Terms & Conditions

Creative Computing Club
Scratch Introduction

www.pixelh8.co.uk

Set Up

For this tutorial we used Scratch some of the images and names may vary from version to version.

Download and install Scratch as per the developers instructions.

Scratch will automatically generate a “Scratch Projects” folder where your programs will be stored.

Now lets open Scratch and click on File and “Save”. Save the Scratch project .sb as “HIBOT”

Scratch is an amazing “visual” programming languages perfect for introductory sessions in computer programming. Despite it being simple to learn, it can be used to create complex and varied projects from platform games, to drum machines to interactive story books.

Now lets get started.

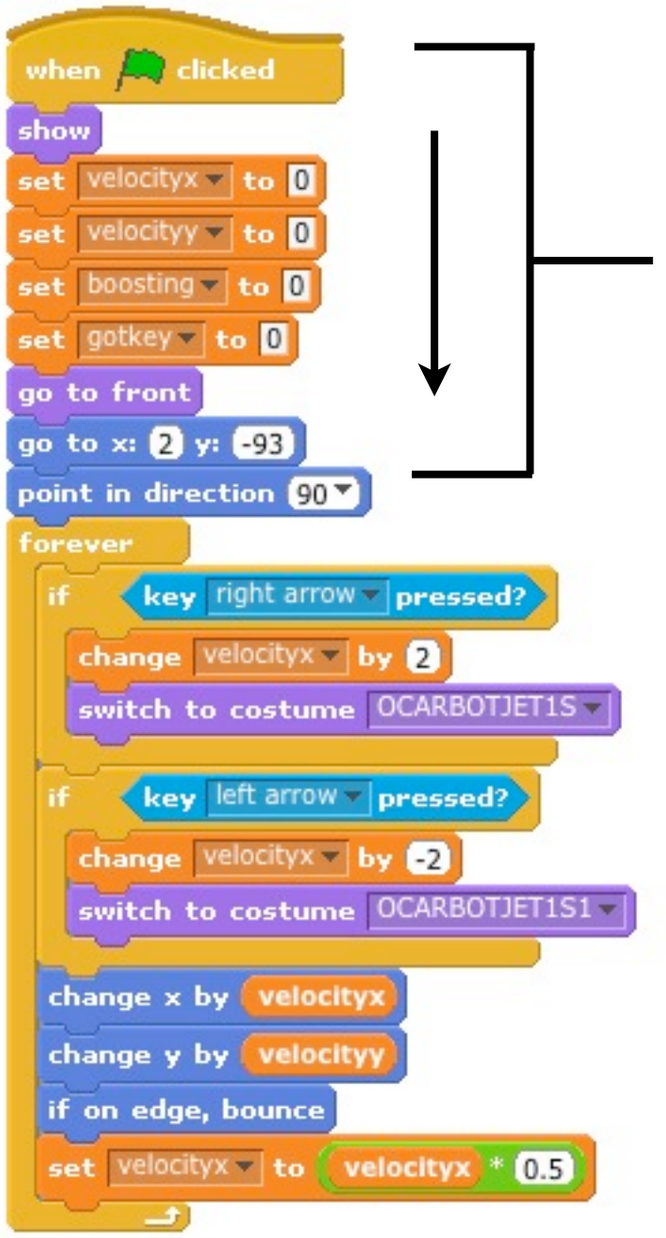
Example 1

In this example we are going to load an image of “HI-BOT” and get it to move left and right on the screen.

First load download the“HI-BOT” graphics and sound zip file and unzip it. Load “OCARBOTJET1S” as a costume for “Sprite1” object and do the same “OCARBOTJET1SL”. Delete the image of the cat. This gives you the robot facing right and left. Select “OCARBOTJET1S” making it your starting costume.

Below is all of the code, let us break it down to see what is happening, the code works from top to bottom.

This is the start of the program.



This code runs once at the beginning of the program and sets up the values of the games variables.

This code loops from top to bottom and starts at top of the “forever” loop.

Example 1

Lets look at the setup of the “HI-BOT” we need to set up five variables and they need to be “for all sprites” this way they can all share the information.
Think of variables as little boxes that store information, in these examples, they are storing whole numbers. We need to make a “velocityx”, “velocityy”, “boosting” and “got key”. The other sprites will need to know what the value of these variables are from time to time. Making the variables accessible like this make them global variables, if we didn’t let them share the information they would be called local variables.

We then send the “HI-BOT” sprite to the front that way it doesn’t get lost behind any other graphics. We give “HI-BOT” a starting position, and start it facing right.

This ensures the sprite is visible on the screen.

This sets “velocityy” to 0 and controls the up and down motion.

This sets “gotkey” to 0, thus locking the door, when “HI-BOT” collects the key it changes to 1, the door unlocks.

This sets up the starting position of “HI-BOT”.

This sets “velocityx” to 0 and controls the left and right motion.

This sets “boosting” to 0. Which controls the height of the jump.

This puts the “HI-BOT” at the front of everything else.

This “points’ the “HI-BOT” right, ensuring the it is facing the correct direction at the start.

Example 1

Inside our “forever” loop we have two questions or “if” statements lets look at them first. The first “if” checks the “right arrow” key and moves the “HI-BOT” sprite right, the second “if” does the same but to the left.

The program then goes on to update the sprite location based on values in “velocityx” and “velocityy”, checks if the sprite is trying to leave the screen and then “bounces” back in to the viewable area.

Finally it either speeds up the motion to 2 or down to 0 if nothing is being pressed.

This “if” statement asks if the right arrow is pressed

Add “2” to the current value of “velocityx” and change the appearance of “HI-BOT” to show the right facing image. This moves the sprite 2 pixels right and shows the correct image.

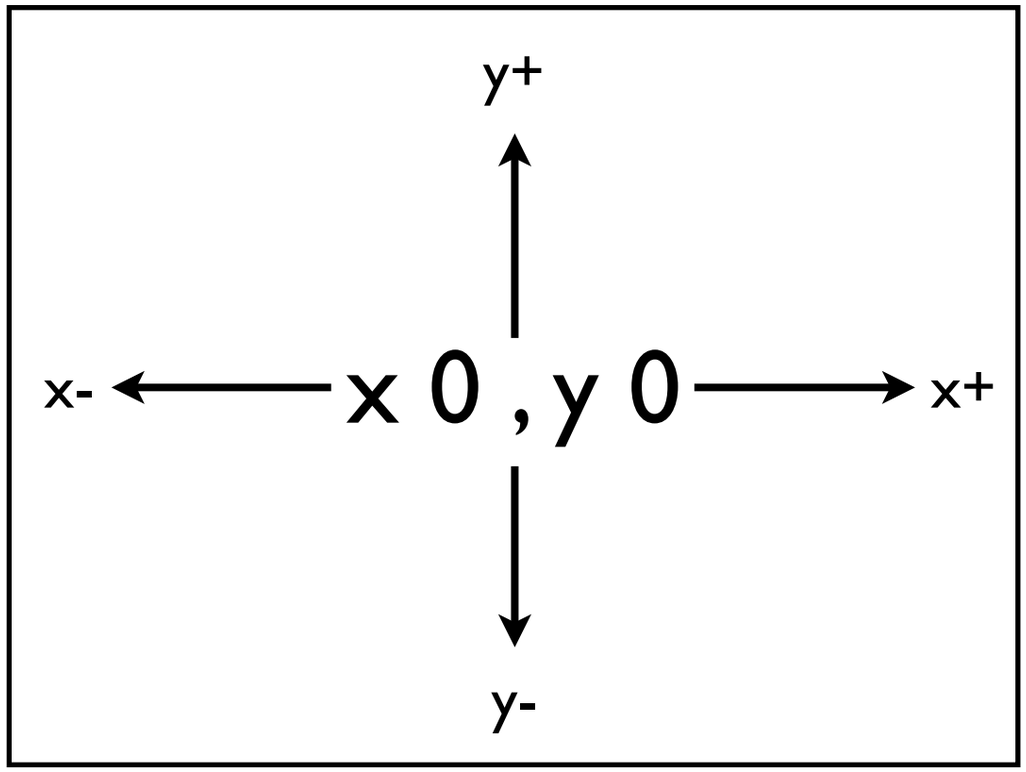
The same is done for the left arrow.

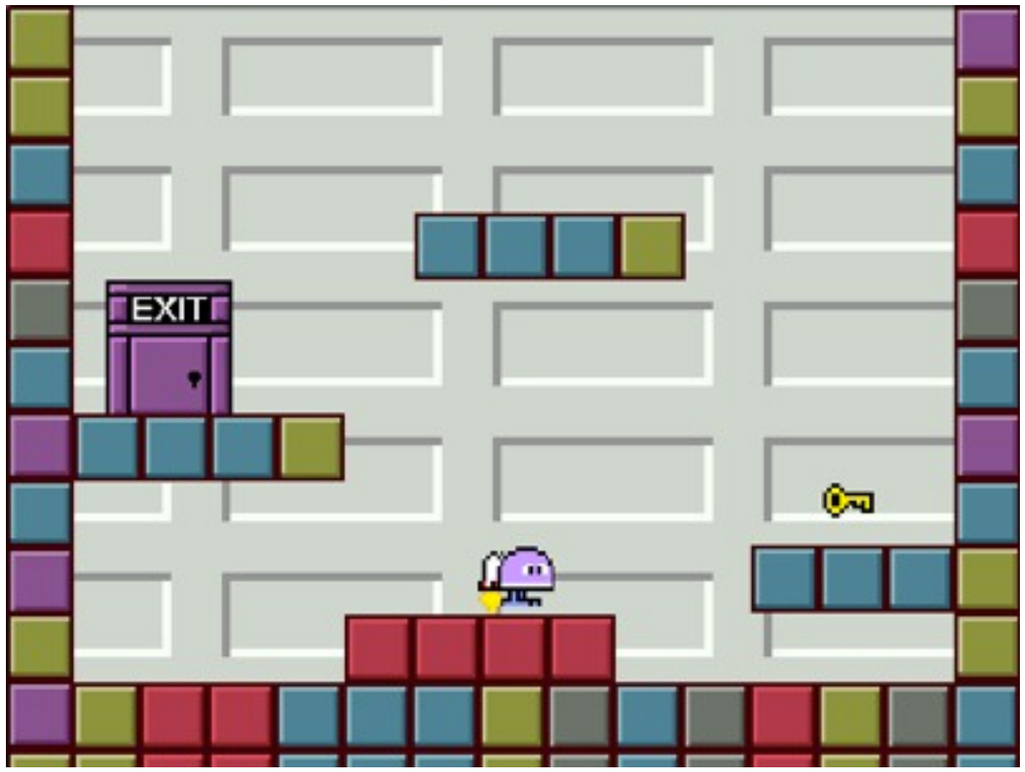
Updates x coordinates with “velocityx” by adding it to the existing value, the same is done with y and “velocityy”.

If leaving the screen, put it back on the screen.

Speeds up and slows down left and right motion.

Screen Layout





The centre of the screen is x 0 , y 0.

Example 2

Great our character can move left and right but we need to make them “boost” using the up key.

First thing load up the “SPRING” sound in to the “HI-BOT” object, then load the “BG” and “BG1” in to the “Stage” object as different costumes.
“BG” is for the first level background “BG1” is for the second level. Select “BG” making it your starting level background.

Then add these two if statements below the other two, and before the area where the screen is updated. There is also a nested “if”, a question inside a question.

Add the change “velocityy” block to make the “HI-BOT” fall.

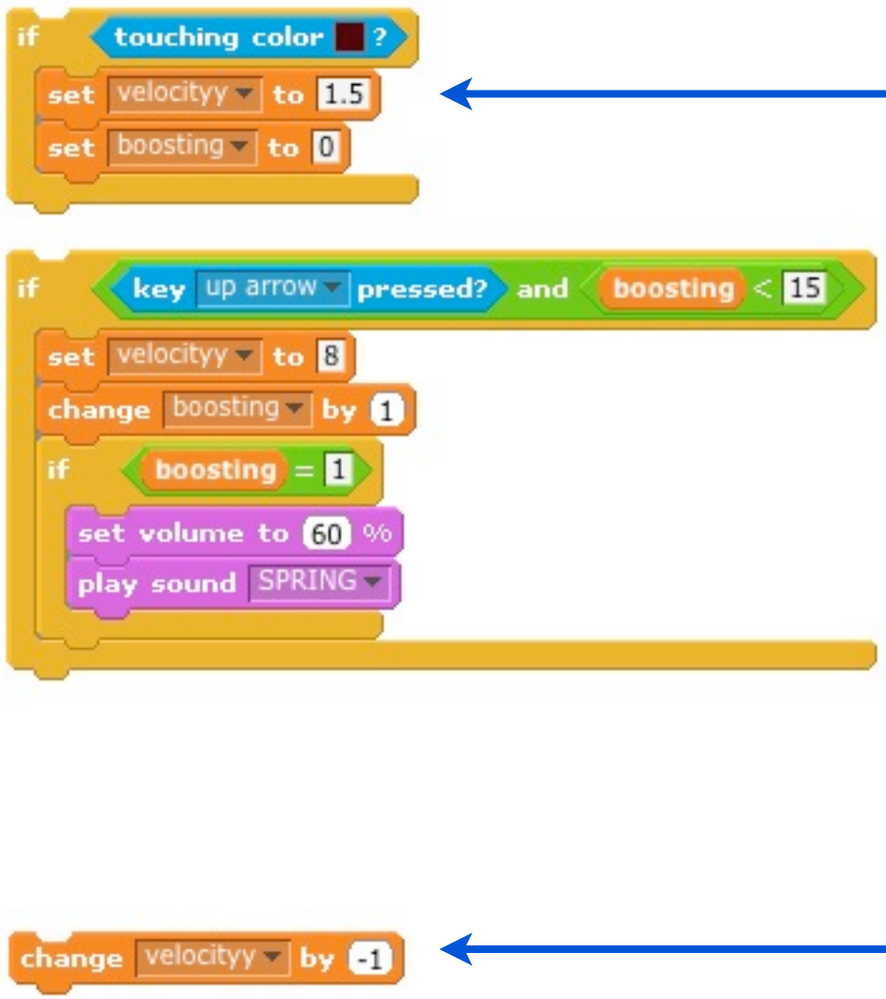
Finally test it you should be able to move left and right, and boost with your jetpack.

This “if” statement asks if the “HI-BOT” is touching the reddish-black that surrounds the platforms in the background image, if so stop “HI-BOT” from falling and re-charge the jet pack by setting the “boosting” back to 0.

This “if” statement checks if the “up arrow” and “boosting” is less than 15, set velocity to 8, moving the sprite up 8 pixels, change “boosting” by 1.

This “nested if” asks if “boosting” is equal to 1, and as above (is the up arrow being pressed and boosting is less than 15) then set the volume to 60% and play the spring sound. This plays the sound once at the start of the jump and not a possible 15 times during the jump.

Finally add this variable change to “velocityy” this pushes the robot down the screen. The combination of this pushing down and it touching a reddish-black outlined platform pushing up, gives it the animated appearance of a shaking jetpack.



Push “HI-BOT” up.

Push “HI-BOT” down.

Example 2

The full code for the “HI-BOT” object forever loop should now look like this.

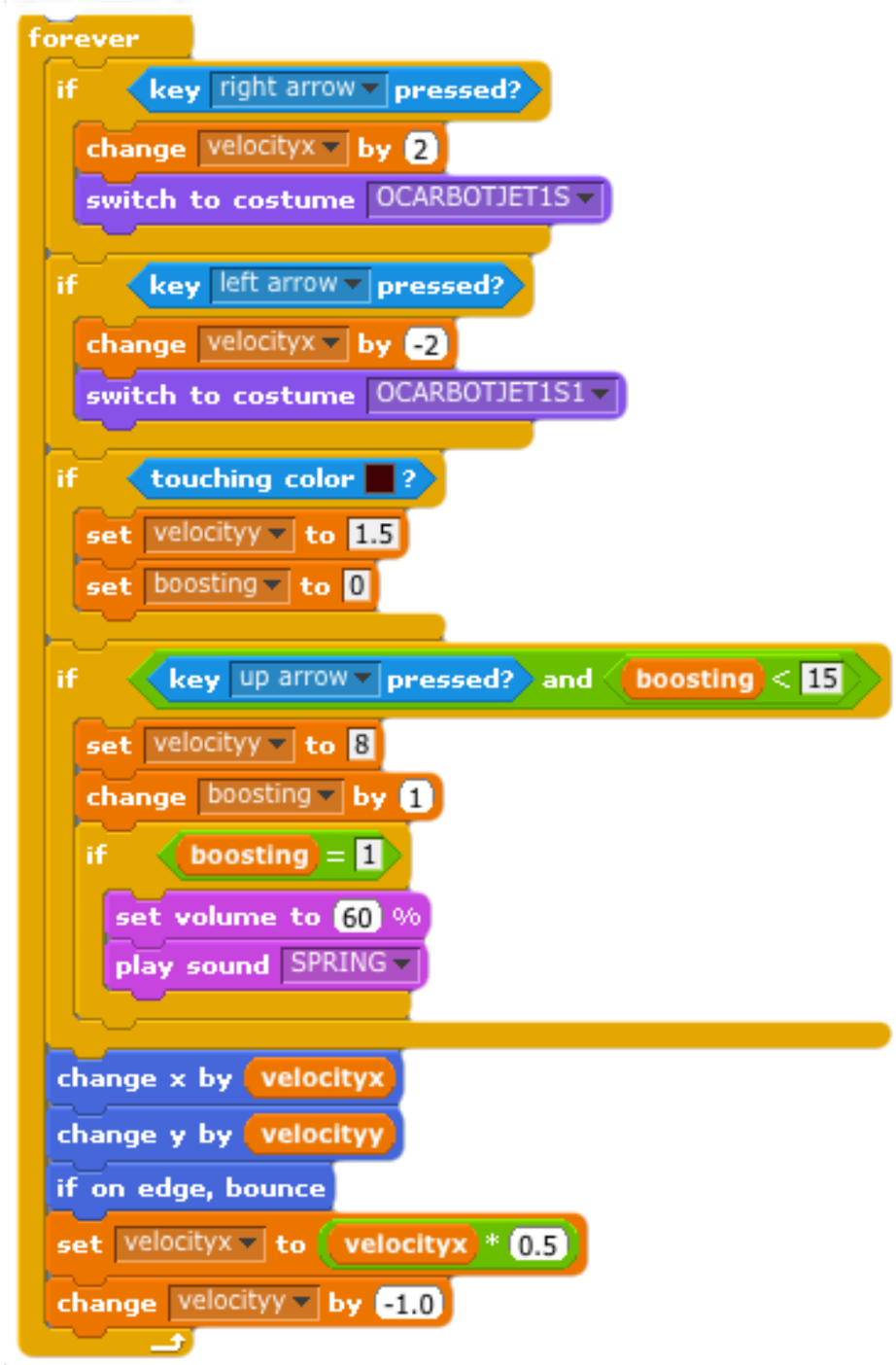
Checks the “right arrow” key.

Checks the “left arrow” key.

Checks collision with platform.

Checks the “up arrow” key.

Updates location of “HI-BOT”.



Example 3

Great our character can move left and right and Jump but we now need a bit of progression.
We need to set up a “Key”, an “EXIT”, and edit some of the “Stage” code this is what they will look like when completed.

“Stage” code

“Exit” code

“Key” code

Example 3

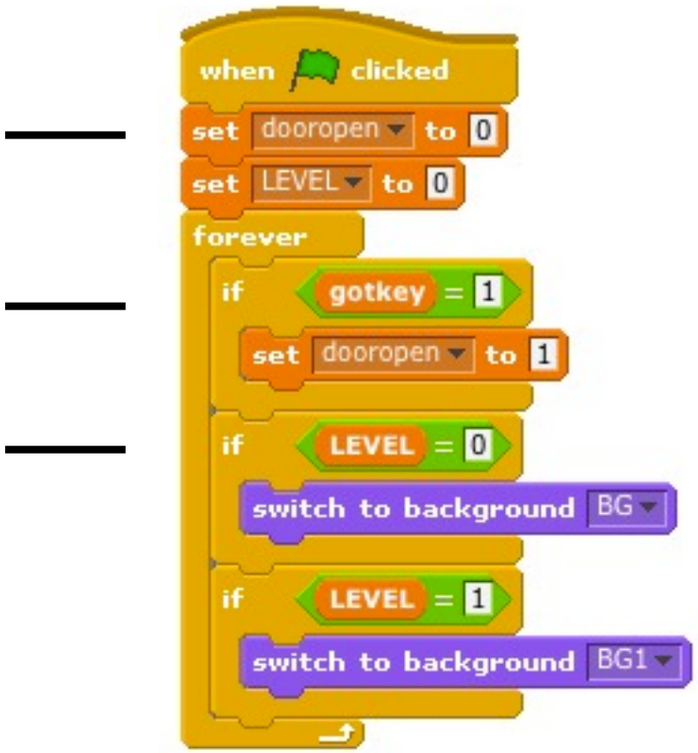
We need to set up a “Key”, an “EXIT”, and edit some of the stage properties. We will need to make some new variables which are accessible to every other object “dooropen” and “LEVEL”.

Step 1: Add this script to the “Stage”

It closes the “dooropen” to “0” to close door.
It sets “LEVEL” to “0” at the beginning of the game.
This is the end of the setup that runs at the beginning.

Within the “forever” loop it checks if you have picked up a key and opens the door by setting “dooropen” to “1”.

It also checks what level your on and puts the correct background on the screen.



Example 3

We need to choose a “new sprite from file” choose the “EXITBLOCK” and load “EXITBLOCK1” as an additional costume, which show a door open and closed respectively.

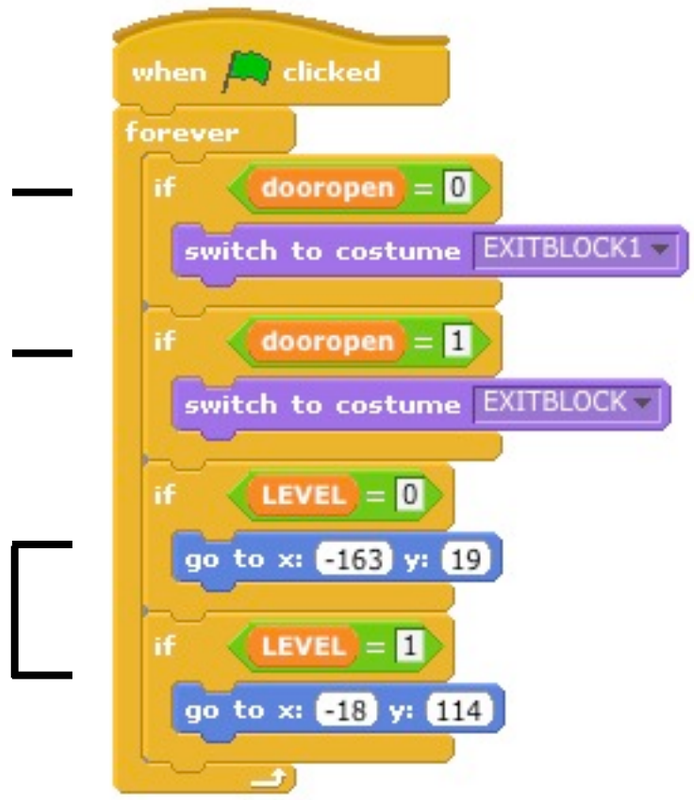
Step 2: Add this script to the “EXIT”

The “Stage” has already setup the new variables and that the door by default should be closed at the start.

This checks if the “dooropen” equals “0” and shows the closed door or....

“dooropen” equals “1” and should be open and shows the open door.

This areas of code puts the “EXIT” in the right place depending on what level we are on. Level “0” is the first level, remember computers count from “0”. The second level is LEVEL “1”



Example 3

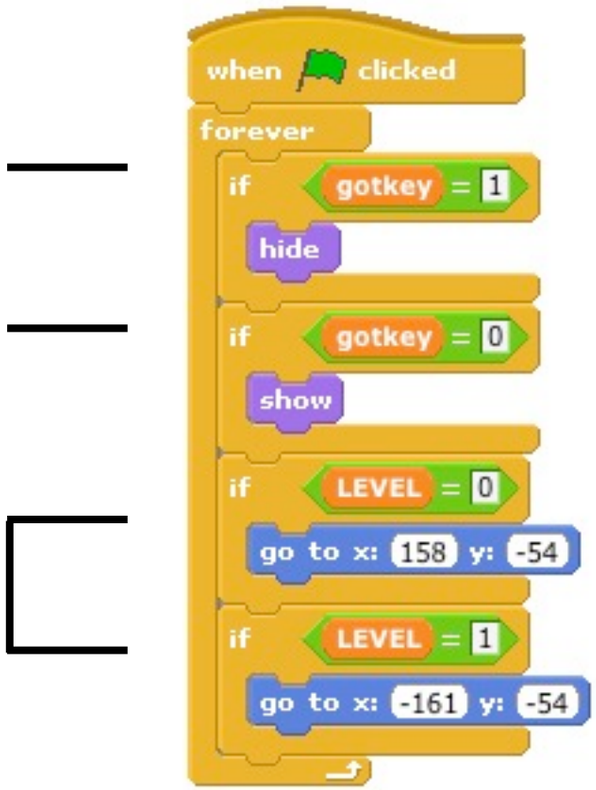
We need to choose a “new sprite from file” choose the “KEY”.

Step 3: Add this script to the “KEY”

This checks if “HI-BOT” has collected the key if “gotkey” equals “1” “hide” it so we can’t collect it again.

If not, and “gotkey” equals “0” then show it.

This areas of code puts the “KEY” in the right place depending on what level we are on. Level “0” is the first level, remember computers count from “0”. The second level is LEVEL “1”.



If we run the program now, nothing will be different as we have not told the “HI-BOT” what to do if it encounters a “KEY” or “EXIT” yet. So we need to go back to the “Sprite1” object and edit that once more.

Example 3

Almost there!
These two blocks of code go in the “Sprite1” area after the “Left”, “Right” and “Up” code blocks and before the screen is updated. You’ll also need to load in the “KEY” sound and the “EXIT” sound.

Step 3: Add this script to the “Sprite1”

This checks if “HI-BOT” has collected the key by checking the collision of the “lilac” of the robot against the “yellow” of the “KEY”, if this occurs it plays a sound and sets “gotkey” to “1”

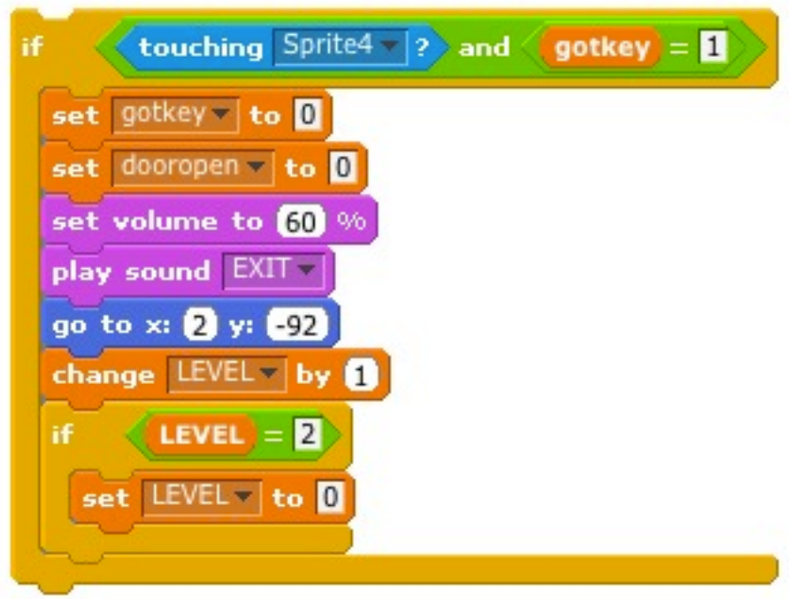
This checks if the “HI-BOT” is touching an exit and has collected a “KEY”, it then sets the “gotkey” to “0”, the “dooropen” to “0” setting up the next level of play.

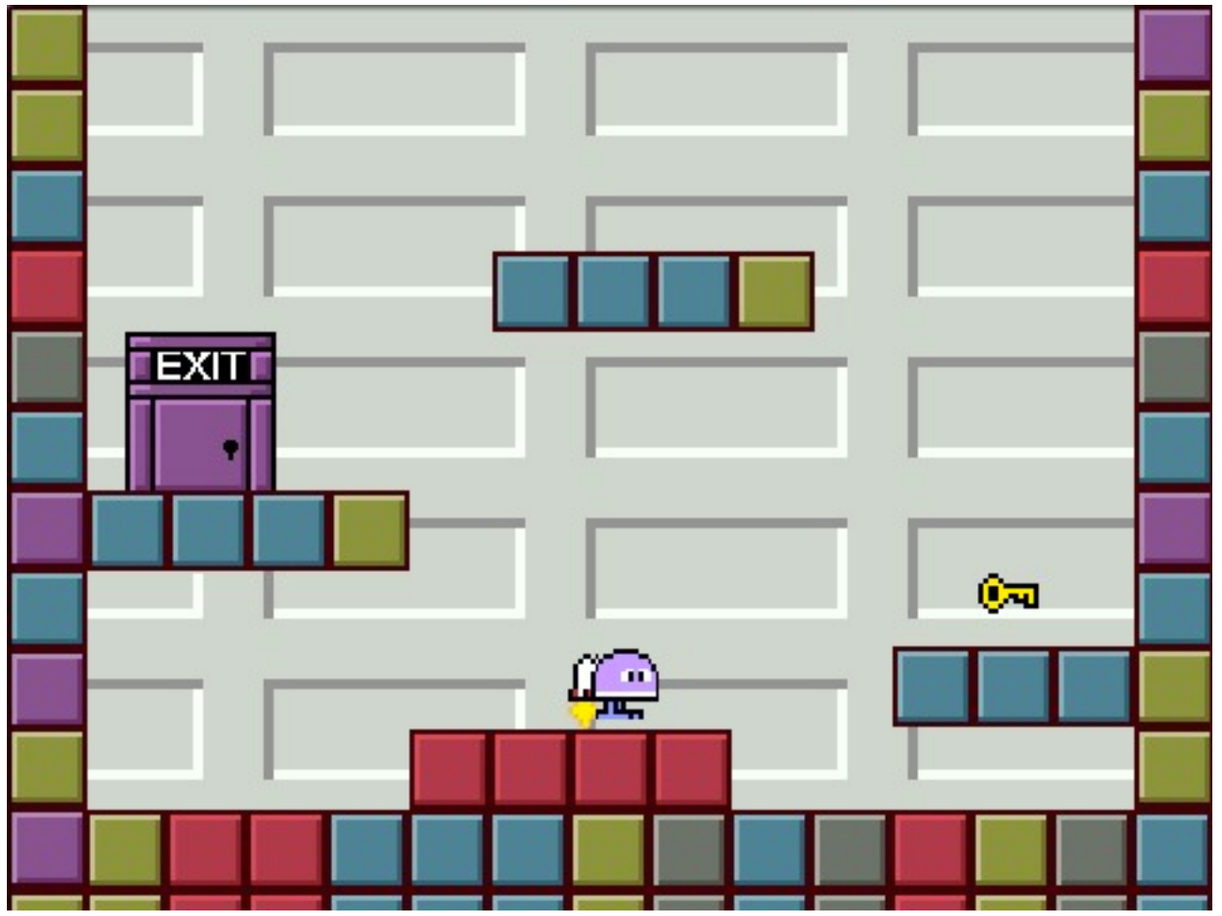
It then plays a sound.

Puts “HI-BOT” in the starting position for the next level, and then changes the current “LEVEL” by “1”.

If we get to “LEVEL” “2” we tell it to go back to level “0” as we haven’t made a “LEVEL” “2”, yet.

Test it, you should be able to, collect a key and exit the level. You can add more levels and obstacles now that you have the tools and know-how.





Now what?

Now that you have learned a bit about Processing and what it can do maybe start to think about what you could do with it.

Processing is a very powerful and fun language, with a huge community and hundreds of examples online. This was purely an introduction to the wonderful Processing language which is capable of so much more, the rest however is up to you.