CODERDOJO TRAMORE 2022 – SCRATCH TUTORIAL OCTOBER 2022

How to code your own Platformer game using Scratch!

[PART 2 – NEXT LEVEL]

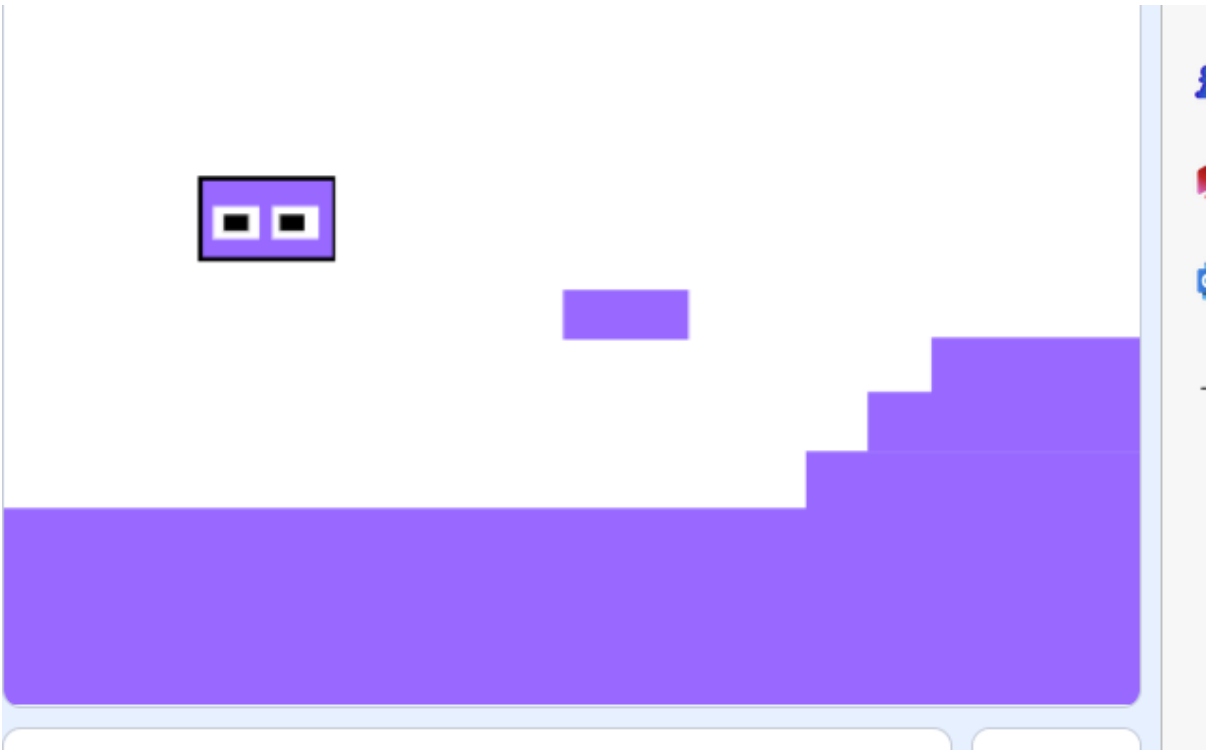
This tutorial series is a written break down of the beginner's guide to writing your own platformer game in scratch, as posted by the amazing Griffpatch on youtube.

This tutorial will describe the steps you need, but if you would like to see the original video at any point the link is as follows: -

https://youtu.be/HdFxavSE9H8

Please note this is part 2, you need to have completed the part 1 of this tutorial [Basics] before working on this part.

So to recap, we finished part 1 with this basic level designed, and all the physics in place to run, jump and collide realistically with the level parts: -



But while drawing the level as a backdrop costume got us up and running quite quickly, it is not the best way to design the level. We are relying heavily on "touching color" sensing to detect collisions and we need multiple of these if we want to add multiple colours.

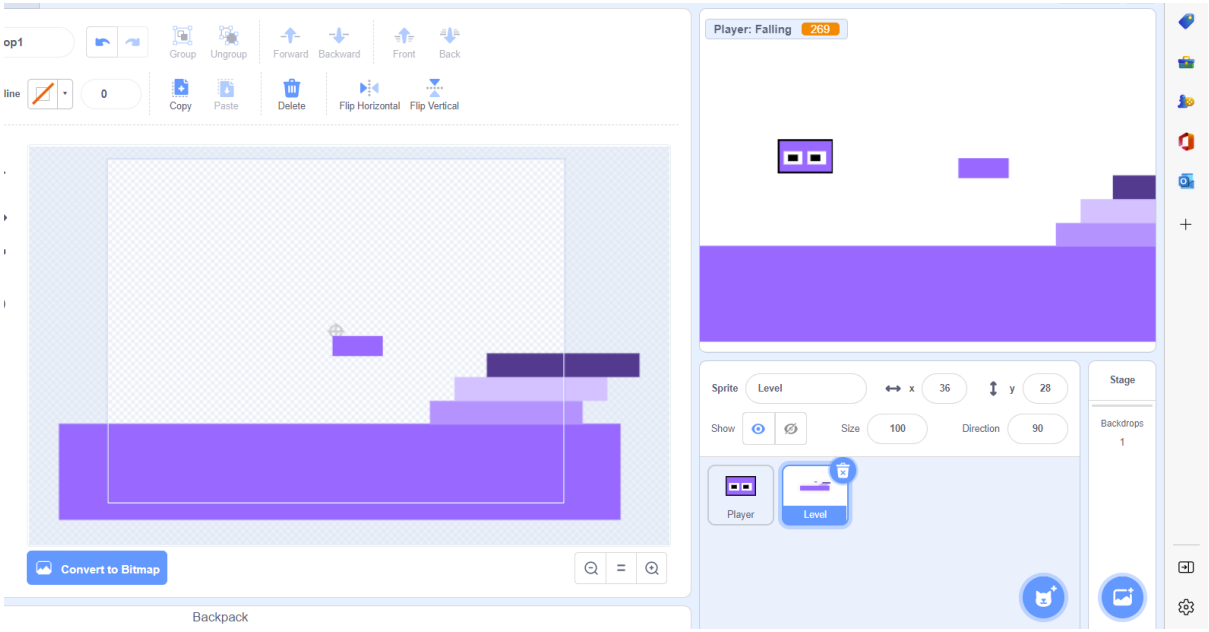
So Griffpatch walks us through changing the level to be a **Sprite** instead. Then we can use the "touching sprite" collision detection and now colour will not be a problem any longer.

So there is a quick way to copy our level from the backdrop costume to a sprite instead.

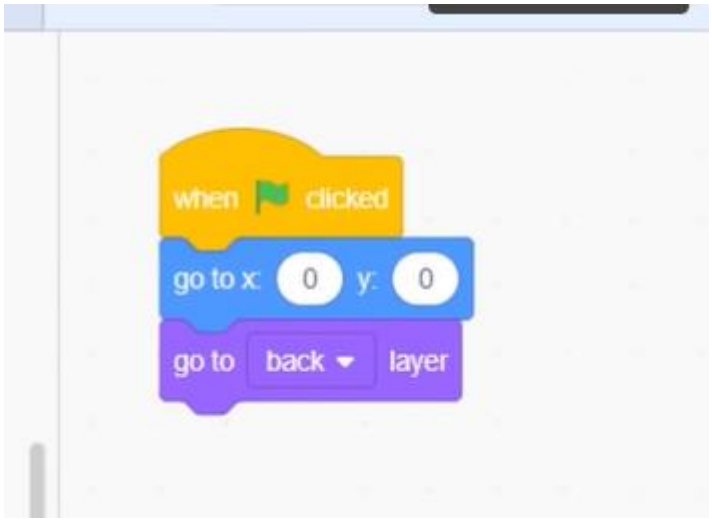
First create a new sprite and call it “Level”.

Now open backdrops costume design and drag the backdrop costume onto the new sprite icon, the costume will automatically be transferred to the sprite!

In the costume editor for the sprite, reselect each of the steps and change them to different colours, something like this: -



Now as this is a sprite that needs to be positioned, let’s give it the following small snippet of code: -



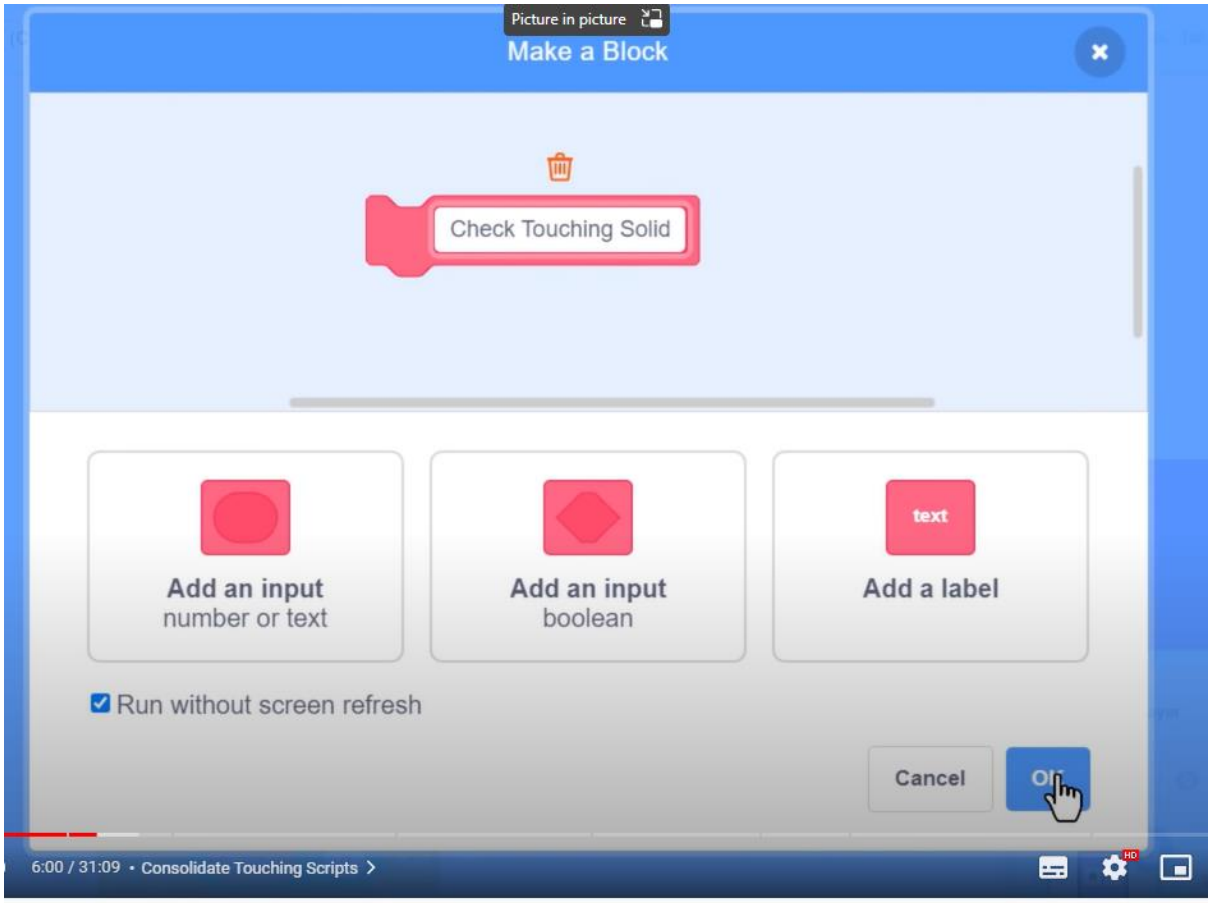
Go to the back layer means if there is any overlapping, this sprite is always drawn over by foreground sprites.

Now one more thing before we can test it out – go to the Player sprite code, and instead of using touching colour in the 2 collision detection spots, we will use “touching Level” instead:

Try it out and despite the now multi-coloured level, everything should work just as before -!

Now, as our platformer gets more and more sophisticated, we're going to have lots of different sprites to possibly collide with, so the next step is to tidy the collision detections into one tidy custom block.

So create a new custom block and call it "Check Touching Solid": - it doesn't need any input variables but make sure to check the "Run without screen refresh" checkbox: -



Add this code, you will need to create a new variable "touching" for this Sprite only: -

So basically, the touching variable will act like a flag so we can tell **outside** the custom block if we have a collision at the moment or not.

So now we can redo the touching detection in our main Player sprite by changing the code as follows: -

So we call “Check Touching Solid” before each of the Ifs now and then test afterwards if touching > 0, which will be true if touching comes back set to 1.

Run the program again to test, and once again everything should work normally as before.

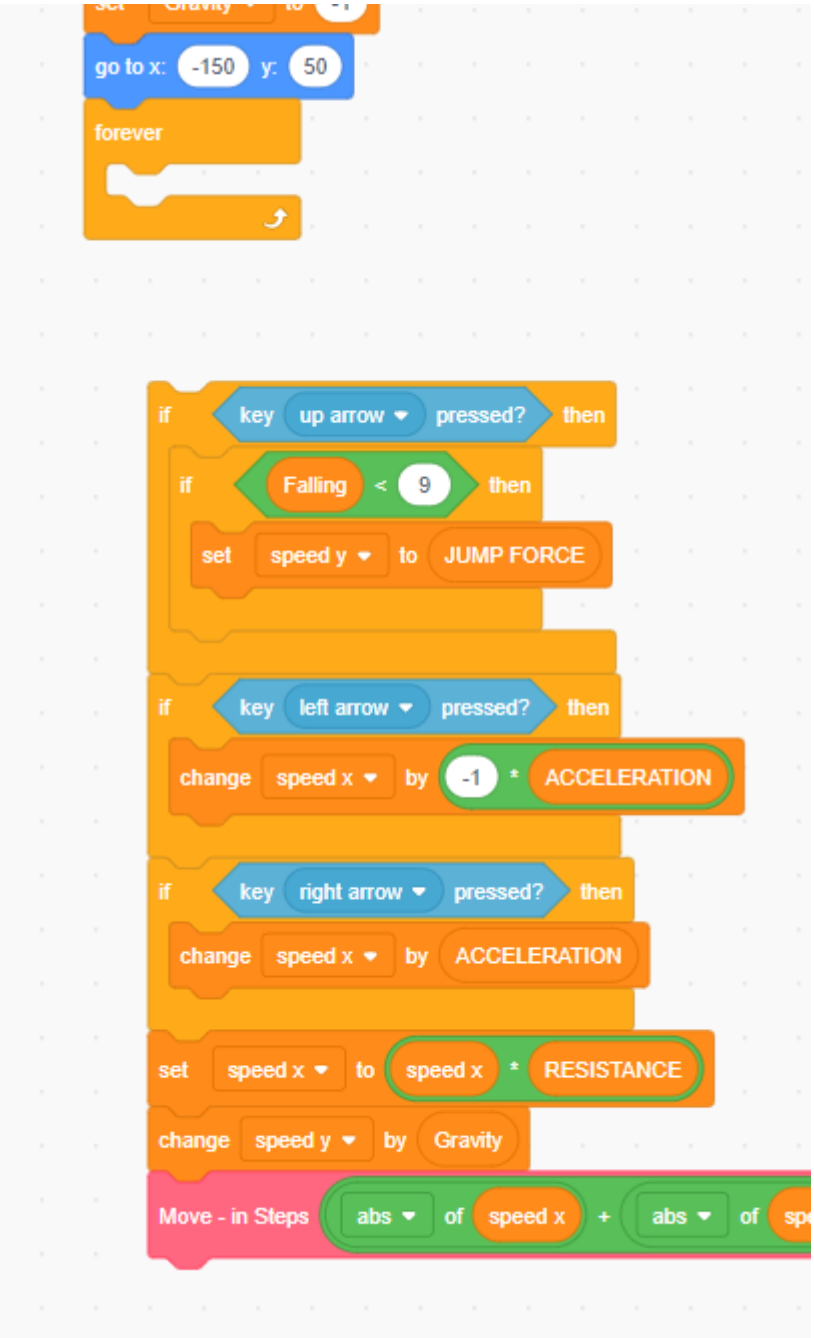
TIDYING UP THE CODE BEFORE MOVING ON

At this point, we’ve quite a lot of code put down, and its starting to get a bit messy and hard to follow the program flow.

So our next task is to do a bit of a tidy up and this will make the program easier to work on going forward.

First let's separate out the contents of the "Forever" loop – this is our main game loop: -

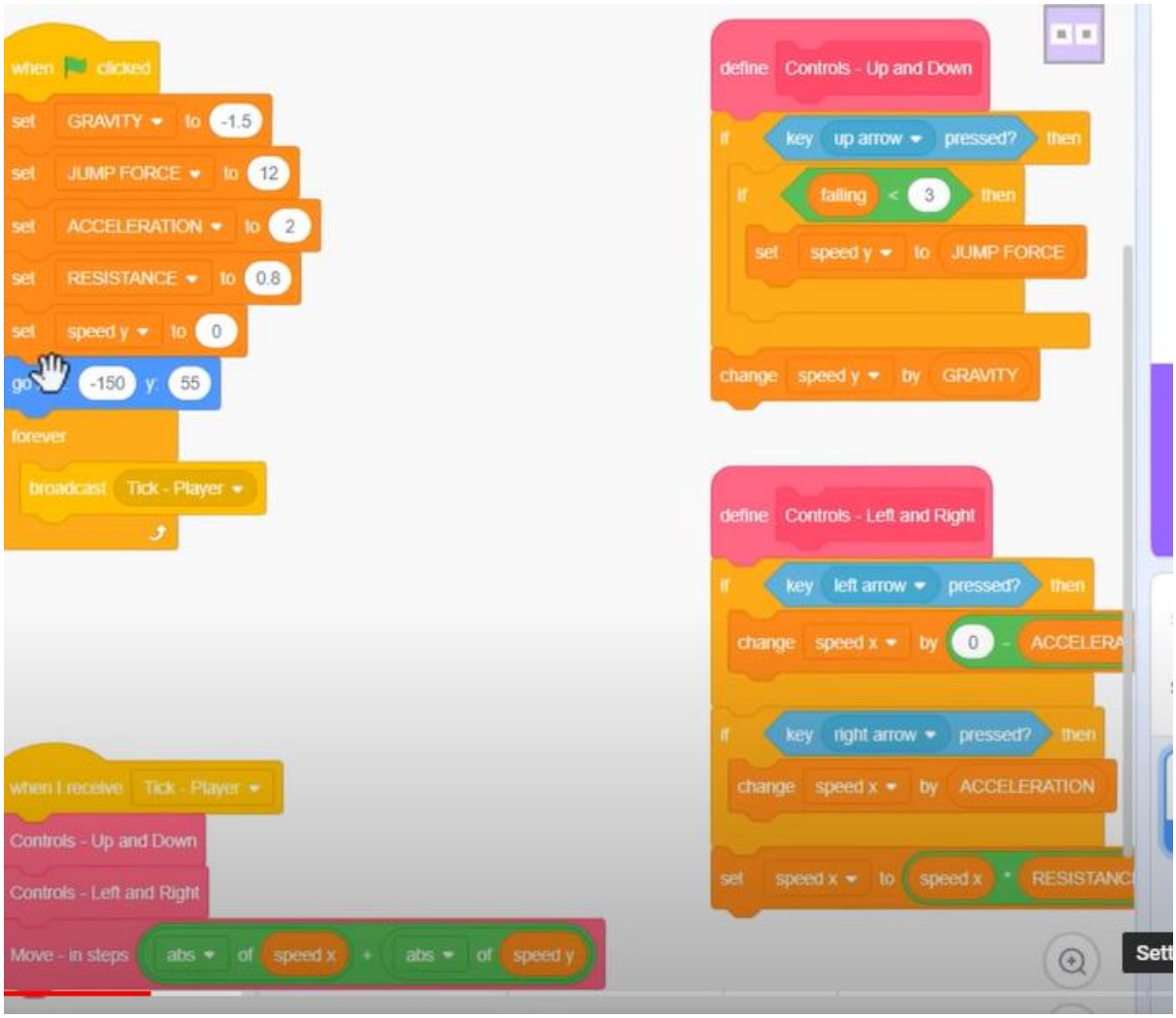
Pull all the blocks out of the forever loop like this: -



Now in the Forever loop put a broadcast message and call it "Tick – Player" tick is like the ticking of a clock, each "tick" is one frame in our game. Also at the top of the former forever loop contents add a "when I receive Tick – Player" block: -

And if you run this now, everything should work as before.

Next by defining custom blocks (always without screen refresh), you can further break up the code in when I receive Tick – Player as follows: -



Just note the change speed Y by Gravity has been moved up into the Controls Up and Down

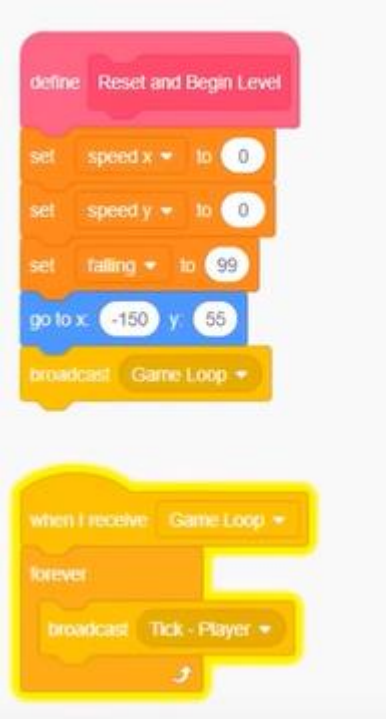
Section so that all the up and down movement controls are together.

And finally let's tidy up the game start process: -

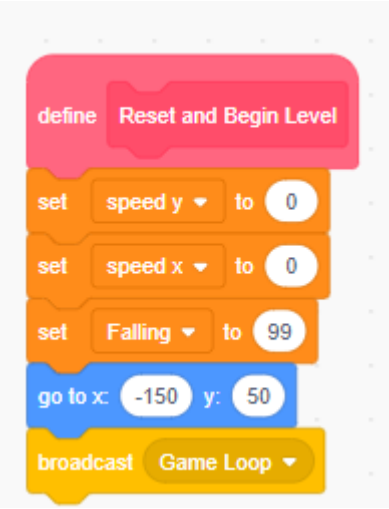
So one more custom block to be created "Reset and Begin Level" – **note do not tick the no screen refresh for this one**.

Add use this to reorganise the game start as follows: -

And finally the last edit – separate the forever loop and have it start by broadcasting a message “Game Loop”: -



Move the initial Set of SpeedY to 0 to this “Reset and Begin” custom block, and add a set SpeedX to 0 also – and add a set falling to 99 – 99 means that when Guy initially starts falling at the very start of the level, you can’t jump in the air.



Now after all that housecleaning, we are ready to make our next “level” screen, this screen will be accessed by moving off the right of Level 1.

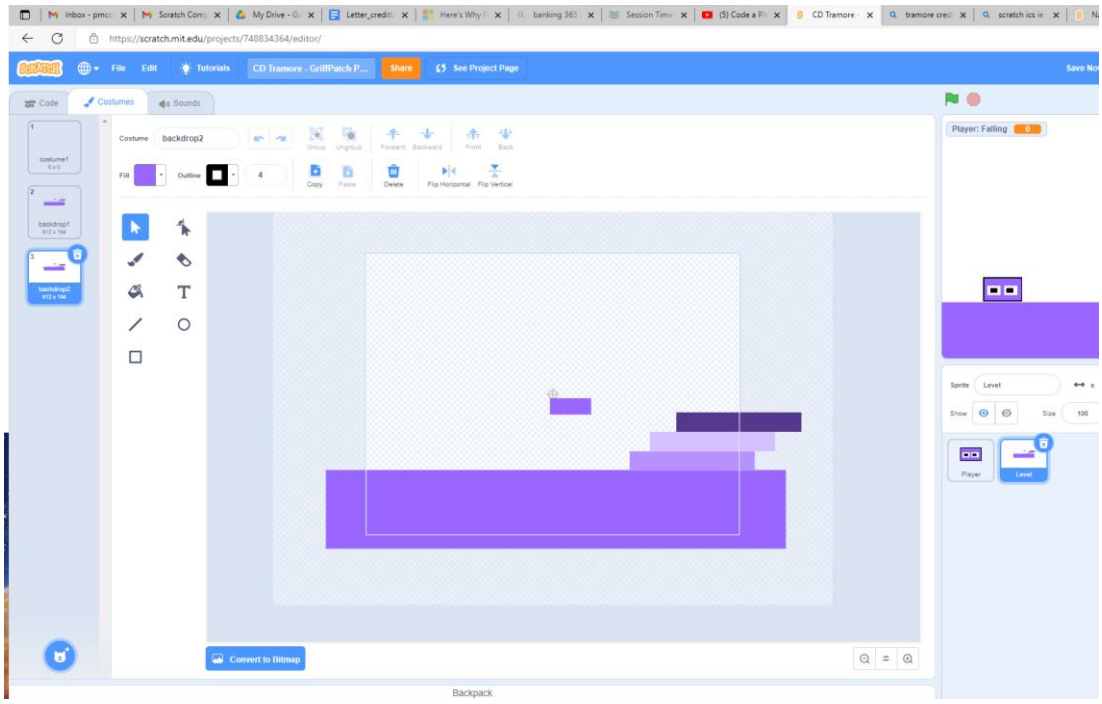
Creating and linking Level 2

Go to the Costumes editor for our level sprite and duplicate the 1st level to a new costume: -

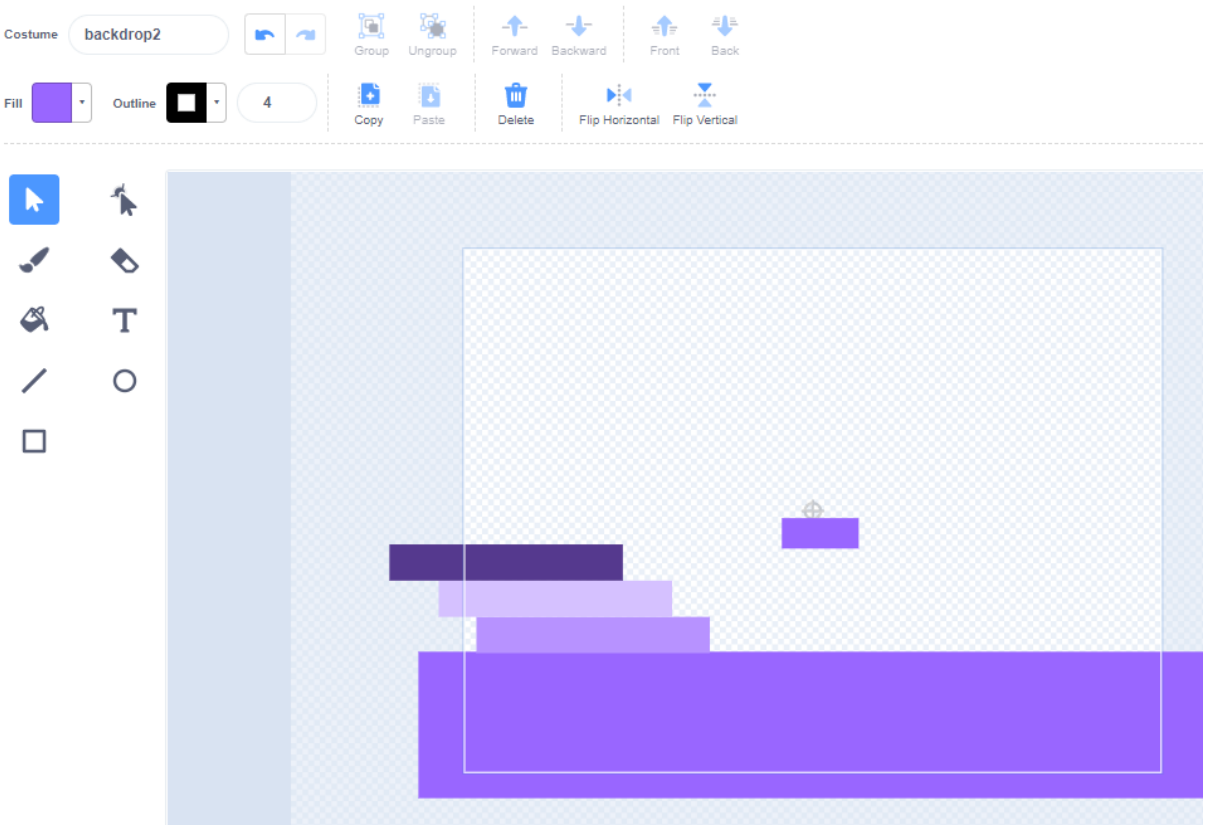
Now we want to “flip” this level horizontally so that the left of this level matches nicely to the right of level 1.

There is a “flip horizontal” function in the top icons but if you try it at the moment it won’t work the way we want: -

Before: -

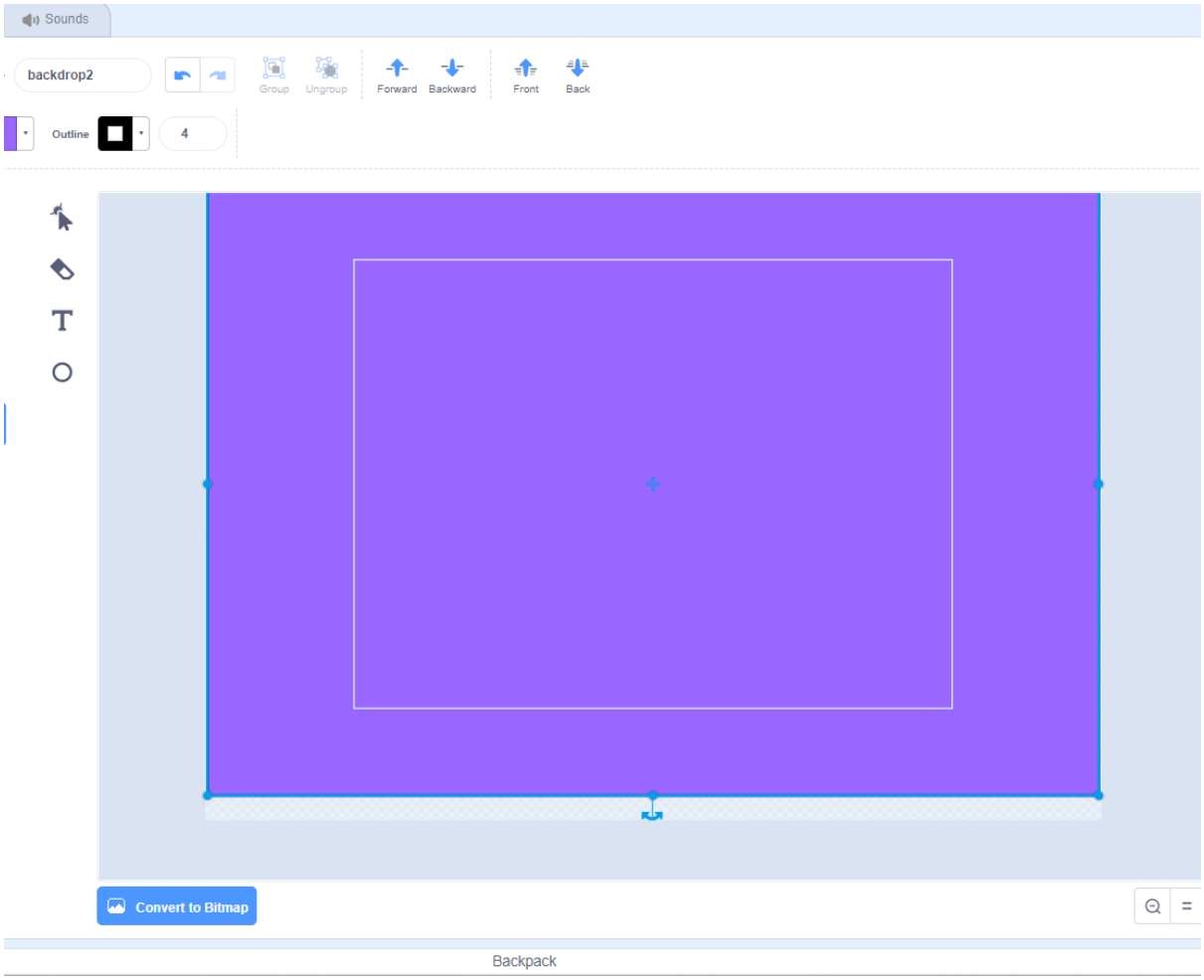


And after: -

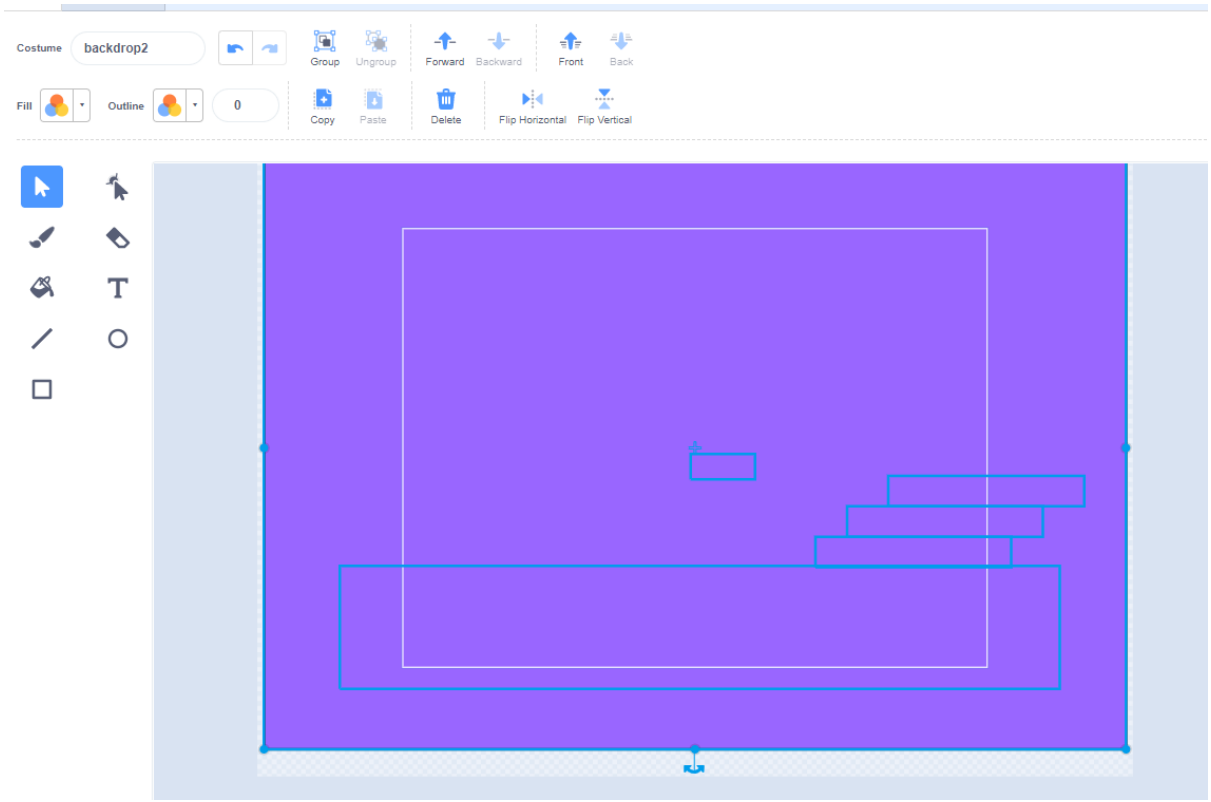


It’s not quite spot on, Griffpatch has a tip to make this flip more accurate: -

The trick is to draw a new temporary rectangle as big as you can, filling the full canvas – then make sure it is centered also: -



Use the select tool to select all objects in the costume: -

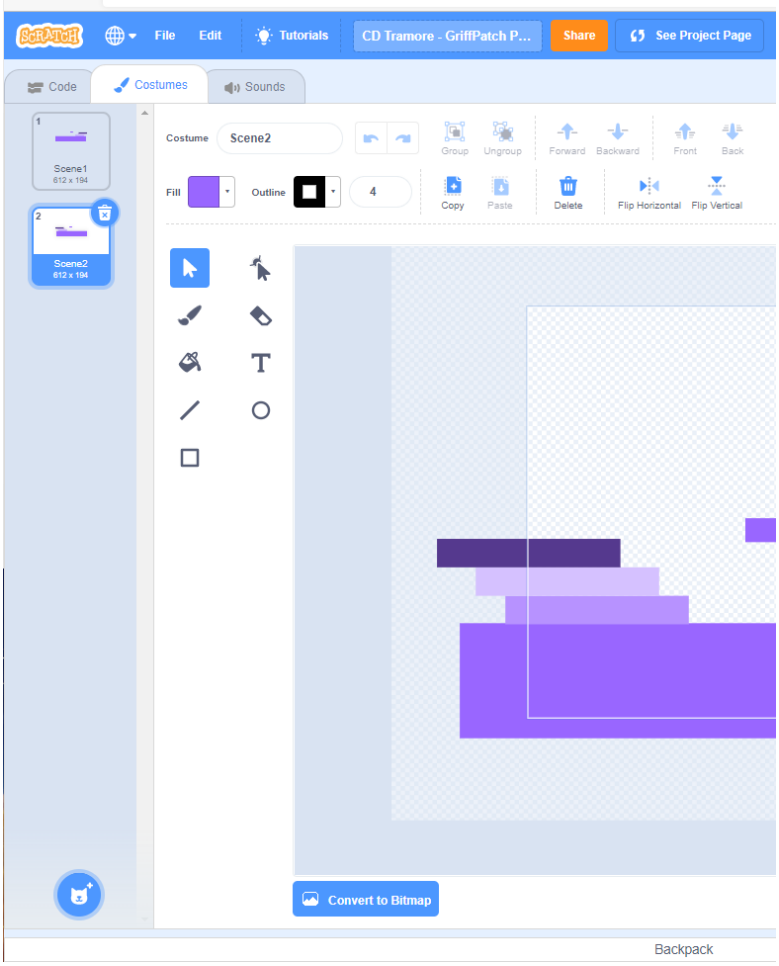


And now the flip horizontal will mirror perfectly, and after this you can remove the temporary box placed on top: -

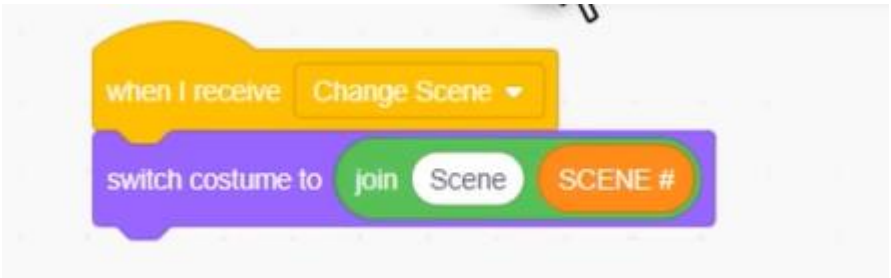
Ta da!

Now rename each of your two levels **exactly** “Scene1” and “Scene2” (including capital S) – this is important because we will be using code to built the costume names later,

Also remove any blank initial costume if your sprite still has one, so after this you should have: -



Now add this code to the Level sprite’s code, creating a variable Scene # **for all sprites** along the way: -

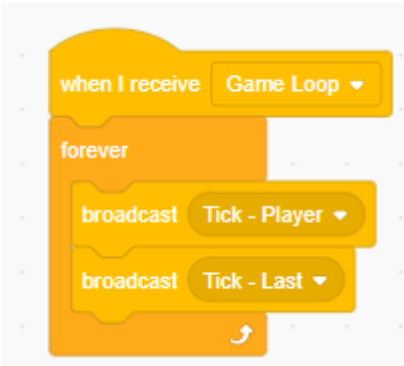


This uses the join to stick the text “Scene” and whatever the current level number in Scene# together to make the correct costume name and then switches to that costume.

Next, return to the Player sprite's code and change the Reset custom block so that the game begins on Level 1: -

We're getting there, but the scene still doesn't automatically change when we move off the right of the screen, so let's add that in now: -

Add a new broadcast message in the game loop AFTER "broadcast Tick – Player" and call this one "broadcast Tick – Last": -



Now add the following to handle Tick – Last : -

What this does is, if the Player's current X (left/right) position on screen is greater than 235, then it increases the level variable by 1 and broadcasts change scene which will update the levels' own costume.

Now you can test that – and it should work, sort of – you can move off the right and the level will change – but the player is still on the right of the screen!

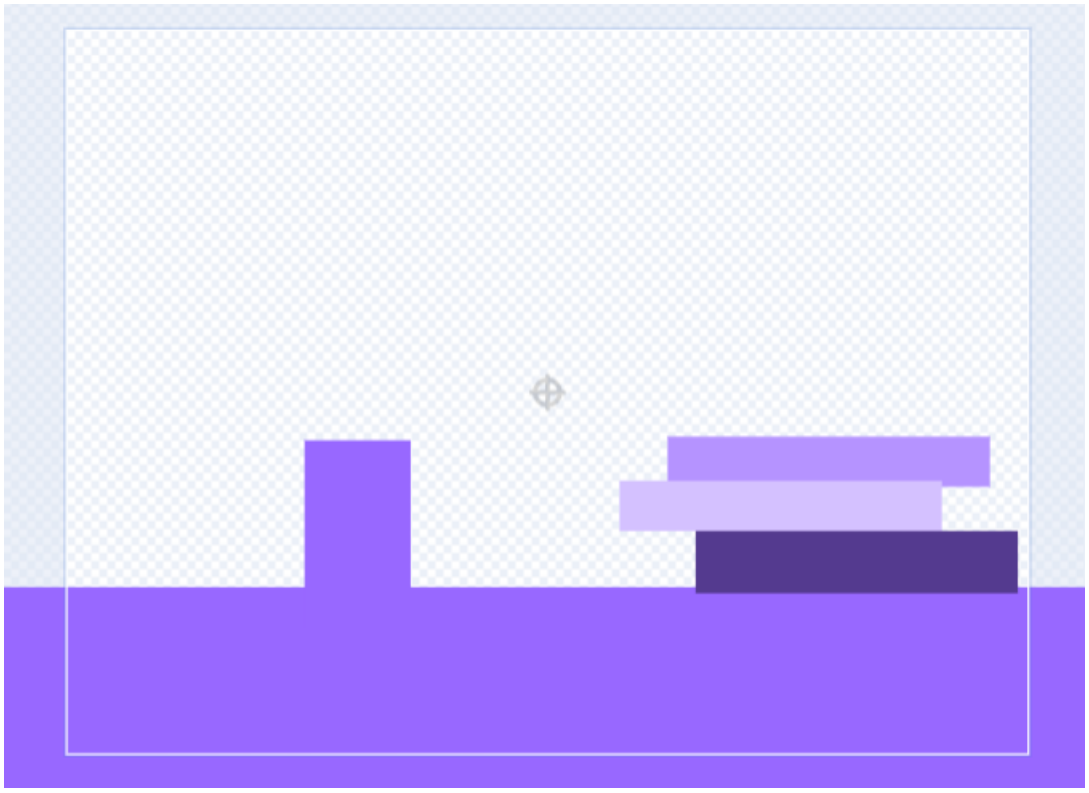
We can fix that fairly easily – we just need to remember to also set the player's X position to **minus 235** :-

And we're done – you can now move into level 2 and the player changes position on screen just as you would expect.

This of course will only allow you to move from left to right, we also need some code to be able to move off the left of the screen, so we duplicate and do more or less the same just in reverse: -

OK now let's add another level by duplicating the costume "Scene2" to "Scene3" and then editing it: -

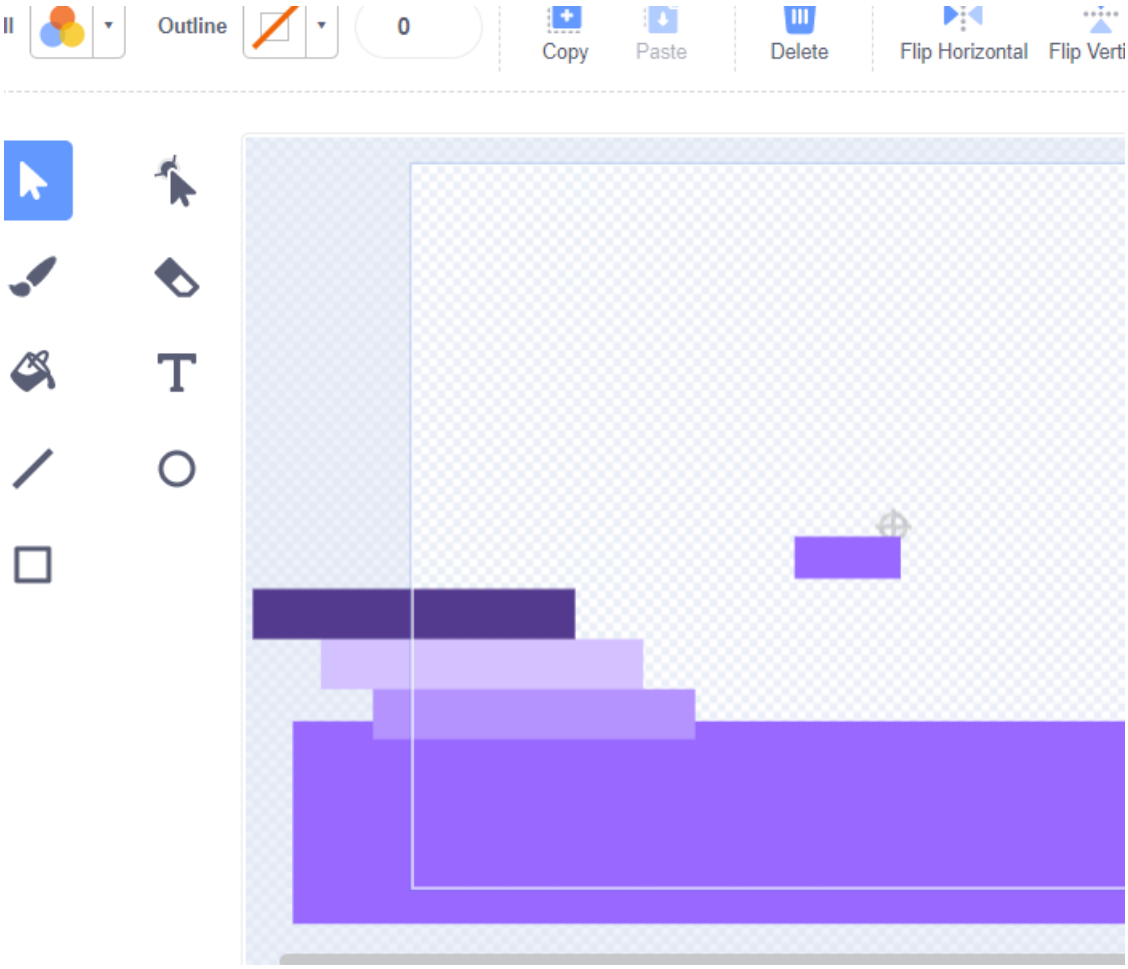
You can go with your own design but perhaps something a bit like this: -



And now so long as the costume is called exactly "Scene3", you can now access this level straight away by moving off the right of Level 2 – try it out!

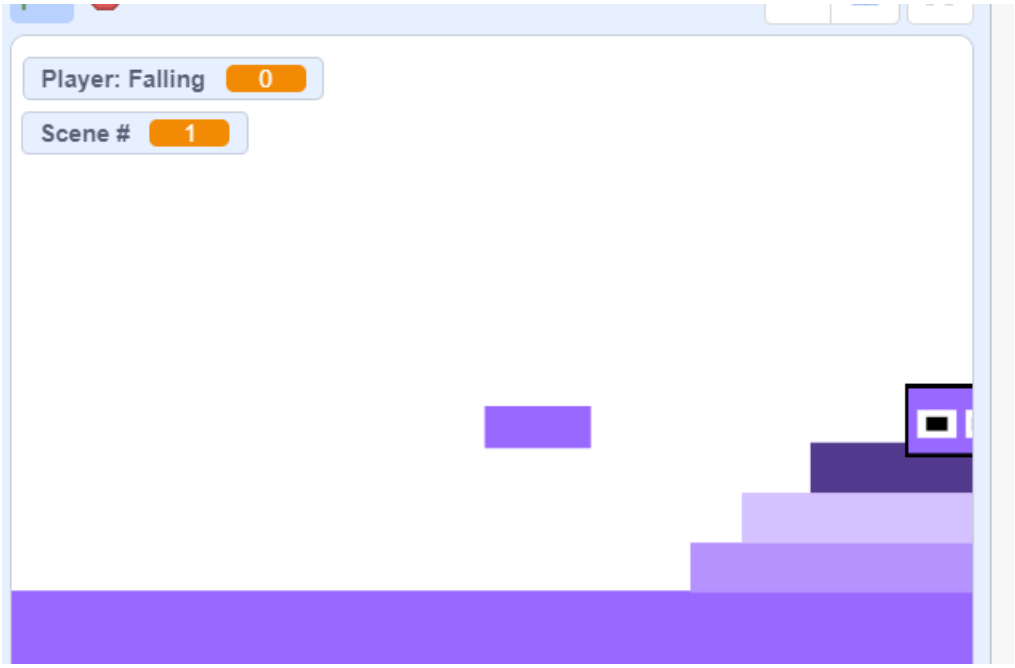
Now as you design your levels, if the floor level is not exactly the same down to the pixel, at the moment a bug will occur and the player will become stuck – let's demonstrate this.

In the costume editor, select the "steps" of level2 and move them down slightly, make sure to unselect them and save your project before restarting to test: -



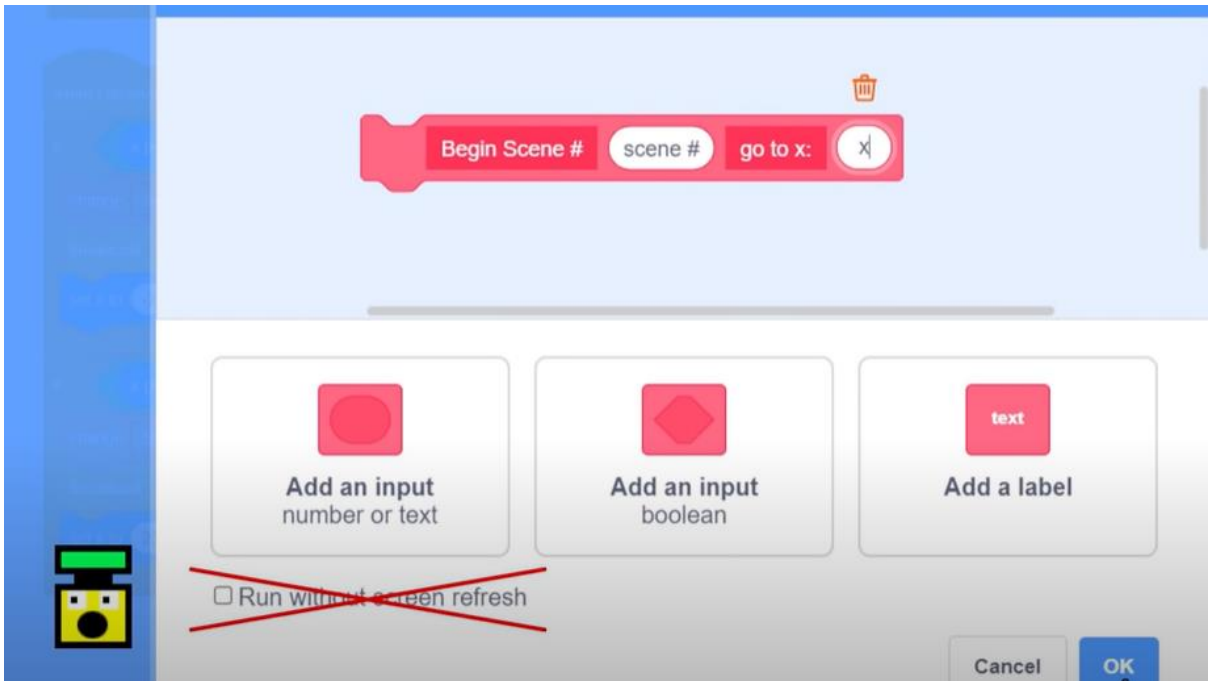
Now when you run your program, you can move from level 1 to level 2 ok, but moving back to level 1 should see your player stuck permanently in the ground: -

Ouch!



So what Griff adds next is some code to detect when this has happened, and move the player up to floor level much like the falling and moving collision detection.

First a bit of tidying in the “Tick – Last” event handler, the screen move right and move left sections are very similar, as we’re going to expand them now, let’s code them both into one custom block – create the following custom block: -



“Begin Scene #” is a number input, “Go to X”