Finding and fixing problems with your scripts

What is a bug?

Sometimes things don't quite work as expected. Problems with scripts are called 'bugs'. When you find bugs in your project, it can be difficult to understand what part of your script is causing them. The process of fixing bugs is called, 'Debugging'.

Useful ways to debug your scripts

Check the values of variables

If you are using variables to keep track of things like scores, you will be able to see these on the screen. Make sure that the values of the variables are what you expect them to be.

For example, in the Gravity Jump game, the horizontal_x variable is used to keep track of the scrolling ground and platforms (these are sprites that are moved sideways).

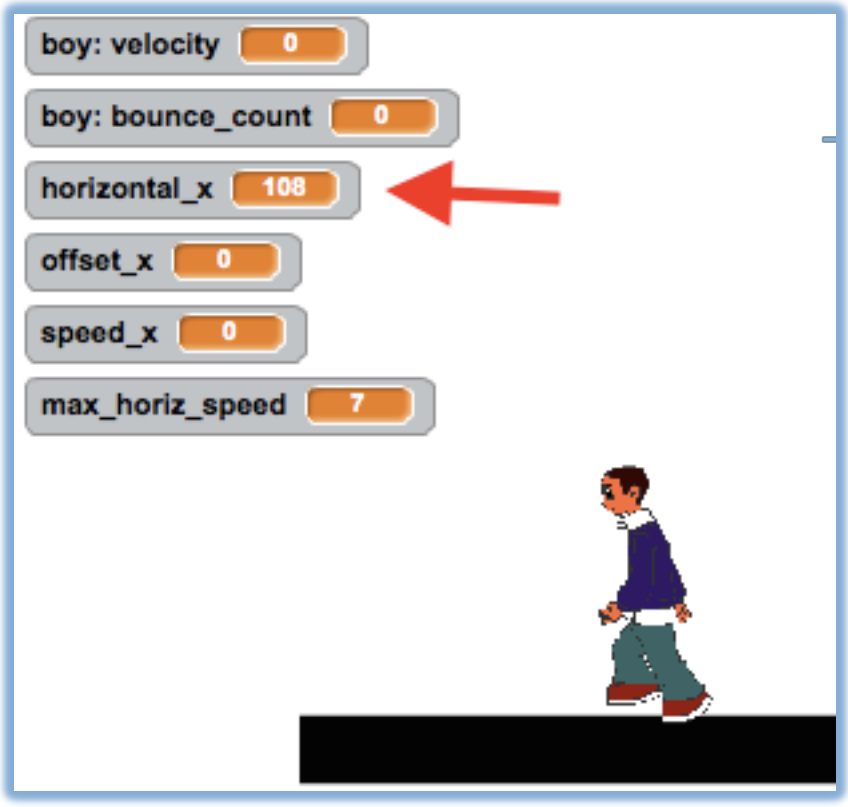

Figure 1: Horizontal position of scrolling ground.

As our character walks along the ground, the horizontal value changes. By checking the value, we can tell whether things are working properly. If a value is not what you expect, check your script to find the problem.

Add a command to print something to the screen

Sometimes it is hard to tell if something is happening in your game. For instance, you may not be sure that an IF statement is detecting something, or whether sprite collisions are being detected properly. Other examples of bugs could be when the game never gets to a part of your code, or when something keeps repeating when it shouldn't.
In these cases, it can be useful to add a command to the problem section of your script, so it prints to the screen when that point of the script is reached.

For example, observe the script below. When the space key is pressed, we may be unsure of whether the sprite is being detected touching a black part of the screen. We can see the *bounce_count* variable by making sure it is shown on the screen, but the touching detection is trickier.

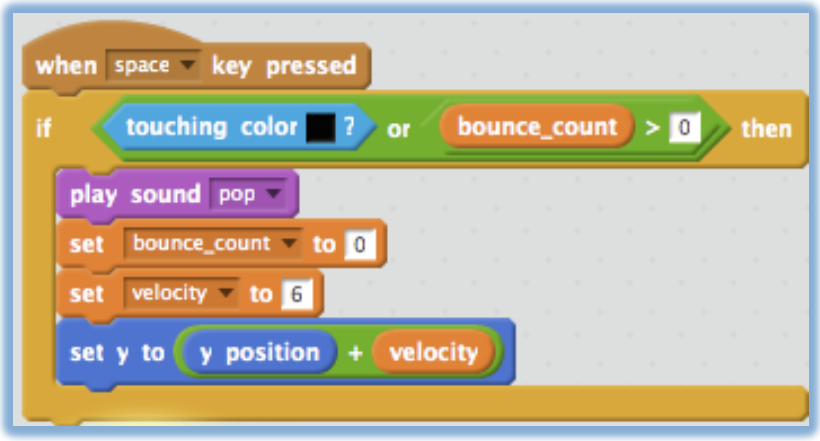

Figure 2: Script example.

A good way to check this could be to wait for the *pop* sound. However, we could also insert a command to write a message on the screen when the *if* condition is detected (see below).

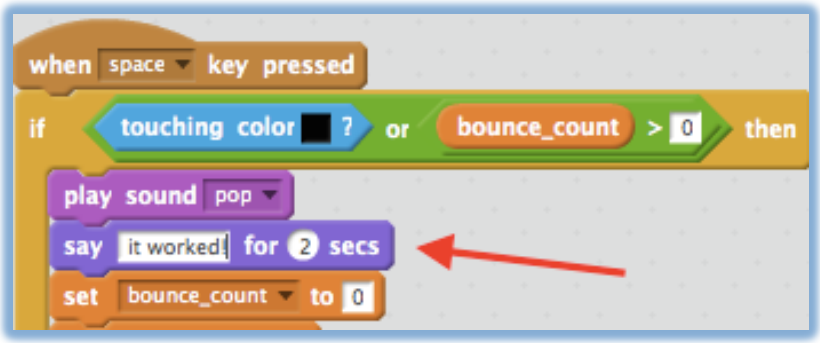

Figure 3: *Say* command added to test if script is working

Inserting such a command can help locate a problem in your script. If 'It worked' is not written to the screen, then that part of the script is not being reached when it is run. You can now look at the script and work out why that is not happening. In some cases, you may need to repeatedly check for an event, by placing a forever loop around the commands. In the above case, the *if* statement will only check for the touching color condition when the space key is pressed.

Move sprites to create the event

Sometimes it can be quicker to drag sprites into position in order to create the situation where a command will trigger, so you can test whether it works. The bug might be deep into the game, so by manually moving the sprite yourself, it will speed up the debugging.

Keep things simple

When you have long scripts, lots of scripts, or just too much going on, it can be easier to move some of the script aside so you can understand what is happening. In some instances, going back to the basic actions can make it easier to see the logic in your script. If things work at that stage, you can put commands back into place, one by one, until something causes the bug to occur.

Conclusion

This should give you some ideas to get started finding out why something doesn't work with your scripts. Once you find out where a problem lies, you may be able to think of a better way to do what you want. Sometimes it can help to make your script easier to read, by breaking scripts down into parts, or stripping them down to the essentials. Changing a script so that it still works, but is easier to read and manage, is called 'refactoring'. This is a very useful skill to learn and will help you write better scripts in future.