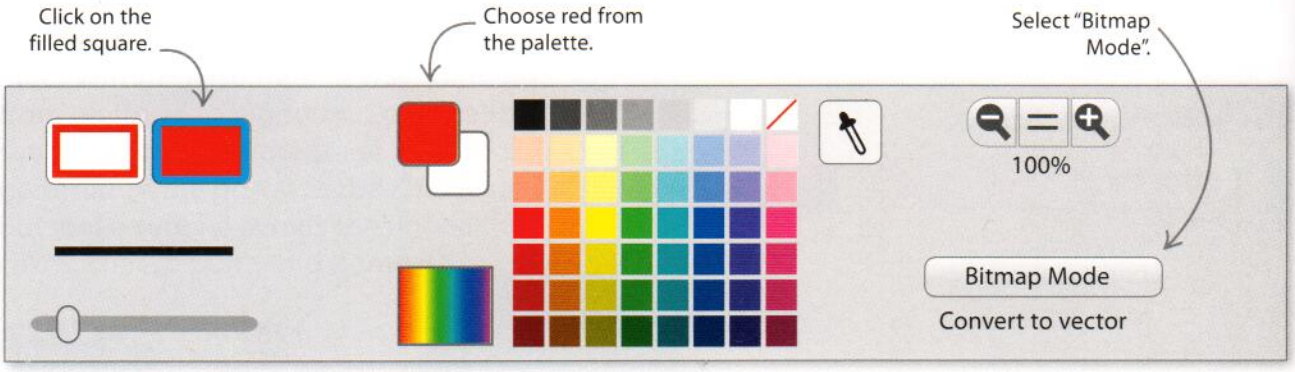
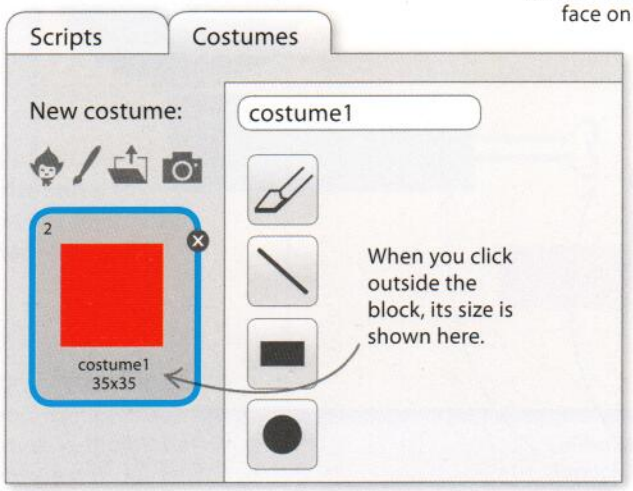
Player on a platform

This is a complicated game, so you'll need to check your work carefully at every stage. But don't worry, the project builds gradually, one step at a time. Start by getting a very simple player sprite to work properly with a platform. At first, the player is just a red square. This makes it easy to sense collisions with the platforms. You can add the blue dog on top of it later.

1 Create a new project and name it "Dog's Dinner". To make your simple player, click the paintbrush symbol at the top of the sprites list. Make sure you're in "Bitmap Mode". Choose red in the color palette in the paint editor, select the rectangle tool, and click on the filled square option.

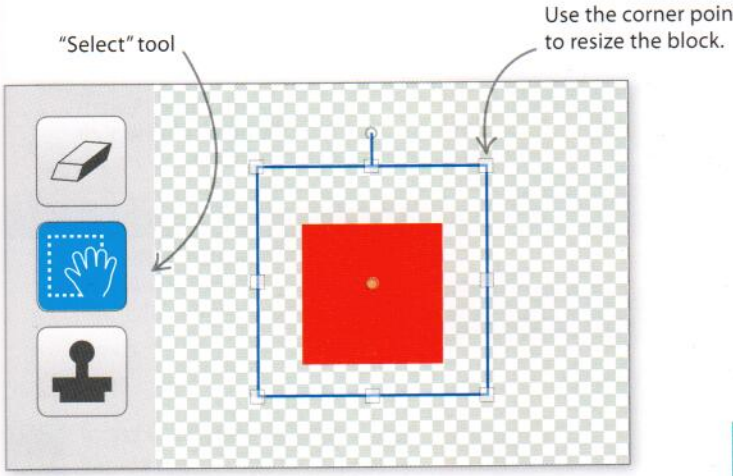


2 Hold down the shift key and drag the mouse-pointer over the paint editor to draw a small red square. If you click outside your block and look at the list of costumes, you'll see the size of the square; aim for 35x35.

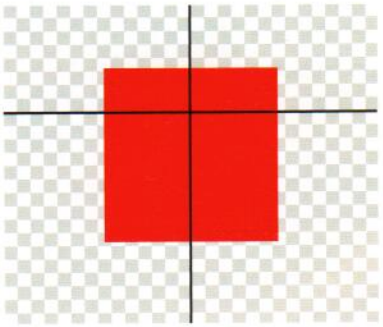


The block should be smaller than the cat's face on the stage.

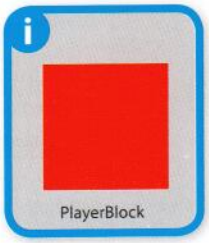
3 You can resize your block if it's too big or too small. Using the "Select" tool, click and drag to draw a square around the block. Use the corner points to resize it. Do this until the size is right.



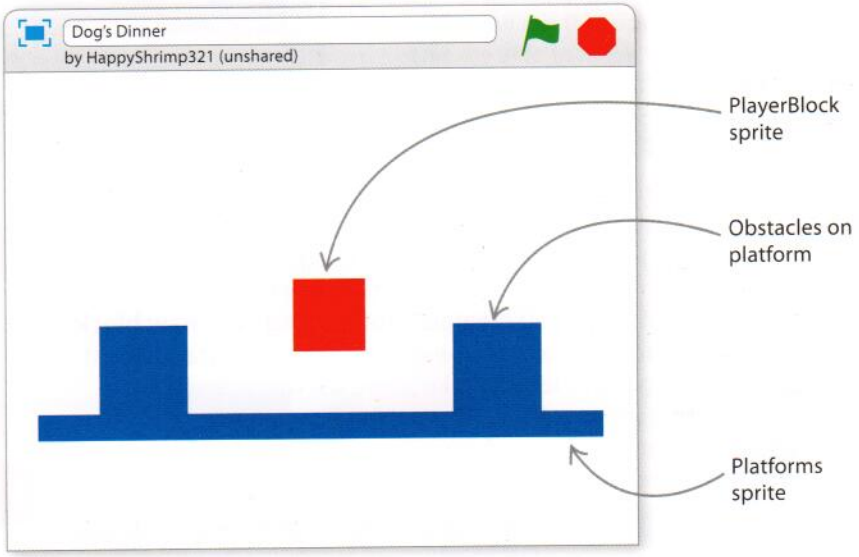
4 Select the "Set costume center" tool in the top-right corner of the paint editor. Set the center of your sprite near the top of the block (this will be useful later).



5 Rename the sprite "PlayerBlock". That's your player sprite done. Now you can delete the cat sprite.



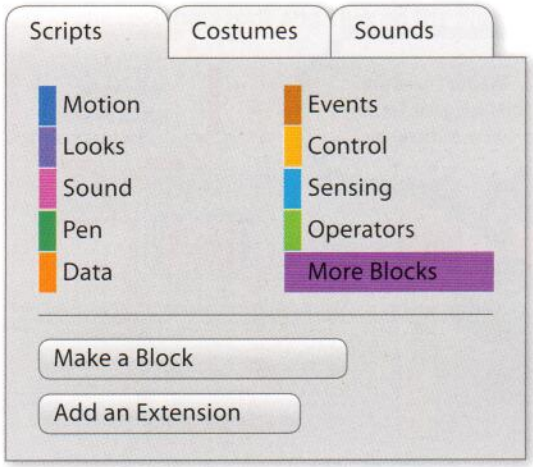
6 Now add a simple platform. Click the paintbrush symbol in the sprites list again to create a new sprite. Use the rectangle tool to draw a floor with two obstacle blocks on top. Call this sprite "Platforms". On the stage, drag your PlayerBlock and place it between the obstacles, but make sure it's not touching the platform.



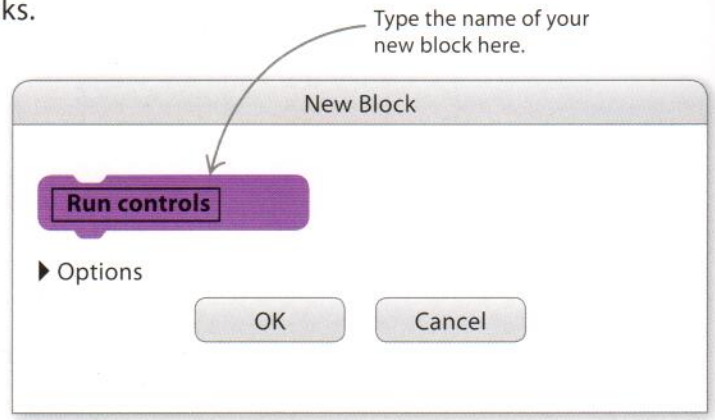
Running around

The next step is to make the PlayerBlock run when the player presses the arrow keys. You'll need a script that will stop it running through obstacles by making it reverse when it touches them. To make the code easier to read, you'll be making your own customized Scratch blocks.

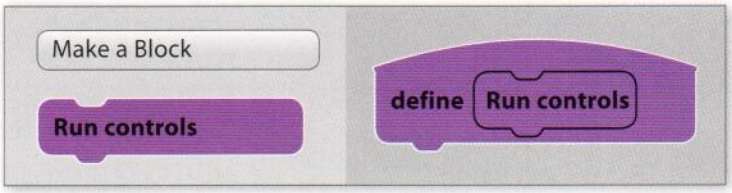
7 With the PlayerBlock sprite selected, go to the blocks palette under the Scripts tab and click on More Blocks.



8 There are no blocks in this section yet, only some buttons. Click on the "Make a block" button, and a box called "New Block" will pop up. Type "Run controls" in the window to name your new block and then click "OK".



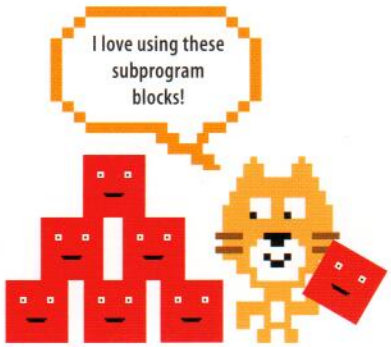
9 The new block appears in the More Blocks section, and a special purple "define" header block appears in the scripts area.



LINGO

Subprograms

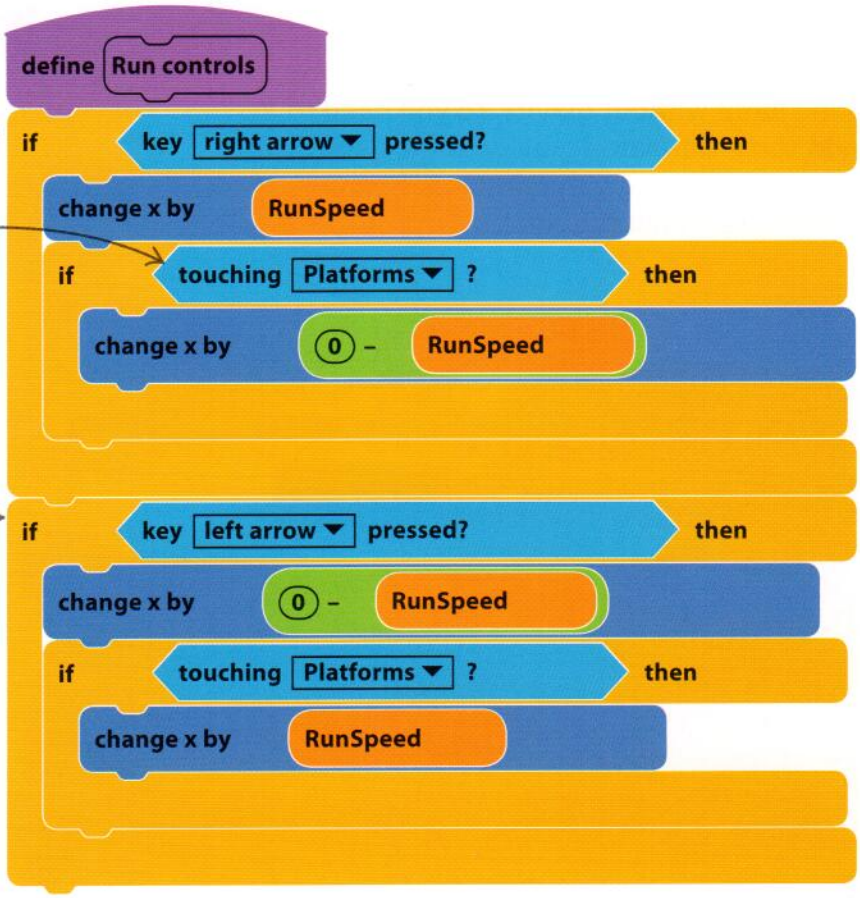
Scratch lets you group together blocks under a "define" header block and run them by using a new block that you name. This saves you building the same group of blocks again if you want to use it in more than one place. (However, the new block will only work with the sprite that you created it for.) Giving your new block a meaningful name will make your code easy to understand. Most programming languages let you take some useful code, give it a name, and wrap it up as a unit. Different languages call these units different things: subprograms, subroutines, procedures, and functions are some common names.



When the right arrow key is pressed, the PlayerBlock moves right.

If the PlayerBlock touches the platform, the move is reversed.

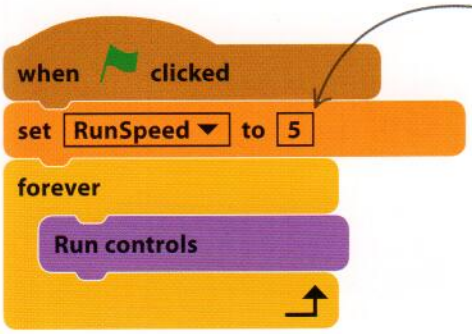
The left control works in a similar way to the right control.



10 In the Data section, create a variable for all sprites called "RunSpeed" to control how fast the player moves left and right. Uncheck the box so it doesn't appear on the stage. Then build the script shown here under the "define" header. Anywhere we use the "Run controls" block, Scratch will now run this script.

11 Next, add the script below to use your new custom block in a "forever" loop.

Type "5" here.



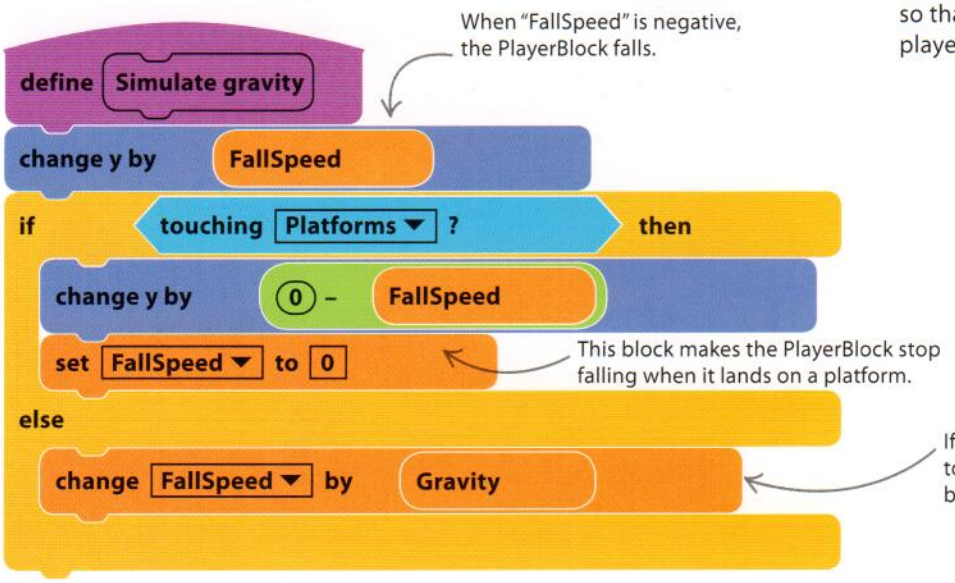
12 Now run the project. You should be able to move the red block left and right with the arrow keys, but not be able to move through the obstacles.

The red block is trapped between the blue squares.

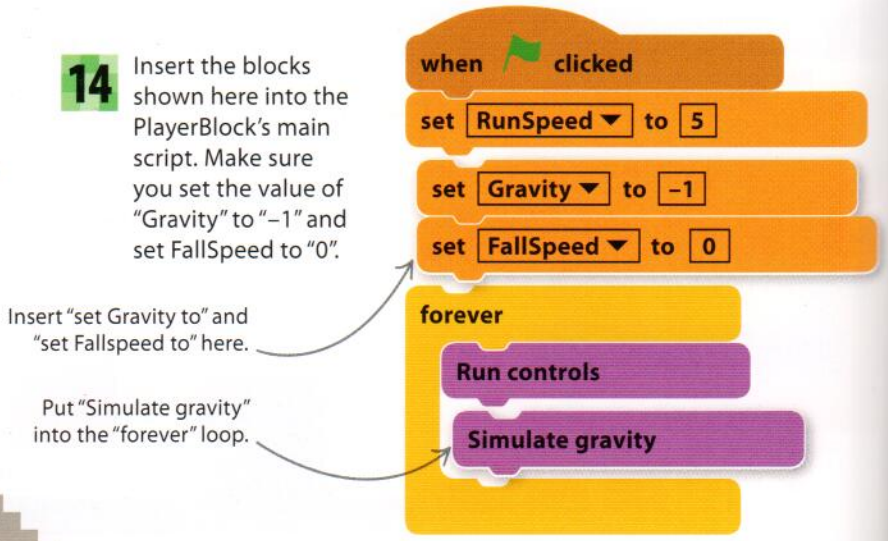
Up and down

Platform games are all about jumping. You can't jump without gravity, so you need to add some simulated gravity to the game. You may recognize how the simulated gravity works if you built the Jumpy Monkey game.

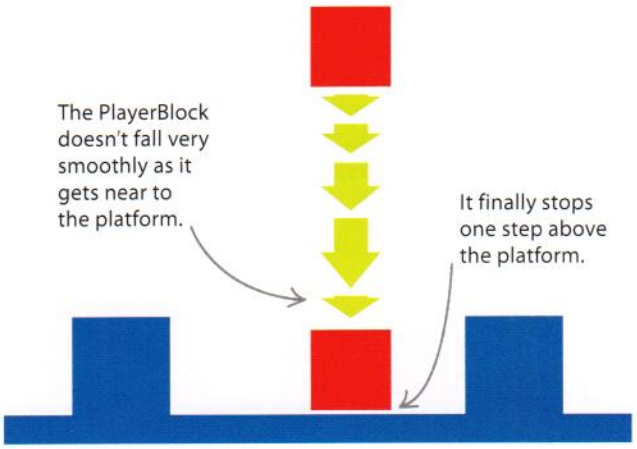
13 Add two more variables for all sprites: "Gravity" and "FallSpeed". Uncheck both boxes. Then click on More Blocks and make a new block called "Simulate gravity", following the script shown here. It moves the PlayerBlock down by the amount "FallSpeed" and then checks to see if the PlayerBlock has hit the platforms. If so, it reverses the last move and sets "FallSpeed" to zero so that the platform stops the player's fall.



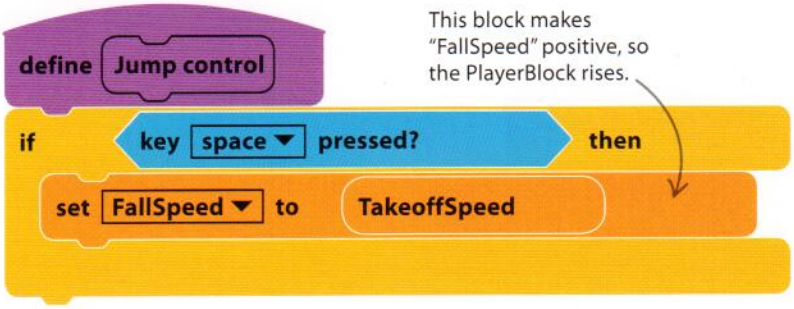
14 Insert the blocks shown here into the PlayerBlock's main script. Make sure you set the value of "Gravity" to "–1" and set FallSpeed to "0".

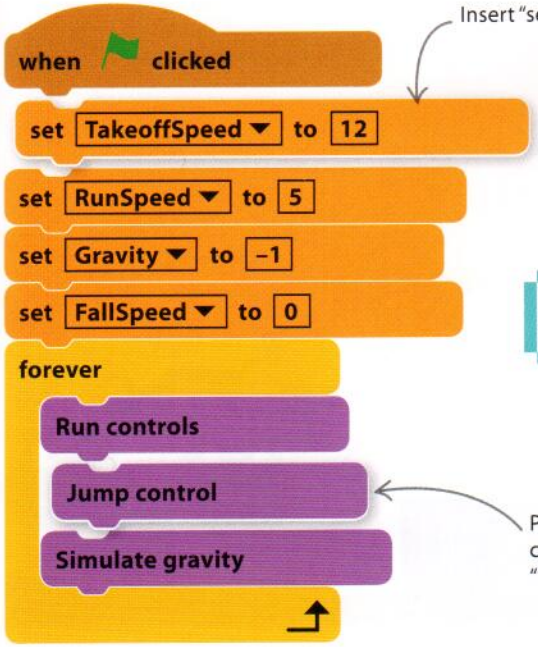


15 Run the project. Grab the red square with the mouse and drop it from above the platform. It will fall down and come to rest on the platform. But there's a problem: it slows down just above the platform. That's because our method makes the block reverse after hitting the platform and then start falling again at a slower speed. We can fix that later.



16 Now to create the jump. It's really easy: just add some new code to give the PlayerBlock an upward kick when you press the space bar. First, make a new variable for all sprites called "TakeoffSpeed". This is the player's upward speed on a jump. Then create a new block called "Jump control" and define it as shown here.

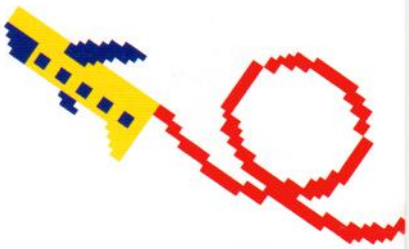


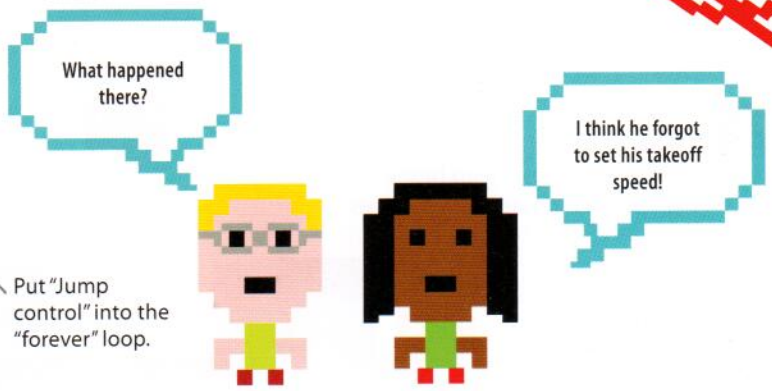


Insert "set TakeoffSpeed to" here.

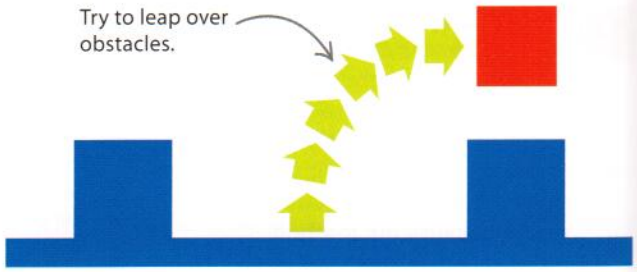
Put "Jump control" into the "forever" loop.

17 Add the "set TakeoffSpeed to" block into the PlayerBlock's main script and set it to "12". Insert the "Jump control" block into the "forever" loop.





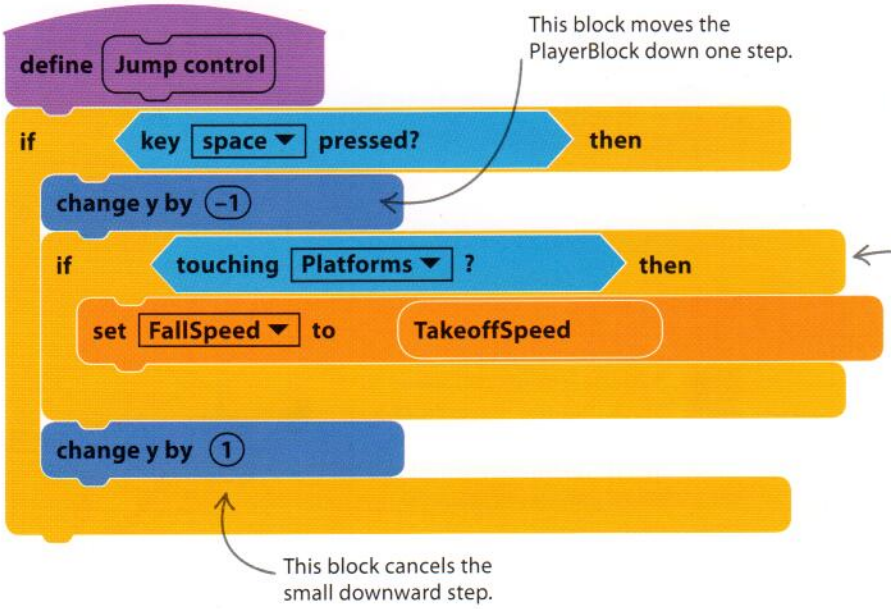
18 Now run the project. Press the space key briefly. The PlayerBlock jumps up and comes back down again. You should be able to combine the run and jump controls to jump onto or over the obstacles on the platform. You now have the makings of a platform game! However, there's another bug: if you keep the space key pressed, the PlayerBlock goes up forever.



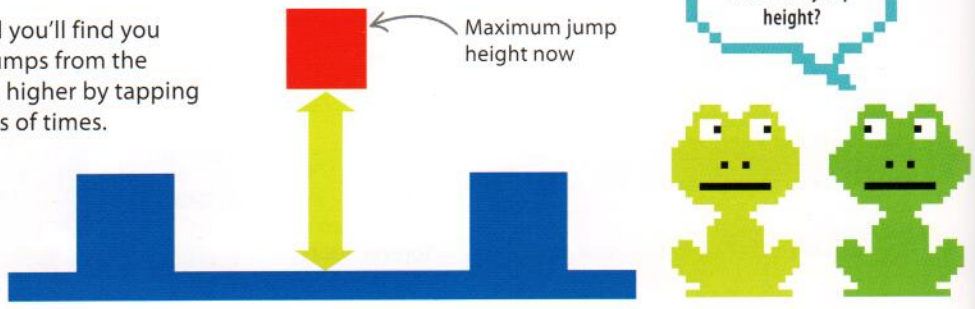
Fixing the jumping bugs

There are two bugs that spoil our jumps: one causes the PlayerBlock to jump infinitely high; the other keeps it from falling smoothly. You can fix them by tweaking the jump and gravity controls.

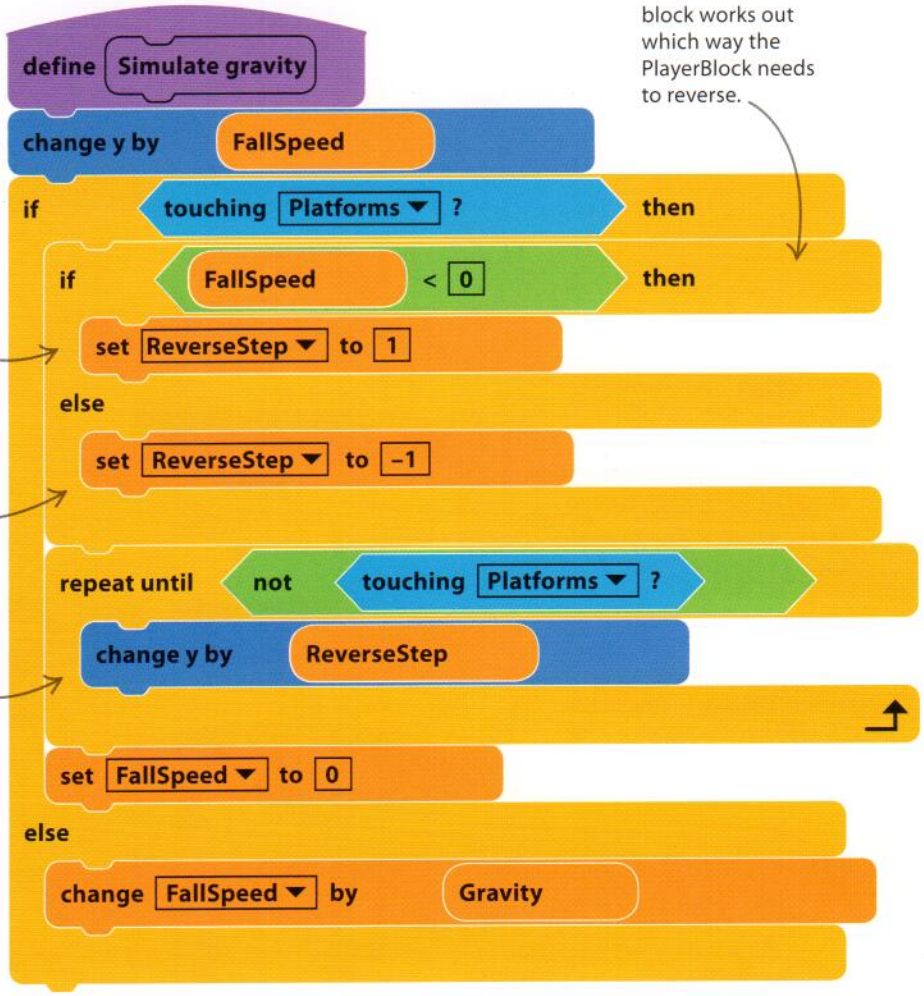
19 To fix the infinite jump bug, add a test to the "Jump control" script to check whether the player is on or just above the platform. (Remember that the "Simulate gravity" script leaves the PlayerBlock one step above the platform, so the two sprites aren't touching.) This fix will disable the space key when the player is in mid-jump.



20 Try the code above and you'll find you can only make single jumps from the platform and can't leap higher by tapping the space key space lots of times.

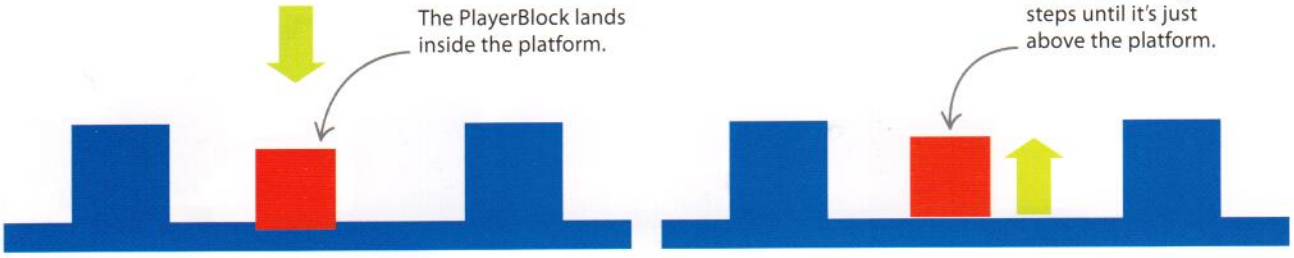


21 To fix the other jumping bug (pausing just above the platform and then falling slowly again), you need to change what happens when the PlayerBlock touches the platform. At the moment, the red square reverses by the whole "FallSpeed" number when it hits a platform. Instead, we'll make it reverse in tiny steps until it's just above the platform. Create a new variable called "ReverseStep" for all sprites. Change the "define Simulate gravity" script as shown here.

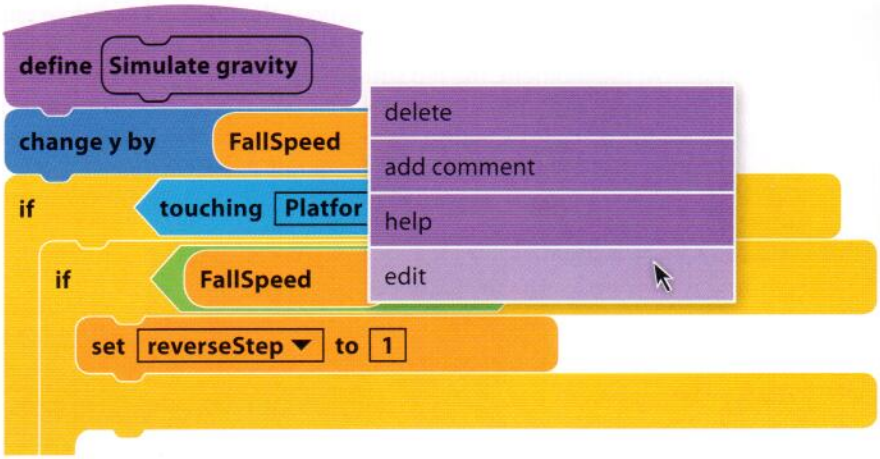


▽ How it works

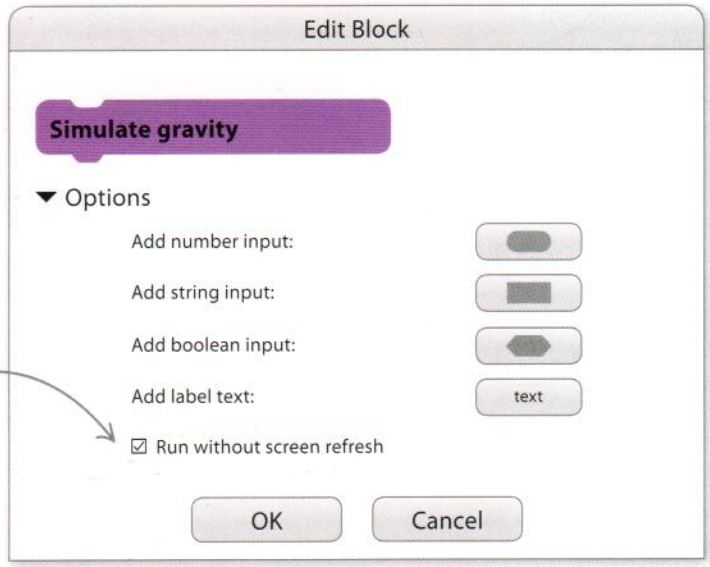
When the PlayerBlock hits the platform, the "repeat until" loop reverses the PlayerBlock until it sits just one step above the platform.



22 Try the jump again to see for yourself. You'll notice that the PlayerBlock rises back out of the platform very slowly. But we don't want that part to happen in slow motion! Scratch has a trick to fix this. Right-click on the "define Simulate gravity" header block and select "edit" from the pop-up menu that appears.

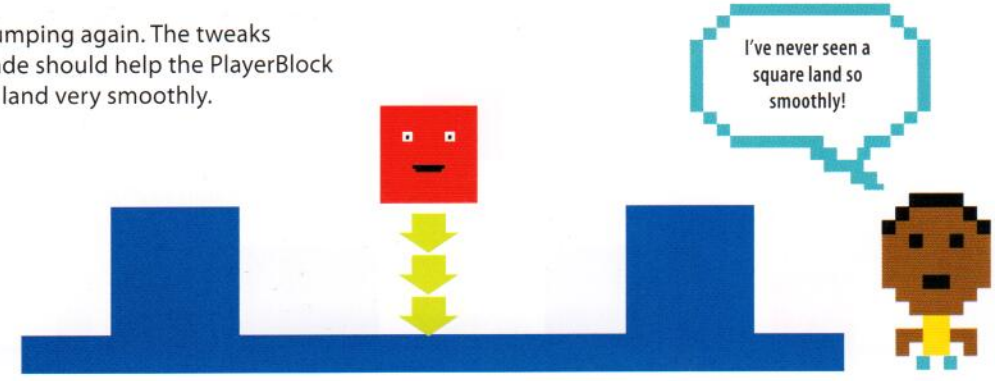


23 An "Edit Block" box appears. Click on "Options" and check "Run without screen refresh". This will make the gravity script run continually (without showing each reverse step), which will get rid of the slow-motion effect.



Check this box and the whole script will run much faster.

24 Now try jumping again. The tweaks you've made should help the PlayerBlock jump and land very smoothly.



GAME DESIGN

Which jump?

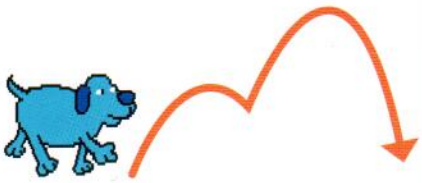
Games use many different types of jump. Which type you choose is key to your game's design. Here are three common jumps.

▽ **Single jump**

This is the jump you have in Dog's Dinner—you can only jump if you're on the ground. You go up and then down, but in some games you can steer left and right during the jump.

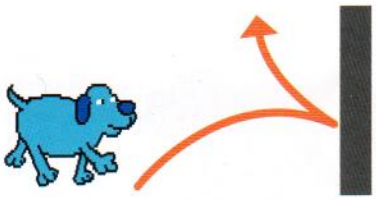
▽ **Double jump**

This is the jump you had before you fixed the infinite jumping bug—you can jump again in the air to go higher. In some games there are limits on double jumping—for example, you can only do it if you're going up.



▽ **Wall jump**

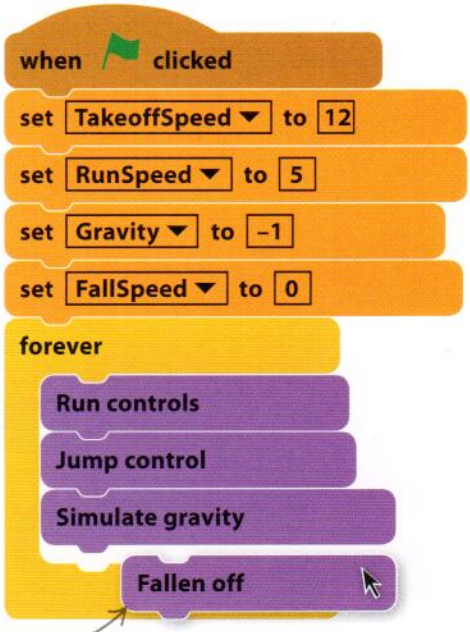
When you touch a wall, you can jump up again. Ninja-type characters often have this power. It's not very realistic but it's lots of fun!



Falling off the level

Platform games are all about staying on the platforms. Add the next script to the PlayerBlock to make the game end if it falls to the bottom of the stage.

25 Make a new block called "Fallen off", shown below, to check whether the PlayerBlock is at the bottom of the stage. Add it to the "forever" loop. Then build the short script at the bottom of the page to stop the sprite when it gets the "GameOver" message. Test the new code: the controls should stop working when you hit the deck.



Drop this block inside the "forever" block.