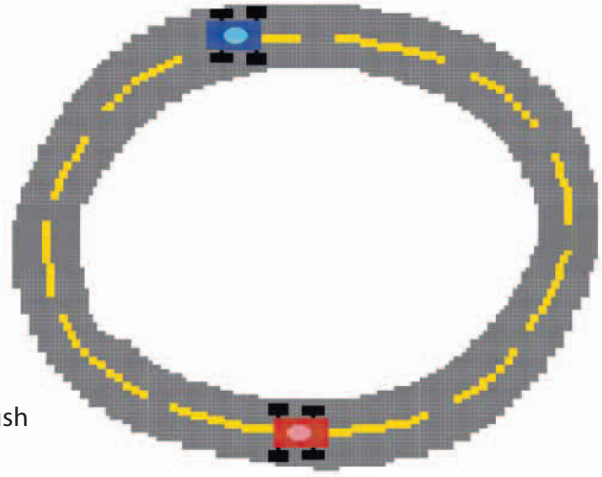


◁ **Icy adventure**

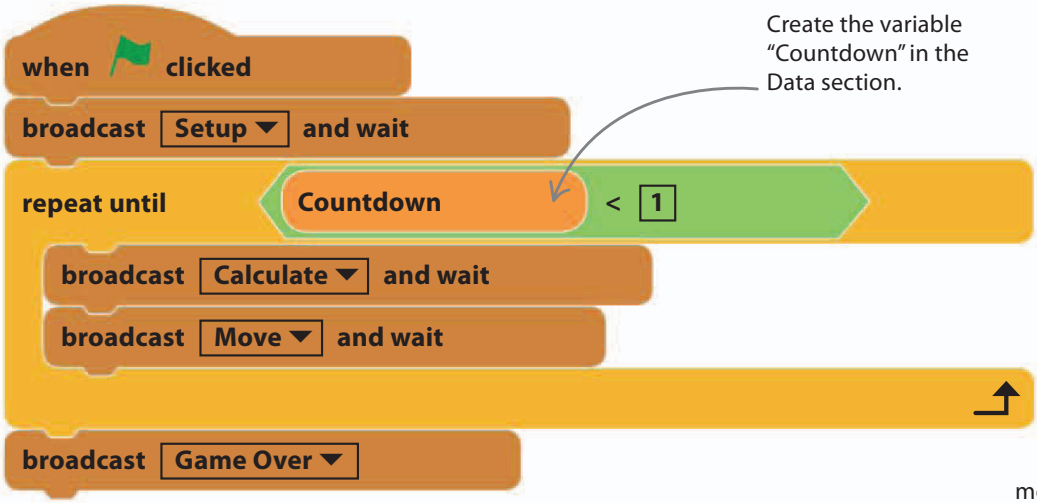
This fast-paced racing game is more fun because you play against an opponent. Challenge a friend or family member to see who can collect the most gems.

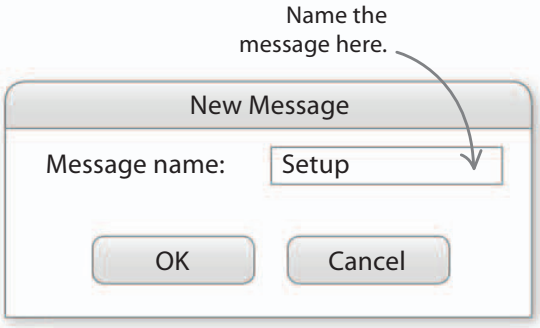
The game loop

Fast games need clever code. This game uses something called a "game loop" to keep all the action happening just when it should. It's as if the game loop bangs a drum, and with each beat all the other sprites move one step. Start by creating a blank sprite to hold the game loop's script.



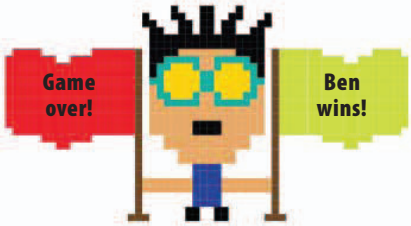
1 Start a new project and delete the cat sprite. Use the paintbrush symbol / to create a blank sprite and rename it "Game Loop". Then make a variable for all sprites called "Countdown" for the game timer and show it on the stage. Build the following script to make the game loop. You'll need to create the messages "Setup", "Calculate", "Move", and "Game Over".





△ How does it work?

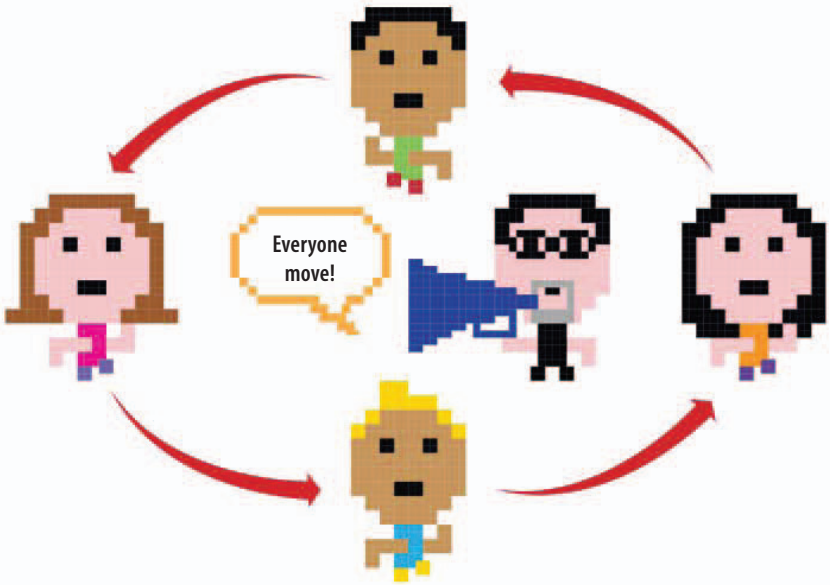
When the project runs, the script sends out a "Setup" message that tells all the sprites to get ready for the game. It waits for them to finish, and then the main loop begins. The loop sends out messages telling every sprite in the game when to run each part of their code. The loop ends only when the countdown reaches zero, at which point the "Game Over!" message is sent so all sprites can perform any final actions and the winner is announced.



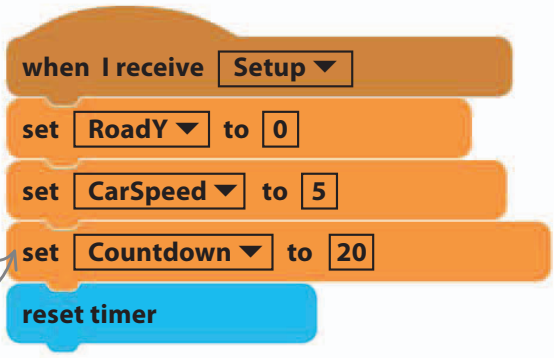
EXPERT TIPS

Game loops

Using one main loop to keep everything in sync is common in computer games. The loop keeps all the sprites in step and makes the code tidy and short. It also helps the game run quickly—in Glacier Race, the game loop runs as fast as 30 times per second. In Scratch, a program with lots of sprites each with their own loops can become slow as the computer has to constantly jump between them. Using a single game loop fixes this problem, but be careful not to use loops elsewhere in the game because they will slow it down.

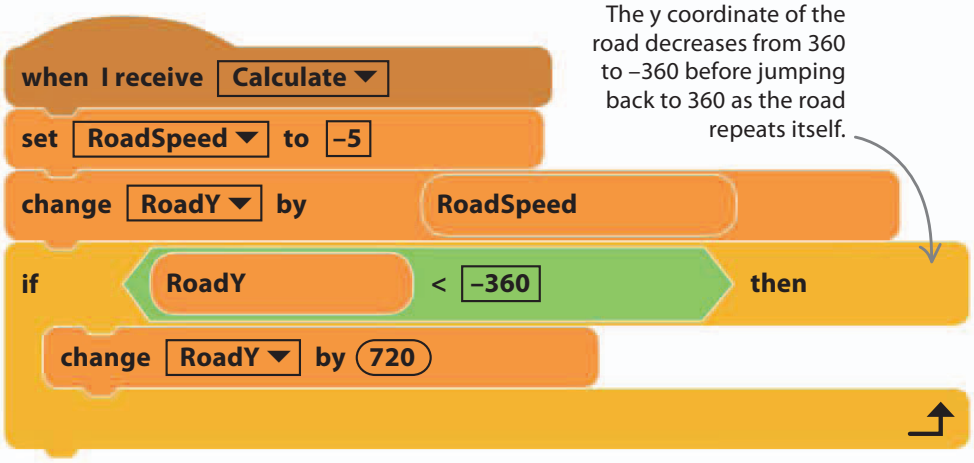


2 Create two new variables for all sprites: "RoadY" (to store the y coordinate used to position our moving scenery) and "CarSpeed" (to set how quickly the cars can move around the stage). Uncheck the boxes in the Data section so they aren't displayed on the stage. Add the script on the right to set the values of the variables at the start of the game.



This block sets the time limit for the game in seconds.

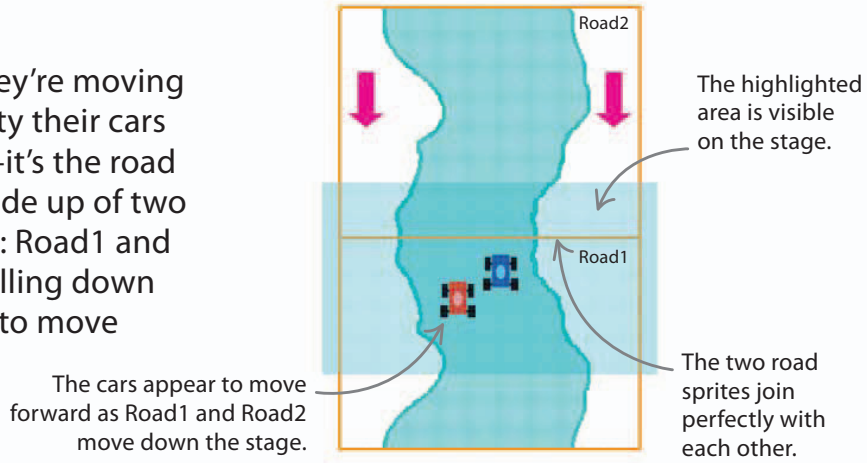
3 Add another variable for all sprites called "RoadSpeed" to store the speed of the moving scenery. Uncheck the box. Then create a script to calculate the position of the road each time the game loop runs. You'll see how this works once you've made the road sprites.



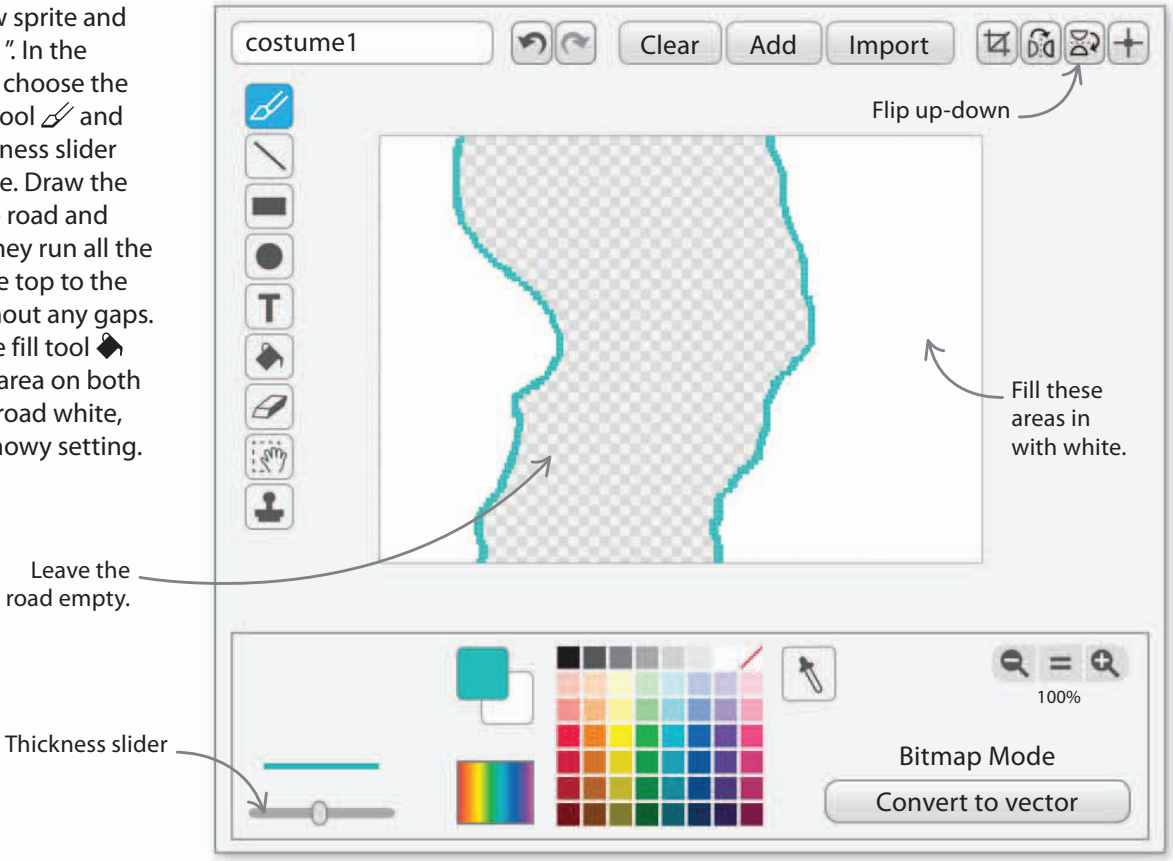
The y coordinate of the road decreases from 360 to –360 before jumping back to 360 as the road repeats itself.

Scrolling road

In Glacier Race, players feel as if they're moving quickly along the road, but in reality their cars don't move very far on the stage—it's the road that moves instead. The road is made up of two sprites that fit together seamlessly: Road1 and Road2. These roads take turns scrolling down the stage, making the cars appear to move faster than they really are.

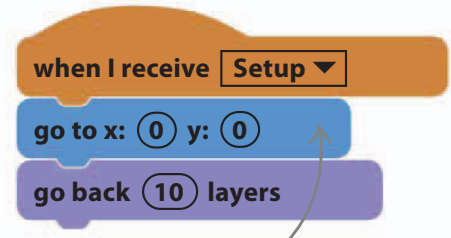


4 Create a new sprite and call it "Road1". In the paint editor, choose the paintbrush tool and set the thickness slider to the middle. Draw the edges of the road and make sure they run all the way from the top to the bottom without any gaps. Then use the fill tool to color the area on both sides of the road white, creating a snowy setting.

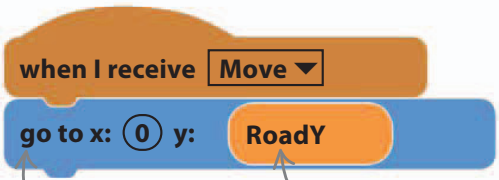


5 Now duplicate the Road1 sprite to make Road2. Select Road2 and go to the Costumes tab. Click on the "Flip up-down" button at the top right and the road costume will turn upside down. The edges of Road1 and Road2 will now match as they are mirror images. They'll look odd on the stage at the moment, but you'll fix that later.

6 Add these scripts to Road1 to get the road moving. They position the road using the "RoadY" variable in the game loop. Try running the project—half the road will scroll down the screen.



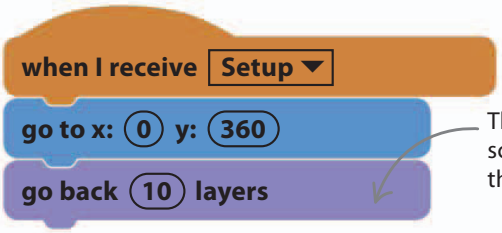
This block makes the game start with Road1 filling the stage.



This block makes Road1 change position when the Game Loop broadcasts the "Move" message.

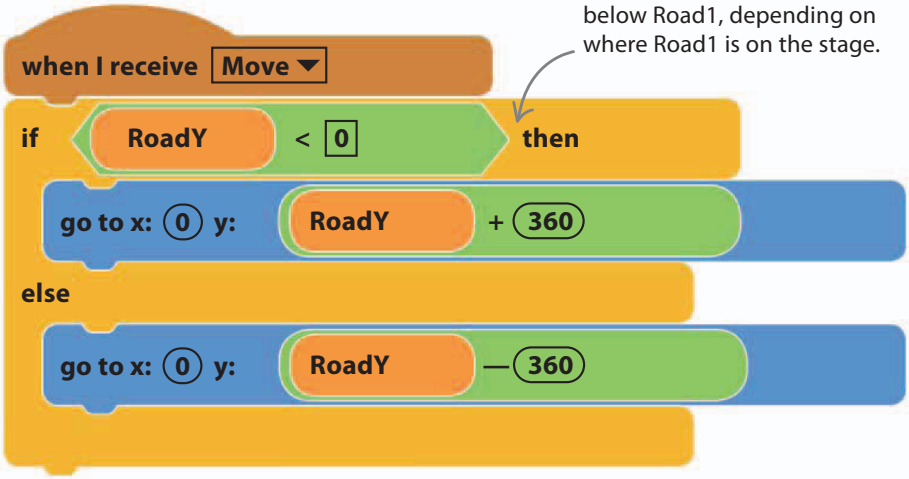
This variable is set in the Game Loop when the message "Calculate" is sent.

7 Now build the following scripts for Road2 to make the second road sprite work together with the first. Run the project—the road should scroll smoothly down the screen.



This makes sure the scenery stays behind the other sprites.

Road2 is positioned above or below Road1, depending on where Road1 is on the stage.



8 To add color to the road, paint the backdrop rather than the sprites, or else the cars will collide with the road surface. Select the stage and click on the Backdrops tab. Use the fill tool to fill it with an icy blue color.

LINGO

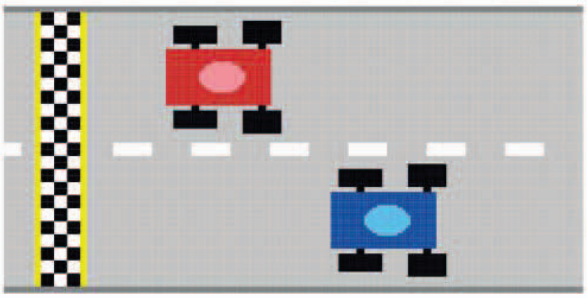
Scrolling

Moving everything on the screen together in the same direction is called scrolling. In Glacier Race, the road scrolls downward. You might have heard of games called side-scrollers, which means the scene moves left or right as the player moves the character on the screen.

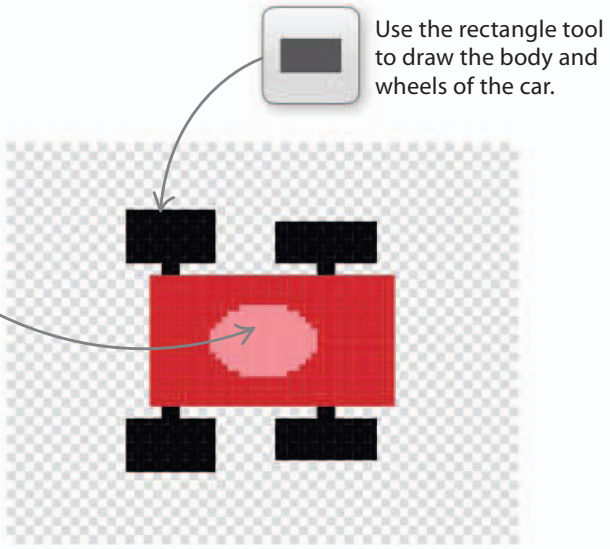
9 Make the scenery more interesting by adding some trees. Select Road1 and click on the Costumes tab. Click the "Add" button on top and add the tree costume. Shrink it by using the selection box and place it on the snow. Add as many trees as you like. Repeat the process for Road2.

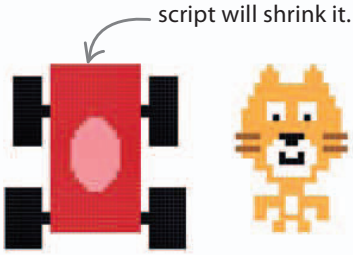
Racecars

Now it's time to add the racecars. Once you've got one car moving, you can duplicate it to make the second one and save yourself a lot of work.

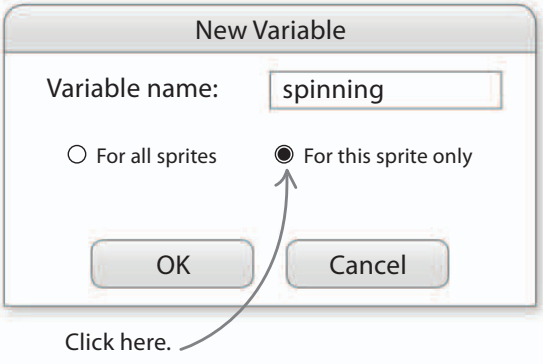


10 Click the sprite symbol and load Cat1 from the library—you can use this sprite to ensure the car is the right size. Now open the paint editor and click on "Convert to bitmap". Use the rectangle and circle tools to draw a car like the one shown here. Make sure you draw the car facing right or it will point the wrong way in the game. Remember to delete the cat image once you've finished and use the "Set costume center" tool to center the car.

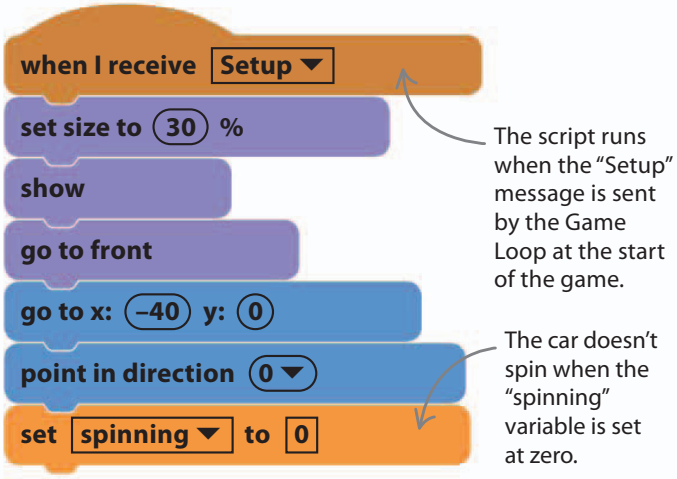




11 Rename the sprite "RedCar" in the sprites list. Then create a new variable, "spinning", which you'll use later to say when a car is in a spin. Note that for this variable, you need to select the option "for this sprite only" and uncheck the box in the Data section so that the variable doesn't show on the stage.



12 Remember that in this project, sprites can run scripts only when they get messages from the Game Loop. Add the following script to set up the red car at the start of the game.



13 You now need to add keyboard controls for the car. Choose More Blocks in the blocks palette and then click on "Make a Block". Create a new block called "car controls" and add this script to its "define" block.

The car usually points straight up the screen.

This block moves the car sideways.

This block makes the car turn a little to the left.

This block moves the car up the stage.

This block makes the car turn a little to the right.

This block makes the car appear to stop by moving it down the stage at the same speed as the road.

```
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
```

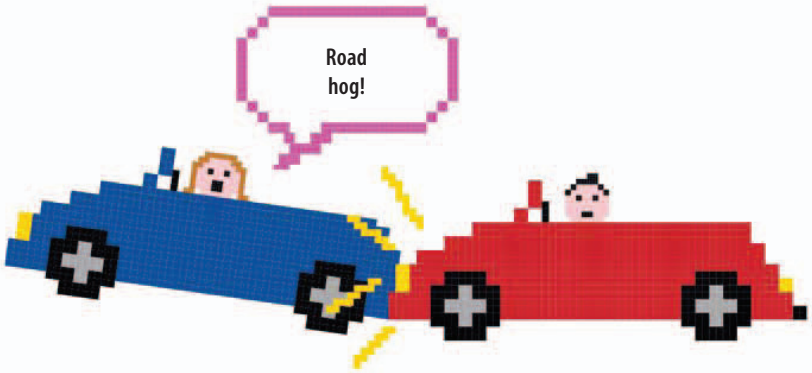
14 Add a script to run the "car controls" block when the car receives the message "Move" from the Game Loop. Run the project. You should now be able to steer the red car along the road using the keys W, A, S, and D.

```
when I receive Move
car controls
```

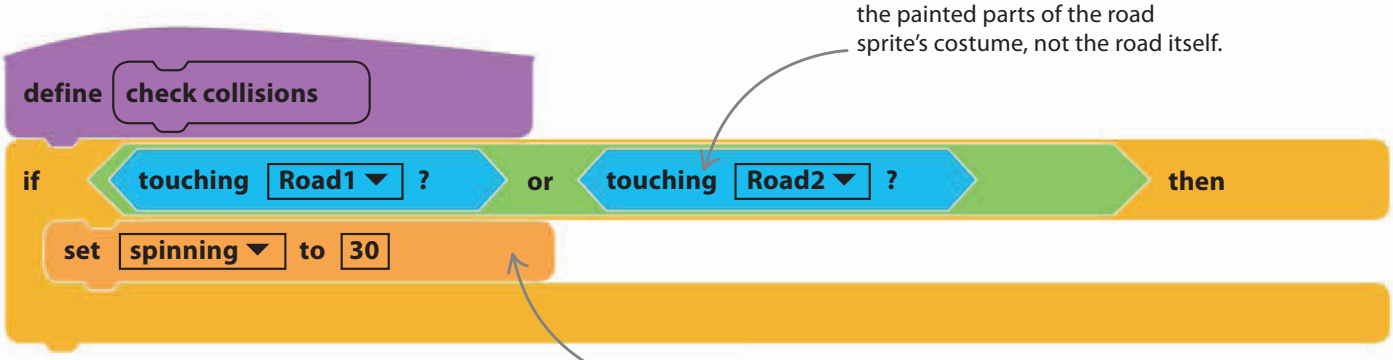
The "Move" message is sent by the Game Loop many times per second.

Collisions and spins

To make the game challenging, you can force players to avoid the snow by making their cars spin out of control if they touch it. You need to create some more new blocks to make this work.



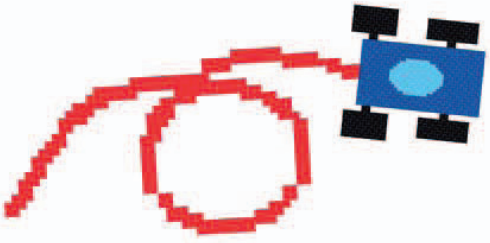
15 With RedCar selected, create a new block to detect the snow. Choose More Blocks in the blocks palette and then click "Make a Block". Name the block "check collisions" and create the following script.

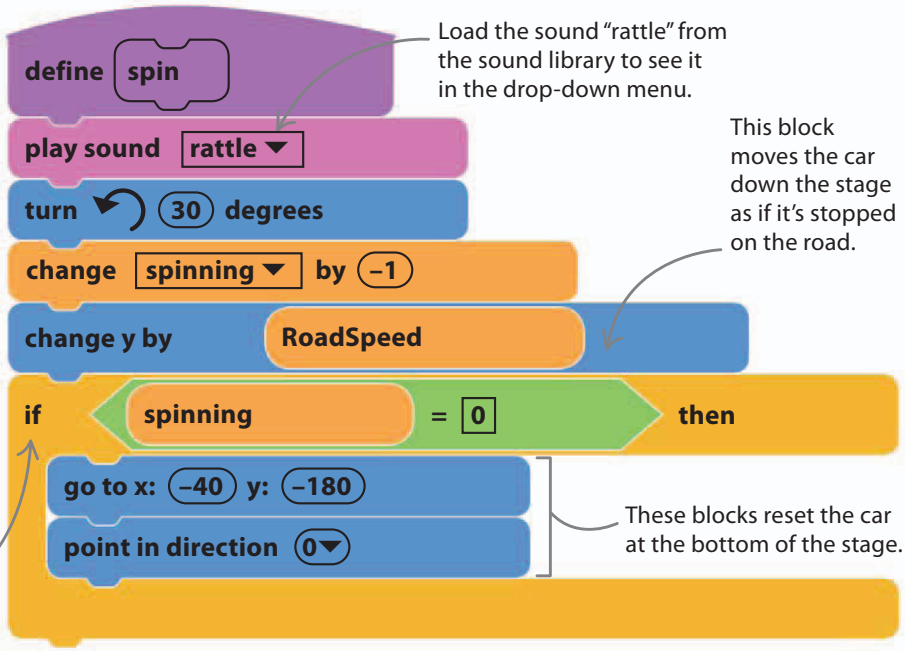


The "touching" block only detects the painted parts of the road sprite's costume, not the road itself.

This block tells the car how long to spin for.

16 Now create another block, call it "spin", and add the script shown here. The "spin" block runs when the car is spinning. It turns the car round 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 stage.





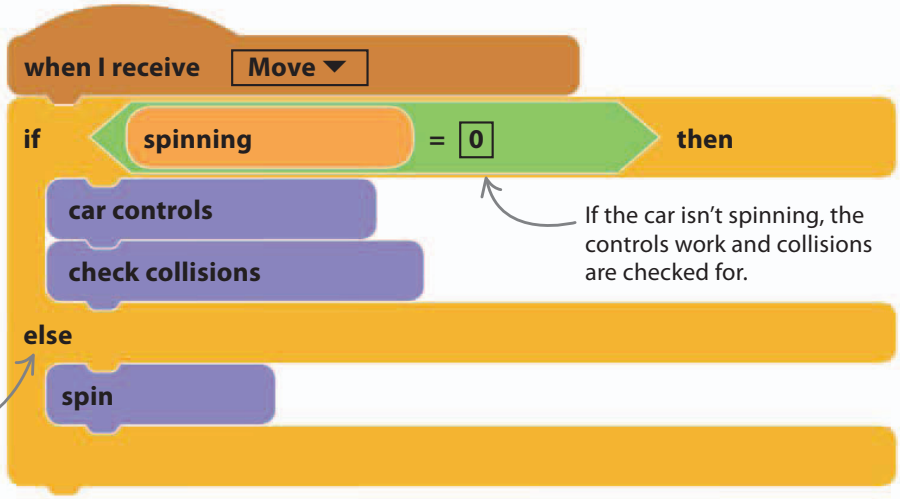
Load the sound "rattle" from the sound library to see it in the drop-down menu.

This block moves the car down the stage as if it's stopped on the road.

These blocks reset the car at the bottom of the stage.

This block checks if the spin is over.

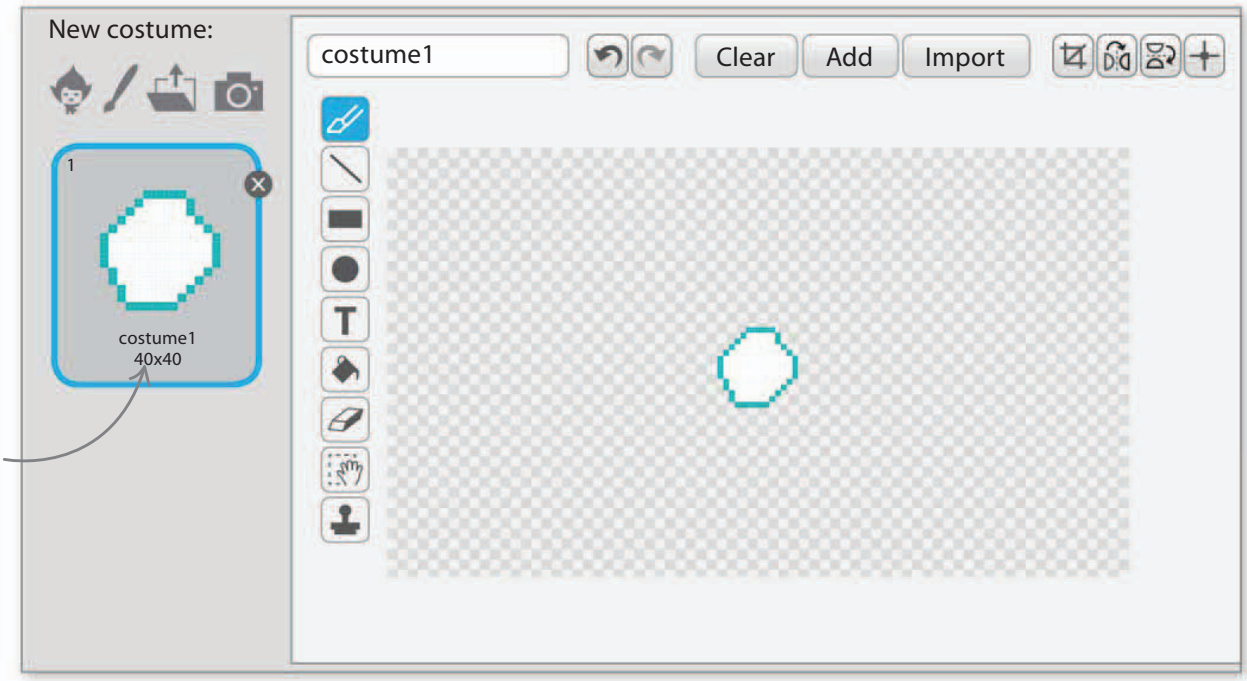
17 Finally, change the existing script triggered by the "Move" message to look like the one shown here. Now you can only control the car if the "spinning" variable is zero. Collisions are checked only when you're not in a spin—otherwise you'd spin forever! Run the game. The car should spin if it hits the snow.



If the car isn't spinning, the controls work and collisions are checked for.

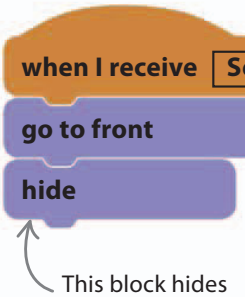
If the value for "spinning" is more than zero, the car will spin.

18 To add some snowball obstacles, create a new sprite in the paint editor. Make it about the size of the car on the stage. To get the correct size, watch it appear on the stage after you've drawn it. You can also see the costume's size in the costume list—aim for about 40x40. Name the new sprite "Snowball".

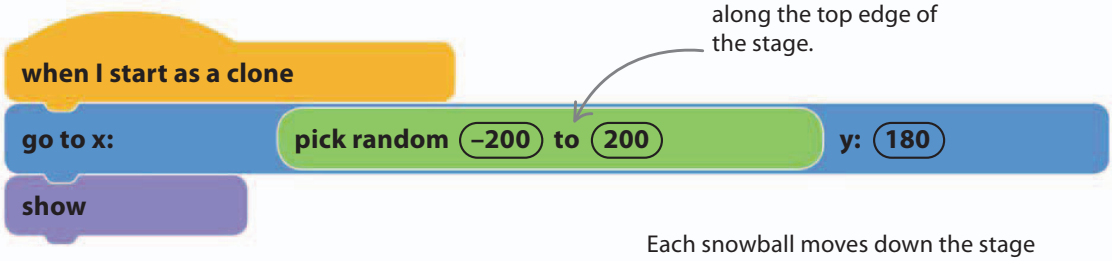


These numbers show the costume's size.

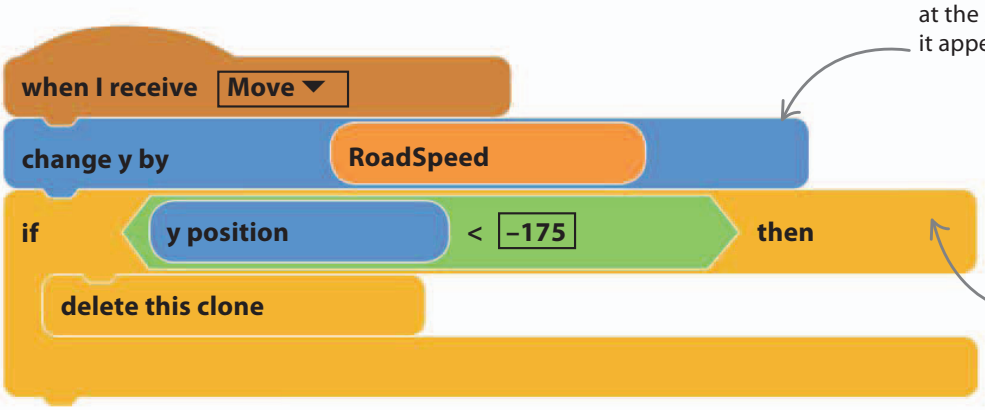
19 Add the following three scripts to the Snowball sprite. The Snowball sprite is cloned to make lots of obstacles, but you might notice that there's no "create clone" block here. The clones will be created by the Game Loop sprite, using some code that we'll add next.



This block hides the original sprite so that you only see the clones.



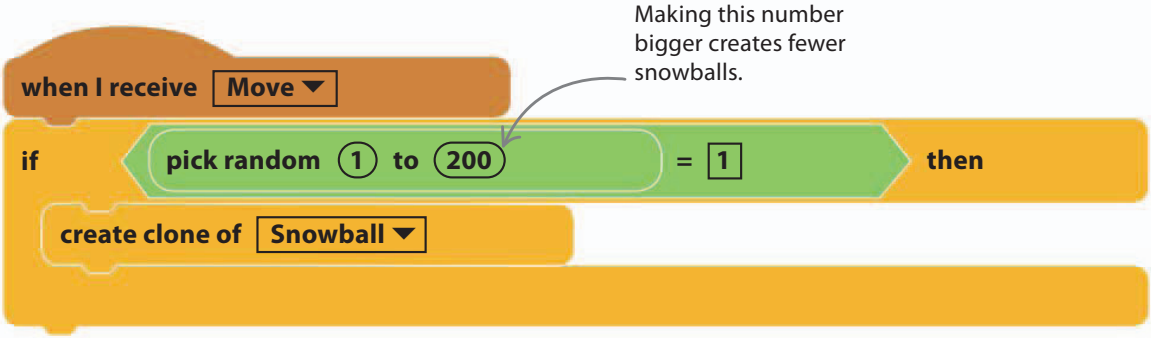
The snowball clone starts at a random point along the top edge of the stage.



Each snowball moves down the stage at the same speed as the road, making it appear stationary.

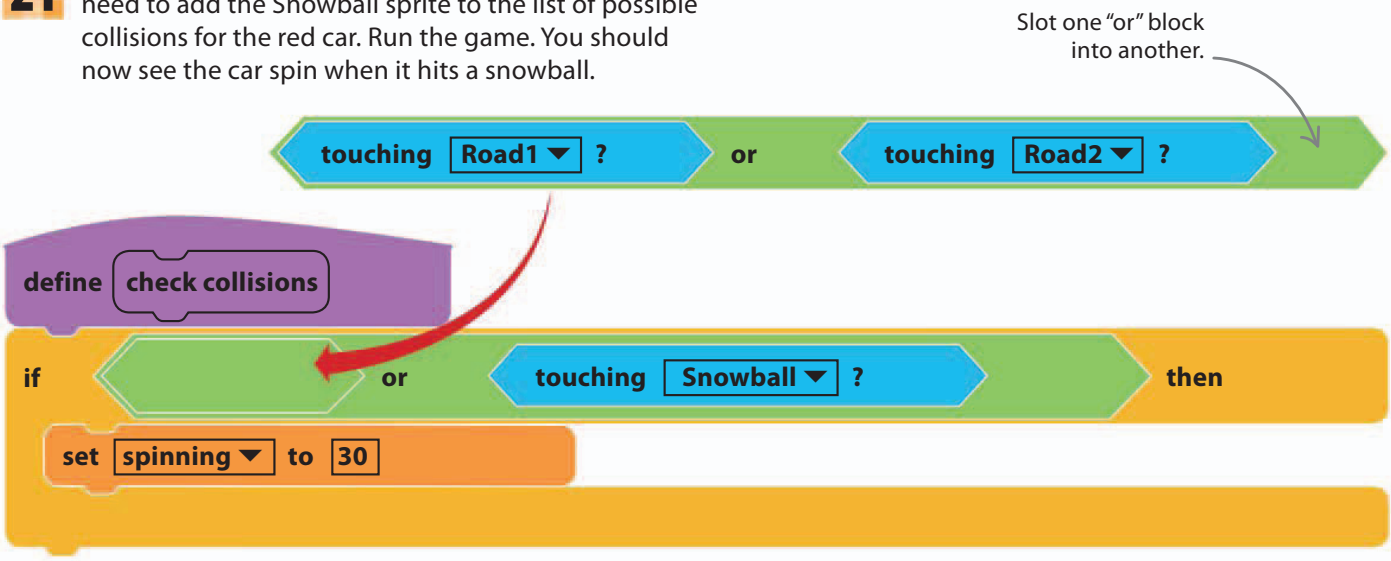
The snowball disappears when it reaches the bottom of the stage.

20 Now select the Game Loop sprite and add this script to make a new snowball appear with a chance of one in 200 every time the loop repeats.



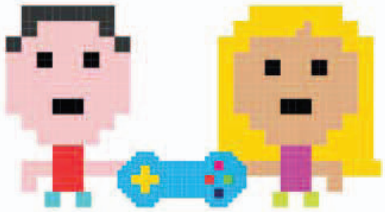
Making this number bigger creates fewer snowballs.

21 To make the car spin when it hits a snowball, you need to add the Snowball sprite to the list of possible collisions for the red car. Run the game. You should now see the car spin when it hits a snowball.

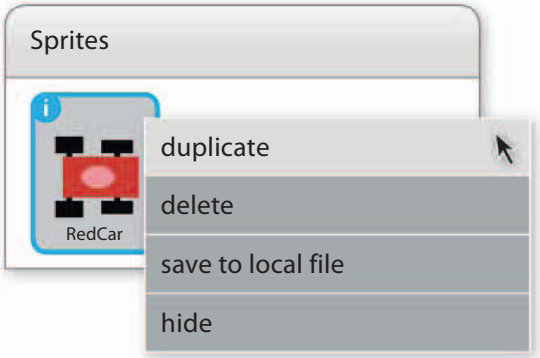


Player two

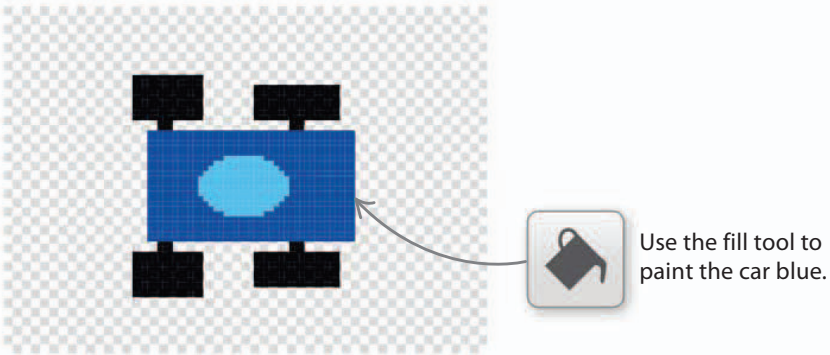
You now need to create the second player's car. Doing this is easy—you simply copy the first car, recolor it blue, and tweak the scripts.



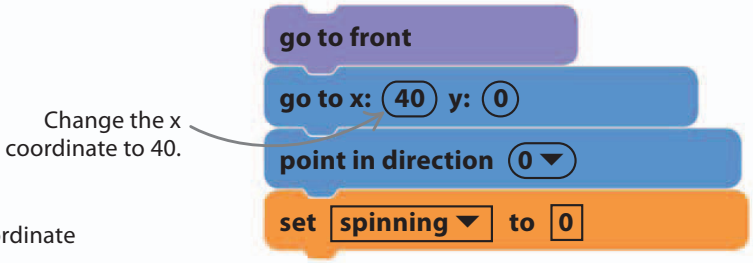
22 Duplicate the RedCar sprite and name the copy "BlueCar". Note that the duplicate sprite gets its own copy of all the scripts. This includes a copy of the "spinning" variable (set to "for this sprite only"), which can be different from the red car's.

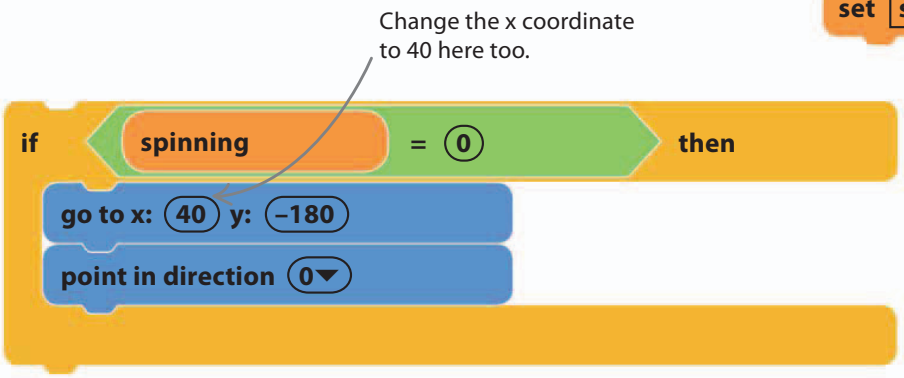


23 Select the BlueCar sprite and click on the Costumes tab to open the paint editor. Use the fill tool to change the color of the car.

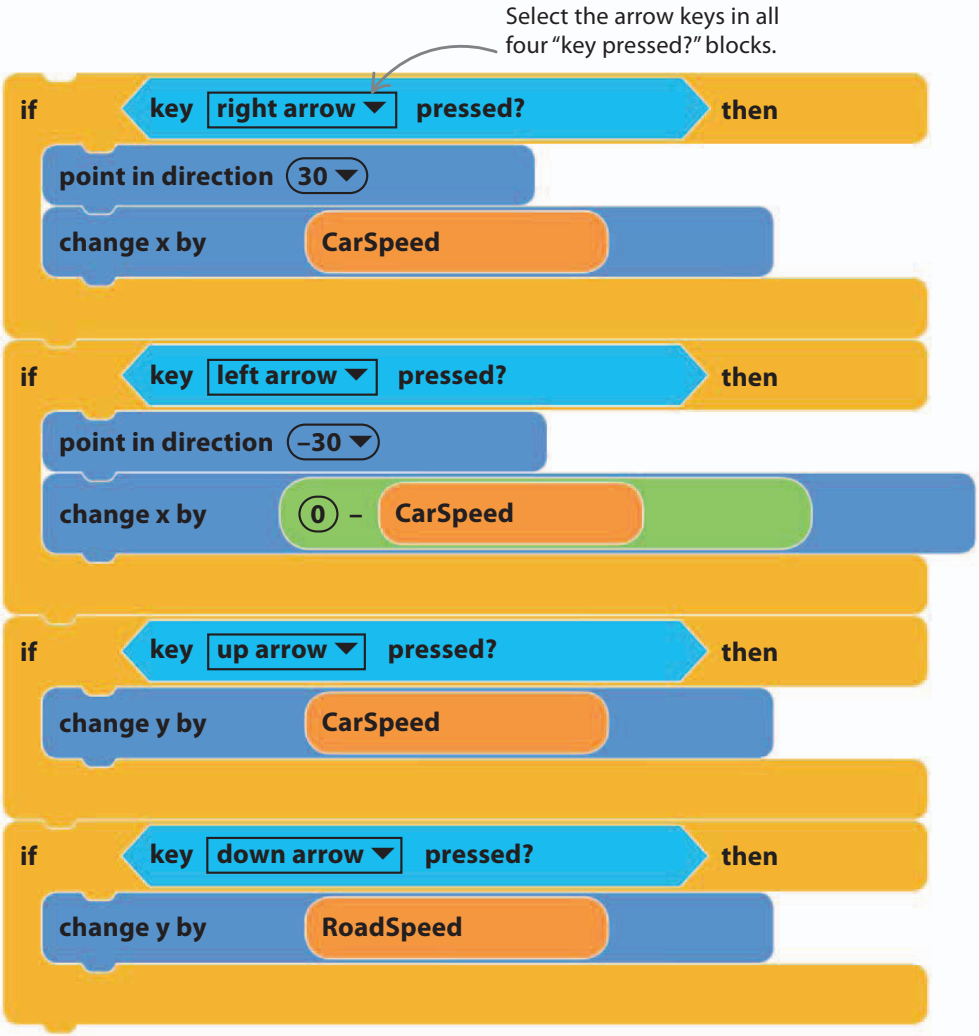


24 Now select the Scripts tab to see BlueCar's scripts. Change the x coordinates in its "go to" blocks to 40 in both the "Define spin" script and the "When I receive Setup" script. This makes the blue and red cars start next to each other.





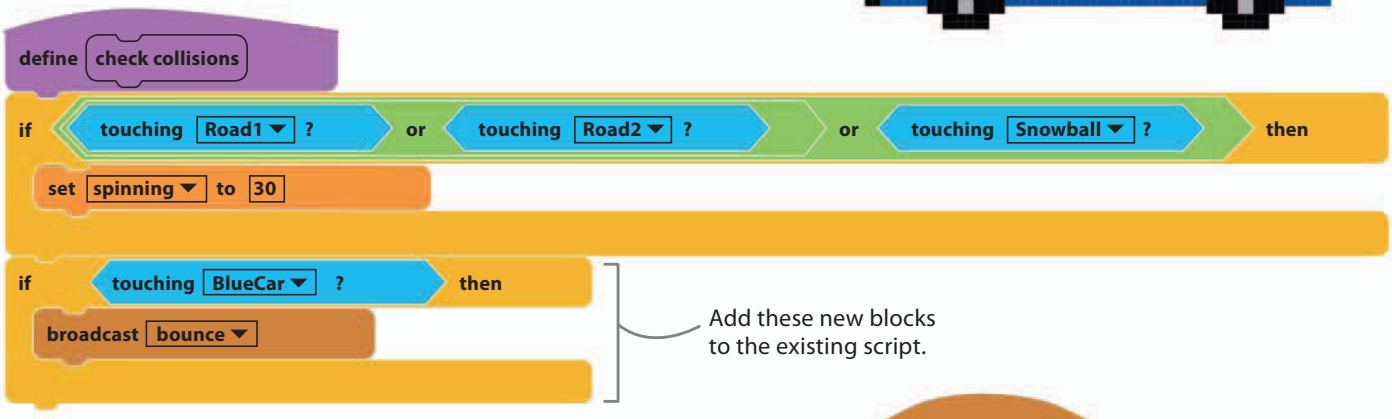
25 In the "Define car controls" script, change the "key pressed" blocks so that the blue car can be steered using the arrow keys on the keyboard. Then run the game. Both the cars should race along the track, but they can drive through each other at the moment.



▷ **Change the script**

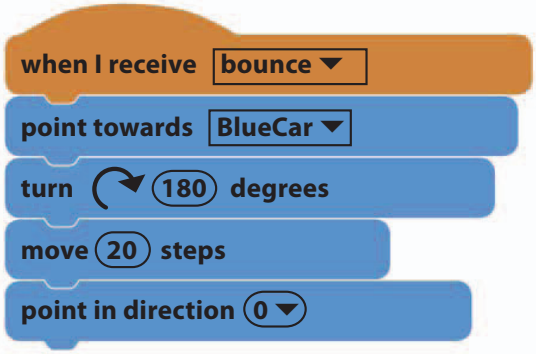
In the "key pressed?" blocks, replace key "d" with "right arrow", key "a" with "left arrow", key "w" with "up arrow", and key "s" with "down arrow".

26 To stop the cars driving through each other, you need to make them sense each other and then bounce apart. Add a new "if then" block to RedCar's "Define check collisions" script as shown here. Create the message "bounce", and then add a new script to make RedCar move away from BlueCar when it receives the message.

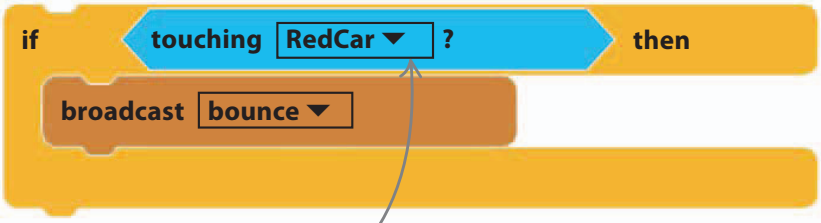


Add these new blocks to the existing script.

This new script makes RedCar bounce away from BlueCar.

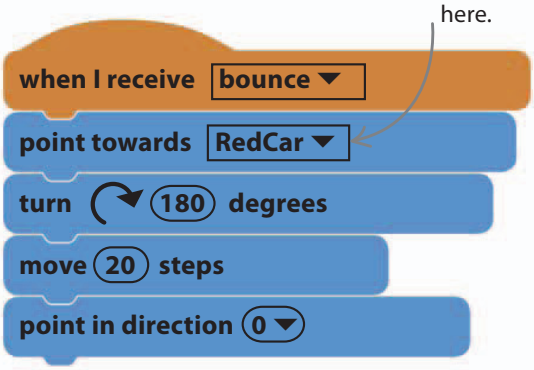


27 Now make the same changes to BlueCar's scripts so it can sense when it touches RedCar and bounce. Run the game to check the cars bounce when they collide.



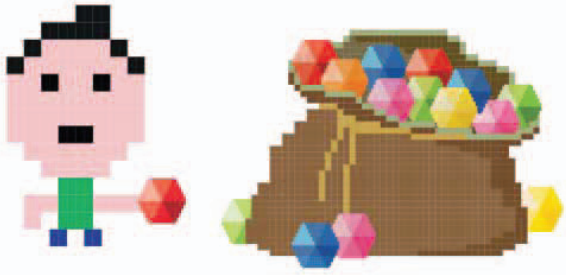
This time the "touching" block checks for collisions with the red car.

Choose RedCar here.

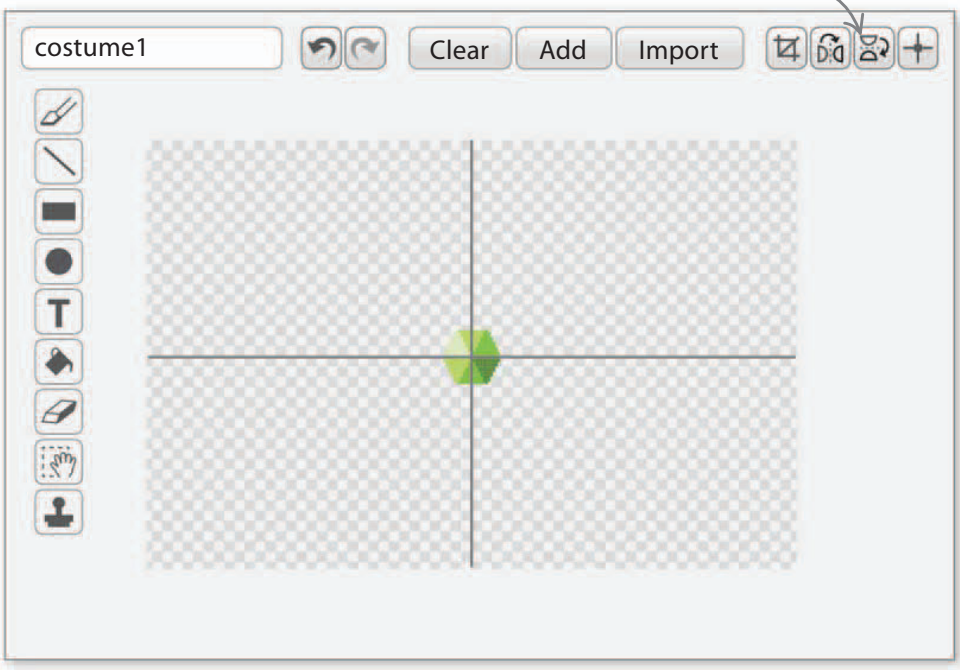


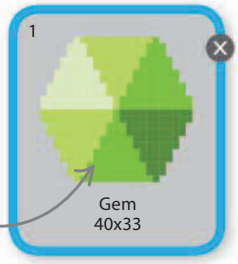
Collecting gems

The next step is to create the colorful gems that the players battle to collect. Each gem will be a clone of a single gem sprite, which makes it easy to put lots of gems on the stage at once.



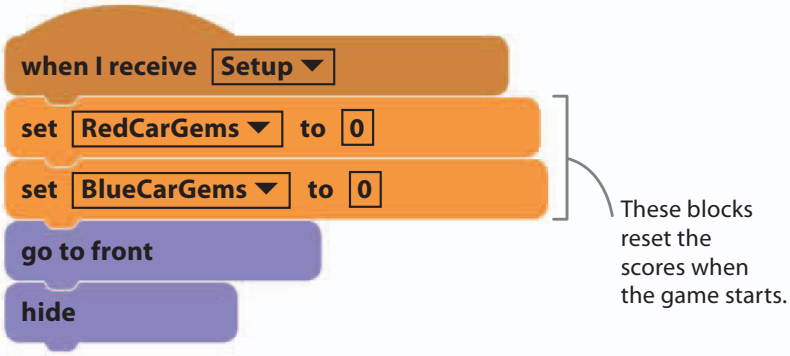
28 Click the paintbrush symbol in the sprites area to create a new sprite with the paint editor. To create a gem, use the line tool to draw six triangles arranged in a hexagon. Fill each one with a different shade of green. Make it similar in size to the snowball and center it when you've finished.

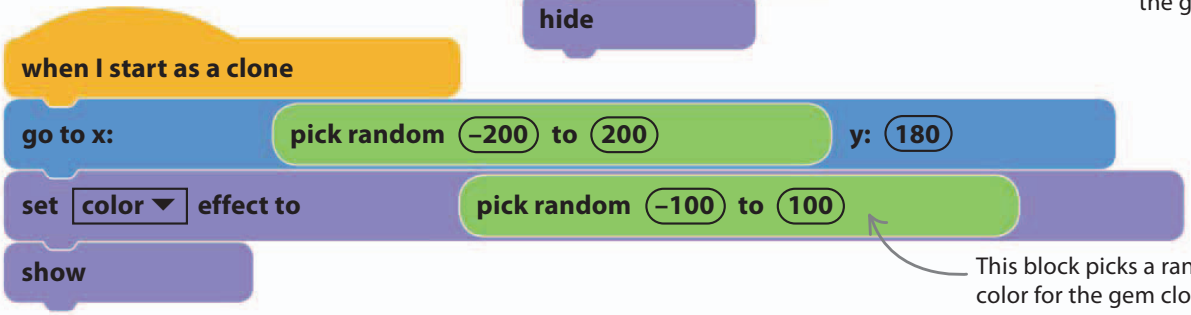




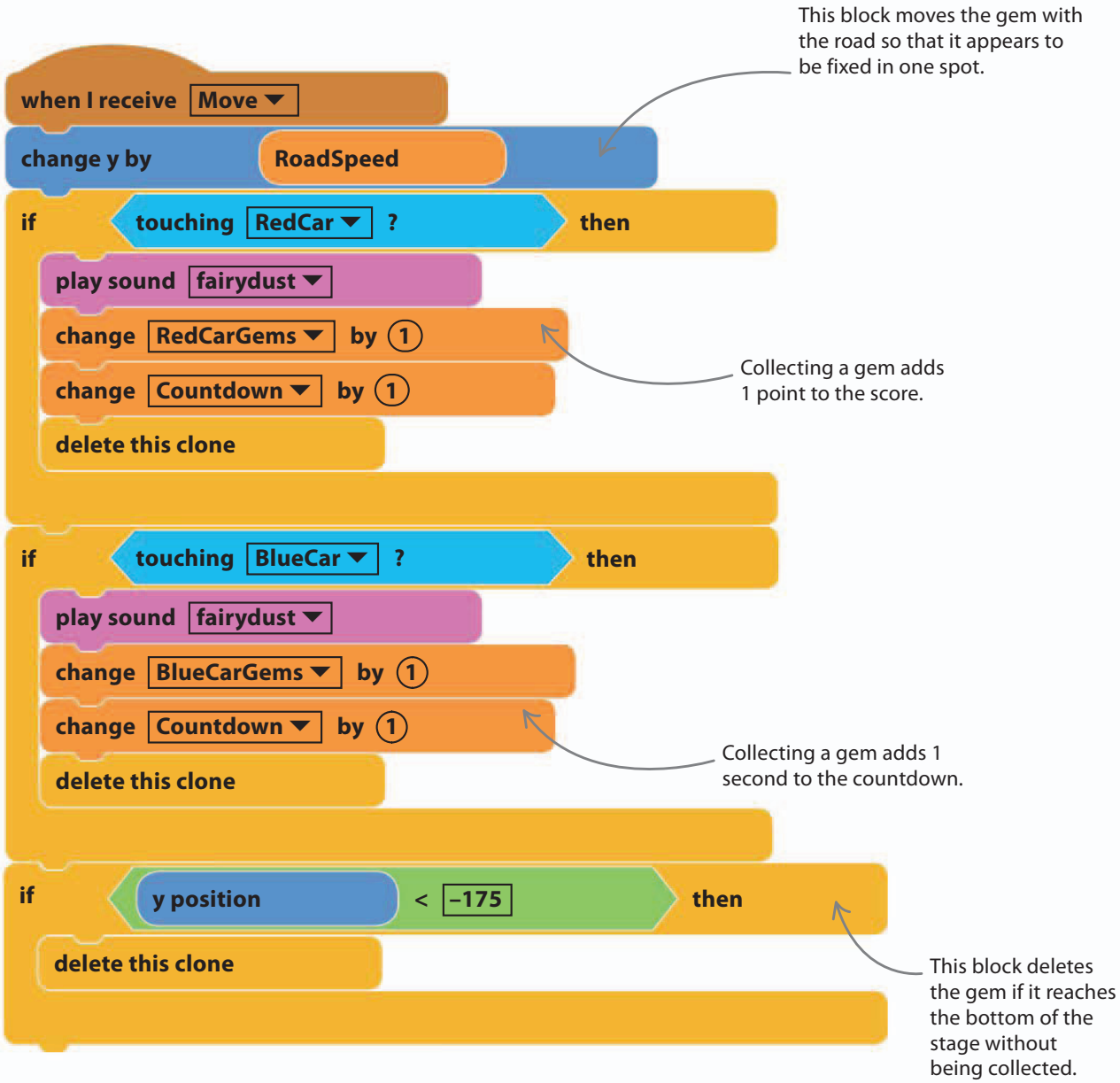
Name the sprite "Gem".

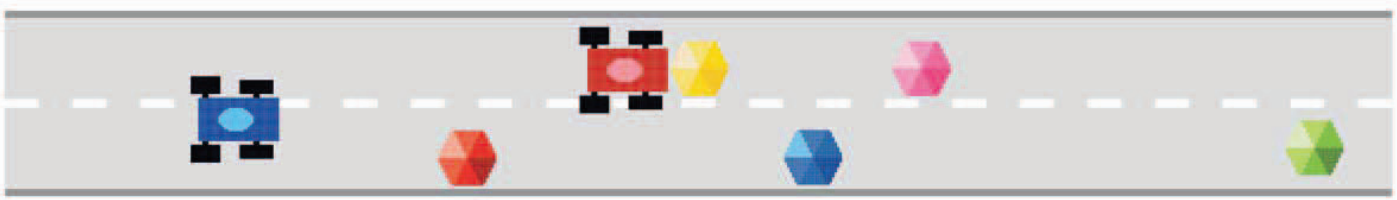
29 Create two variables—"RedCarGems" and "BlueCarGems" (both for all sprites)—to tally how many gems each car collects. Now add these scripts to the Gem sprite; they're similar to the scripts for the snowballs.



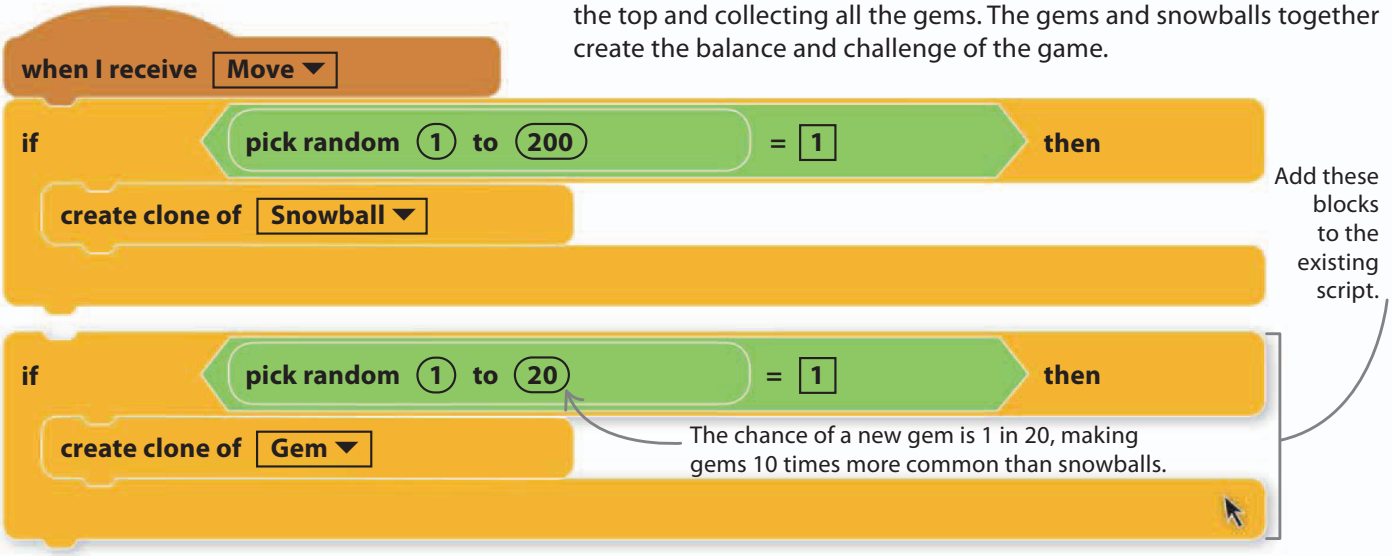


30 Add the following script to move the gems along with the road and to update the total number of gems collected by each car. Load the "fairydust" sound to the Gem sprite so that it plays each time a gem is collected.

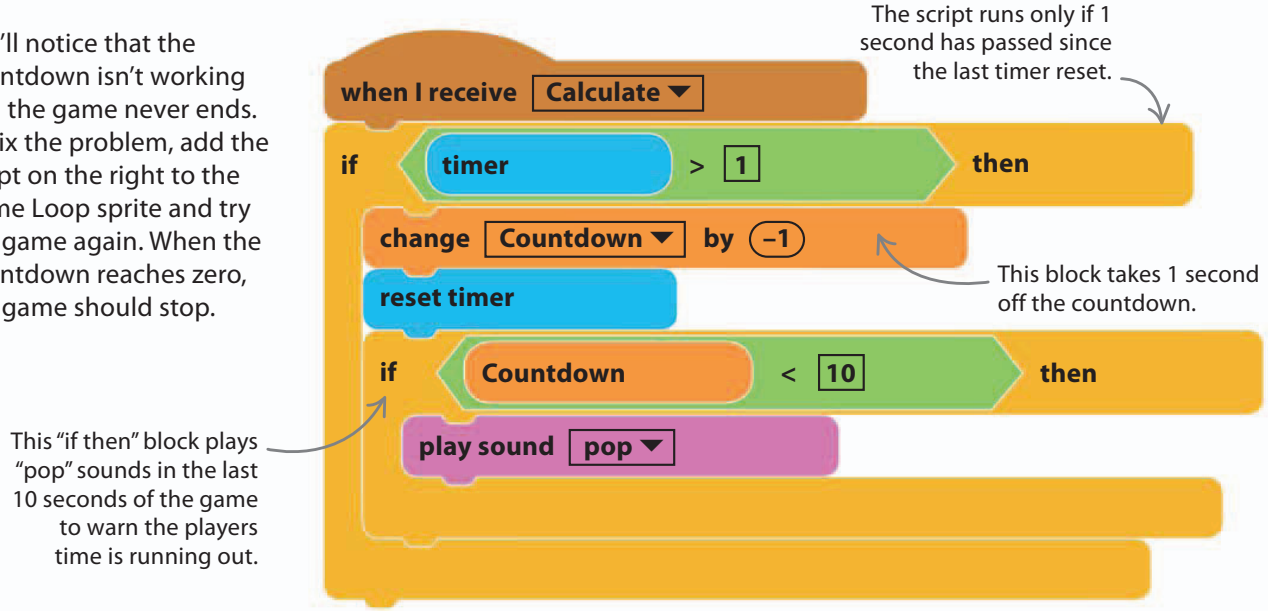




31 In the Game Loop sprite, add a second "if then" block to the "when I receive Move" script to create the gem clones. Run the game and try collecting gems. The snowballs will prevent players from rushing to the top and collecting all the gems. The gems and snowballs together create the balance and challenge of the game.



32 You'll notice that the countdown isn't working and the game never ends. To fix the problem, add the script on the right to the Game Loop sprite and try the game again. When the countdown reaches zero, the game should stop.

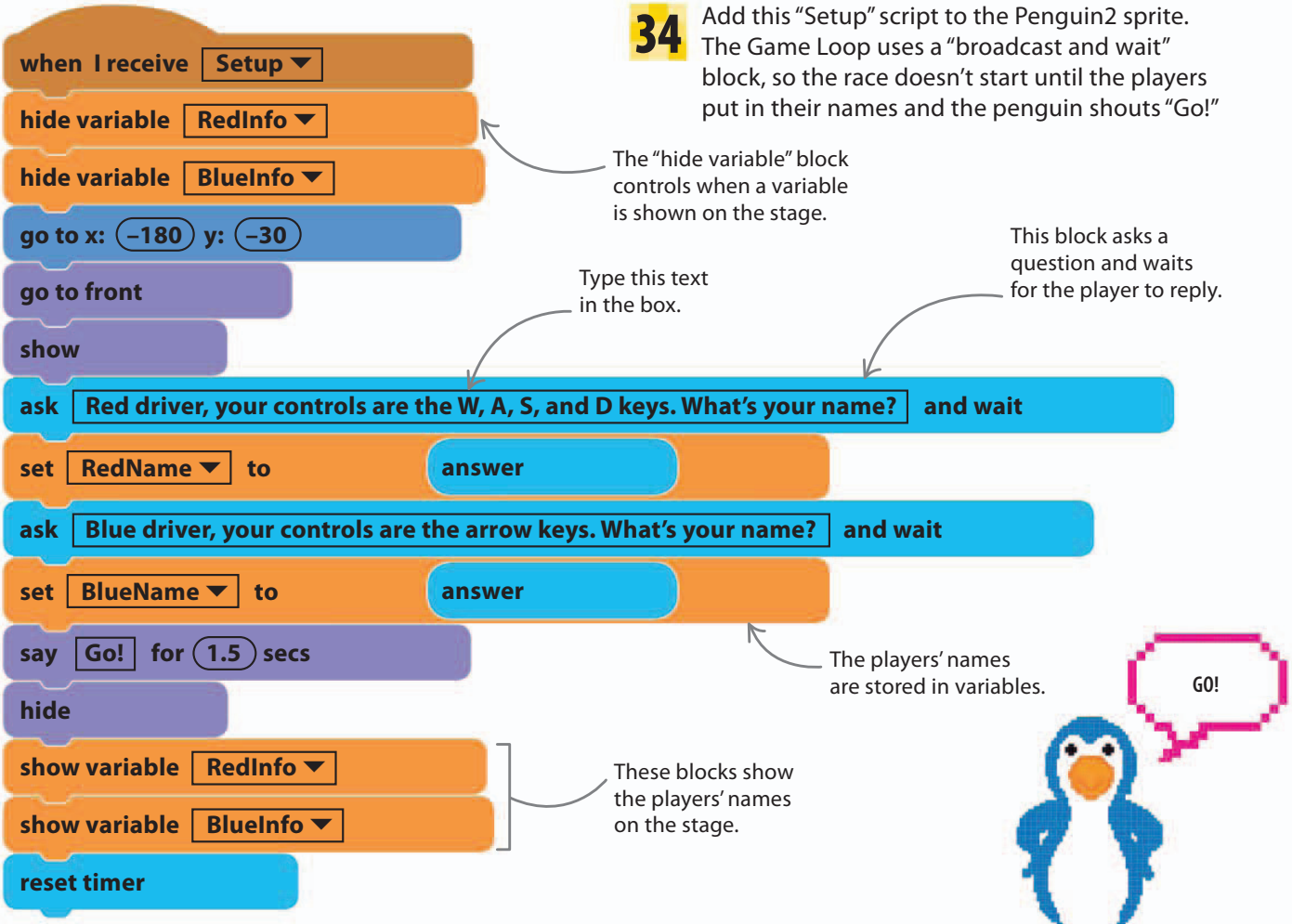


Penguin in charge

A proper start and finish can make a game look more professional. Add a penguin race official to ask the players' names, start the race, and announce the winners.

33 First, create four variables for all sprites: "RedName" and "BlueName" to store each driver's name; and "RedInfo" and "BlueInfo" to show each driver's score during the race. Then add the Penguin2 sprite to talk to the players, and load the "gong" sound from the library to Penguin2.

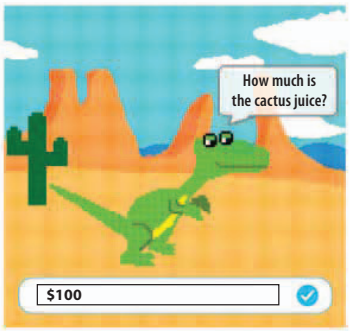
34 Add this "Setup" script to the Penguin2 sprite. The Game Loop uses a "broadcast and wait" block, so the race doesn't start until the players put in their names and the penguin shouts "Go!"

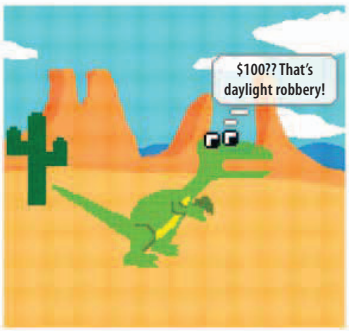


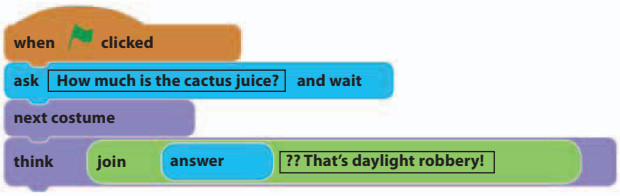
EXPERT TIPS

The ask and answer blocks

A sprite can put a question to the person at the computer by using the "ask" block. Anything typed as the reply is stored in the "answer" block, which can then be used inside other blocks just like a variable can.

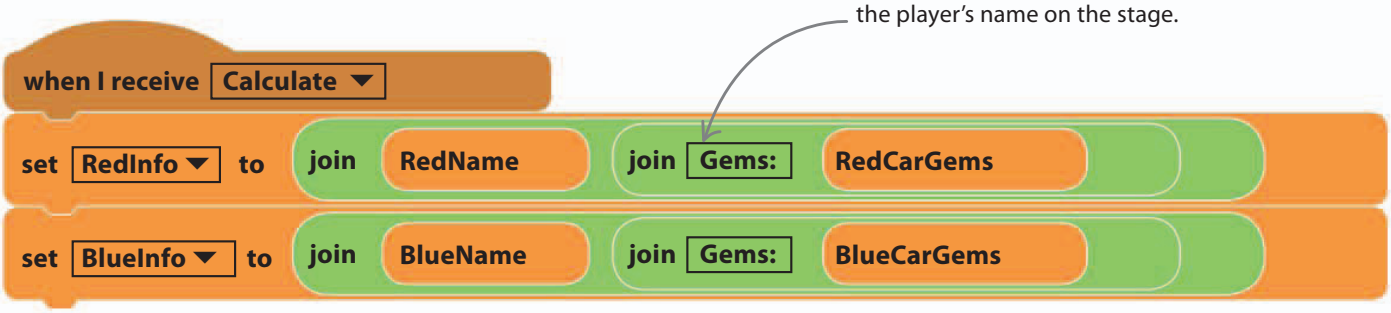






35 Add this script to the Penguin sprite to set the "RedInfo" and "BlueInfo" variables, which are displayed on the screen to show the scores.

Type a space before "Gems:" so that it doesn't form a single word with the player's name on the stage.



36 Run the game. Hide all variables except "Countdown", "RedInfo", and "BlueInfo" by unchecking their boxes in the Data section. Then right-click the RedInfo and BlueInfo signs on the stage and choose "large readout". To make everything look tidy, drag the signs to the top left and move the countdown to the top right.

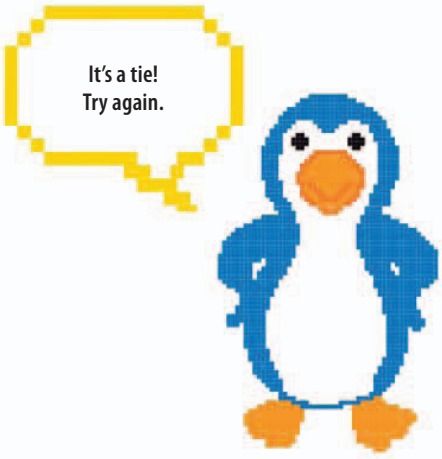
Check boxes to show the variable on the stage.

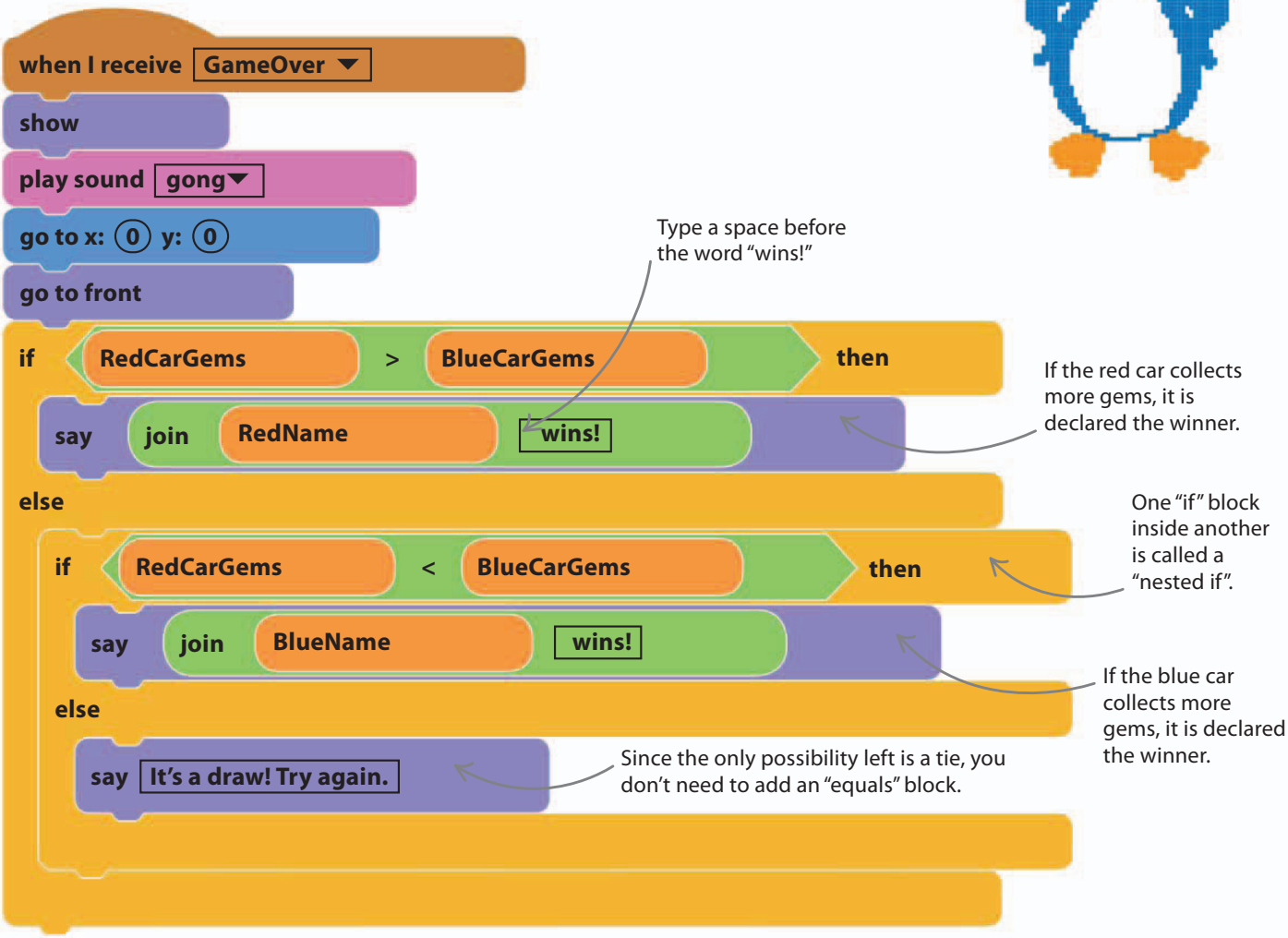
LINGO

String

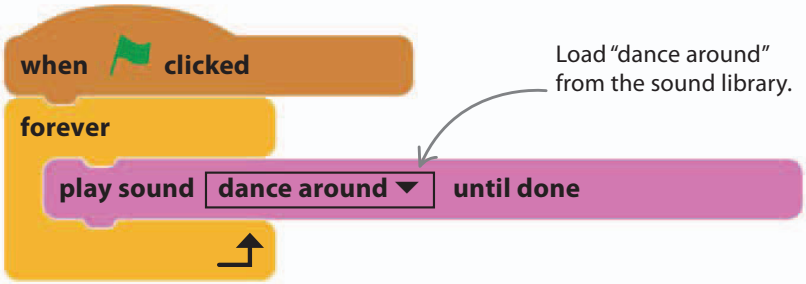
Programmers call an item of data that contains words and letters a "string". It can include any character on the keyboard and can be of any length.

37 To make the penguin announce the winner, add the next script. This script has one "if then else" block inside another. Think about the three possible results—red wins, blue wins, and a tie—and it should all make perfect sense.





38 Finally, add some rhythmic dance music to make the game feel faster. Load "dance around" to the Game Loop sprite and then add this script. It's a loop, and extra loops can slow everything down, but since it only runs once every few seconds it won't affect the game play.



Hacks and tweaks

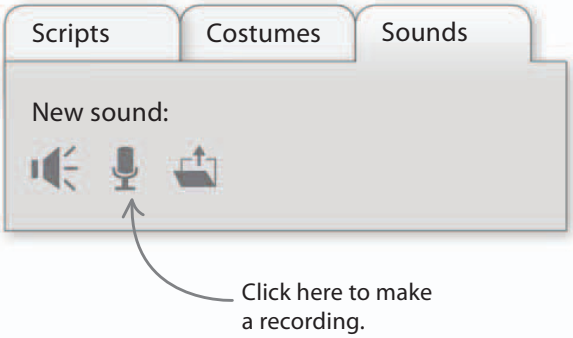
Now over to you! Personalize this race with your own features and adjustments. Make it as fast, slow, hard, fast, serious, or silly as you like.

△ **Instructions**

Remember to add instructions to the project page in Scratch. Make it clear that it's a competition to get the most gems and not a race to the finish line. Give players a helpful hint by telling them they can push the other player off the road.

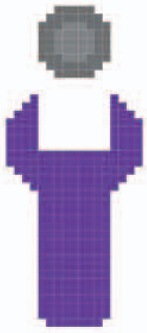
▽ **Record your own sounds**

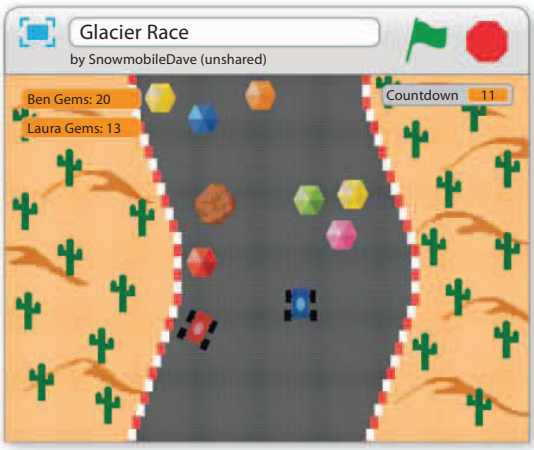
You can use your own voice to make announcements in the game. To record your voice, you need a computer with a microphone. Select the Penguin sprite and click on the Sounds tab. Then click the microphone icon to make a recording. Replace the Penguin's "say" block with a "play sound" block and choose your recording.



▷ **Fine-tuning**

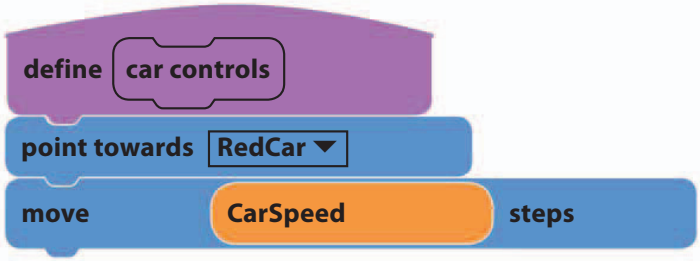
To change how hard or easy the game is, adjust the "CarSpeed", "RoadSpeed", and "Countdown" variables that are set at the start. You can also adjust how long the cars spin after a crash, how big the bounce is when they collide, and how often snowballs and gems appear. Try to get just the right balance to make the game challenging but not too hard.





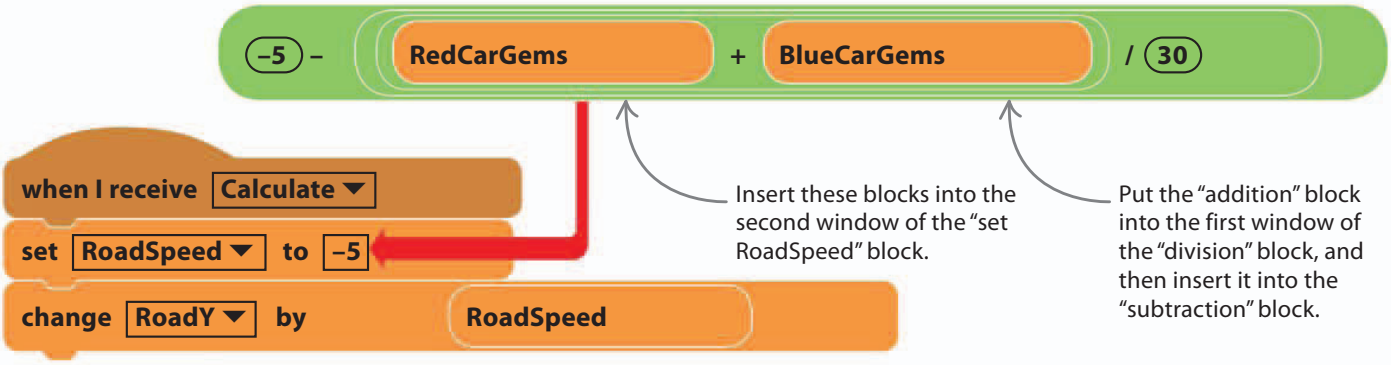
△ **Change the scenery**

It's easy to change the setting of Glacier Race by repainting the scenery. You can make the players race through a desert canyon or a dirt track in a forest. Remember to change the snowballs to match your theme.



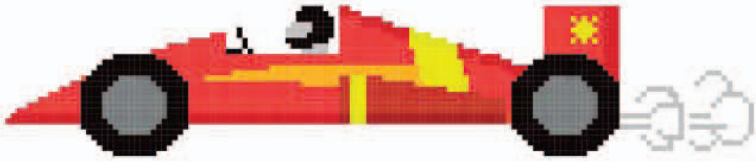
△ **One-player game**

Experiment with a one-player version of the game where you play against a computer-controlled blue car. First save a copy of the project so you don't spoil the two-player version. Change the car controls for the blue car, as shown here, and then try the game. The blue car will chase the red car and crash into it.



△ **Need for speed**

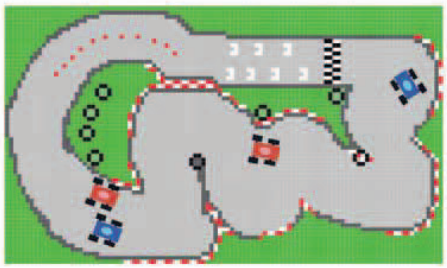
For extra thrills, you can make the game speed up as players collect more gems. To do this, change the "set RoadSpeed" block in the Game Loop sprite so that the variable changes with each gem collected.



GAME DESIGN

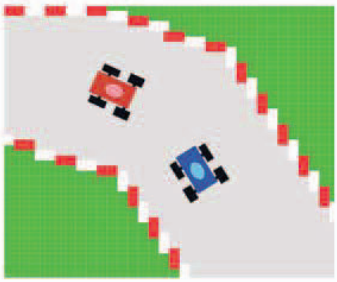
Camera angles

Game designers often talk about the "camera" in a computer game. This refers to how the picture on the screen follows the action in the game. There is no real camera, but if you imagine a camera capturing the action, you can think about different ways of showing what's going on. Here are some common camera views in computer games.



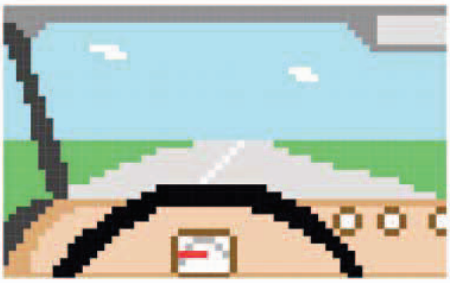
◁ **Fixed**

The camera watches all the action from one spot, without moving. Most of the games in this book use this simple camera, either with a side or bird's-eye view of the action.



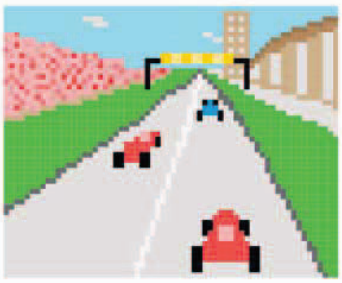
△ **Tracking**

This camera follows the player around the game. In Glacier Race, the camera follows the cars, keeping them in view as the road moves by.



△ **First person**

This camera shows the view the player would see through their own eyes. First-person games make the player feel immersed in the action, rather than watching from afar.



△ **Third person**

This type of camera is positioned just behind the player's sprite. The player feels involved in the action, but can clearly see what the sprite is doing.