Pen Platformer Part 2 – Making the player jump/fall/detect floors and ceilings

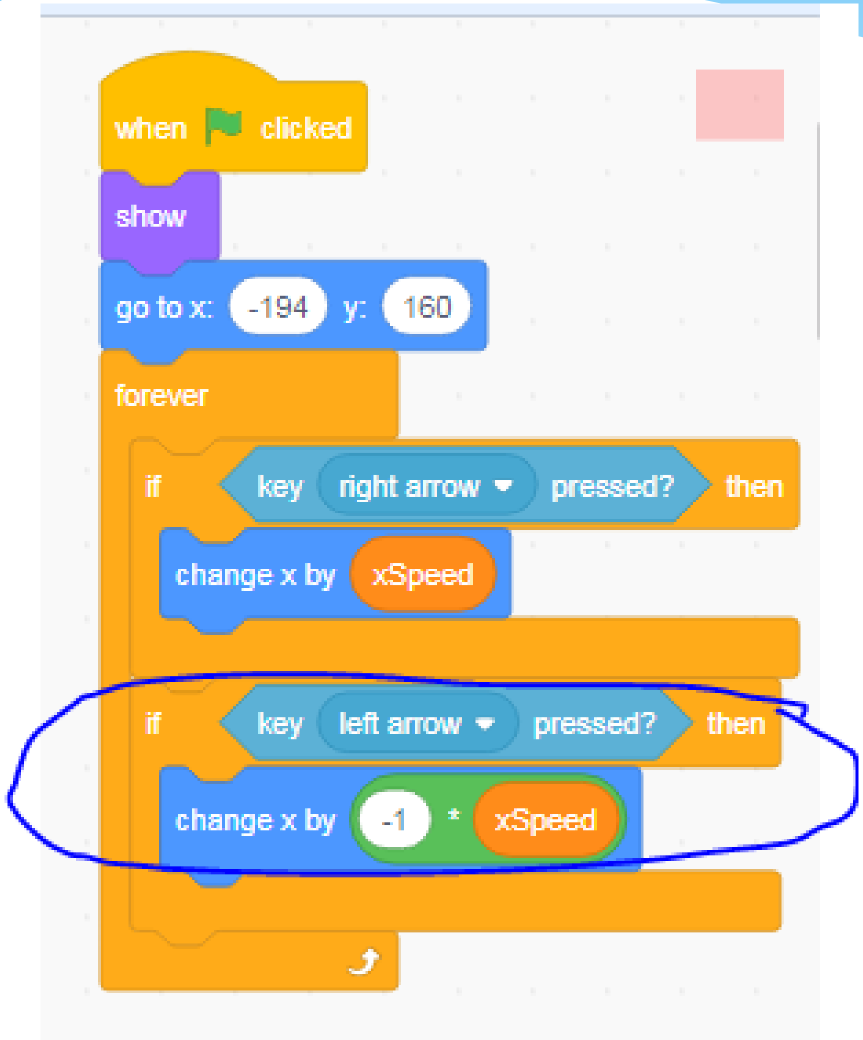
Vineet Srivastava

In this lesson, we will …

* Write code to give movement to our player sprite.
* We will see how the player can jump, fall, go over obstacles, detect ceilings and so on.
* In this process, we will also complete the Pen Platformer game.

WIBYTE

Sprite Moving Left also



```
when [flag] clicked
show
go to x: -194 y: 160
forever
    if <key right arrow pressed?> then
        change x by xSpeed
    if <key left arrow pressed?> then
        change x by (-1 * xSpeed)
```

Notice, this code will make the sprite go LEFT also.

Also, this gives an idea on why we should not use If touching EDGE to go to next level.

Gravity Effect

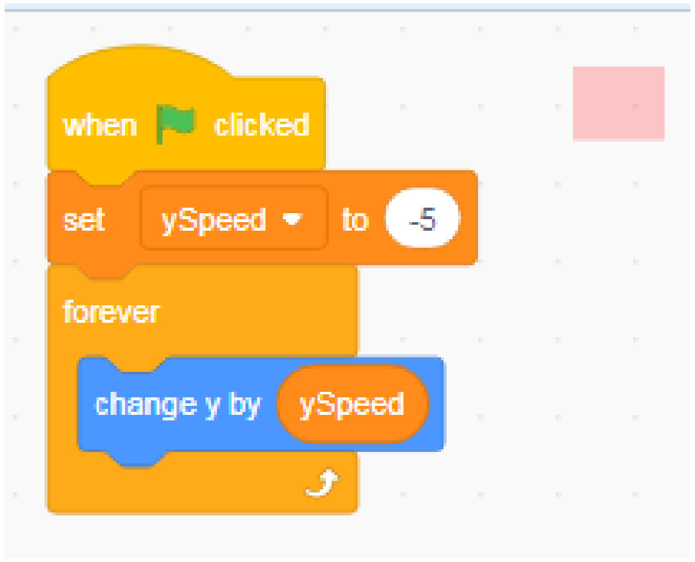
- We want the player to fall – sort of how real objects fall – under the influence of gravity.
- At the same time, we want the player to come to a rest when touching the platform or the floor.
- This looks very simple, but needs some amount of careful coding.
- Let's see this in a step-by-step manner.

How do things fall in the real world?

- In the past, when we made the ‘CATCH GAME’, we had objects falling. Basically we changed their y position forever.
- But, in the real world, things become FASTER as they fall.
- Let’s see how we can achieve this.

Simple fall vs fall with gravity

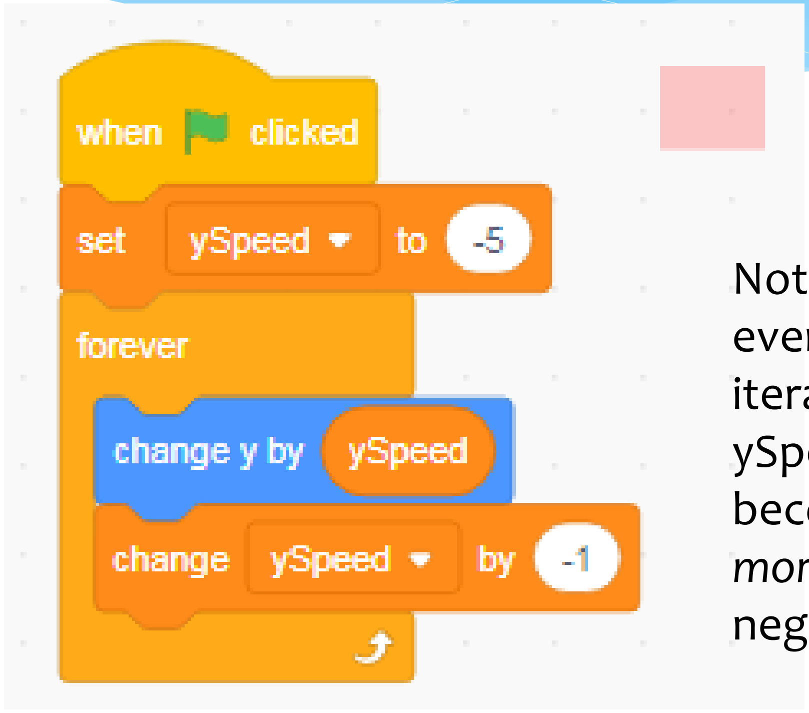
Simple Fall



```
when 🏴 clicked
set ySpeed ▾ to -5
forever
    change y by ySpeed
```

Y keeps reducing by a fixed amount

Fall with Gravity

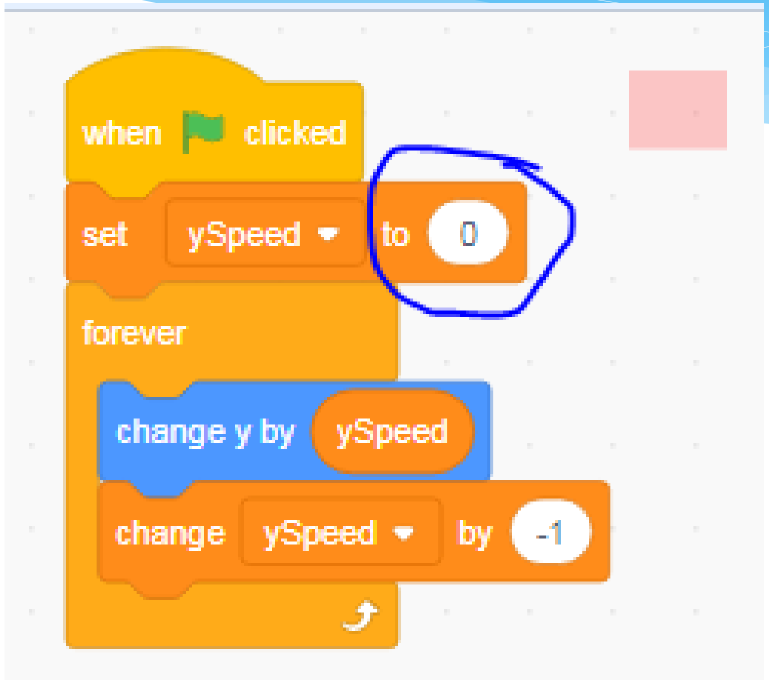


```
when 🏴 clicked
set ySpeed ▾ to -5
forever
    change y by ySpeed
    change ySpeed ▾ by -1
```

Notice, in every iteration, ySpeed becomes *more* negative

Y keeps reducing by an *increasing* amount

A free fall starting from ‘rest’!



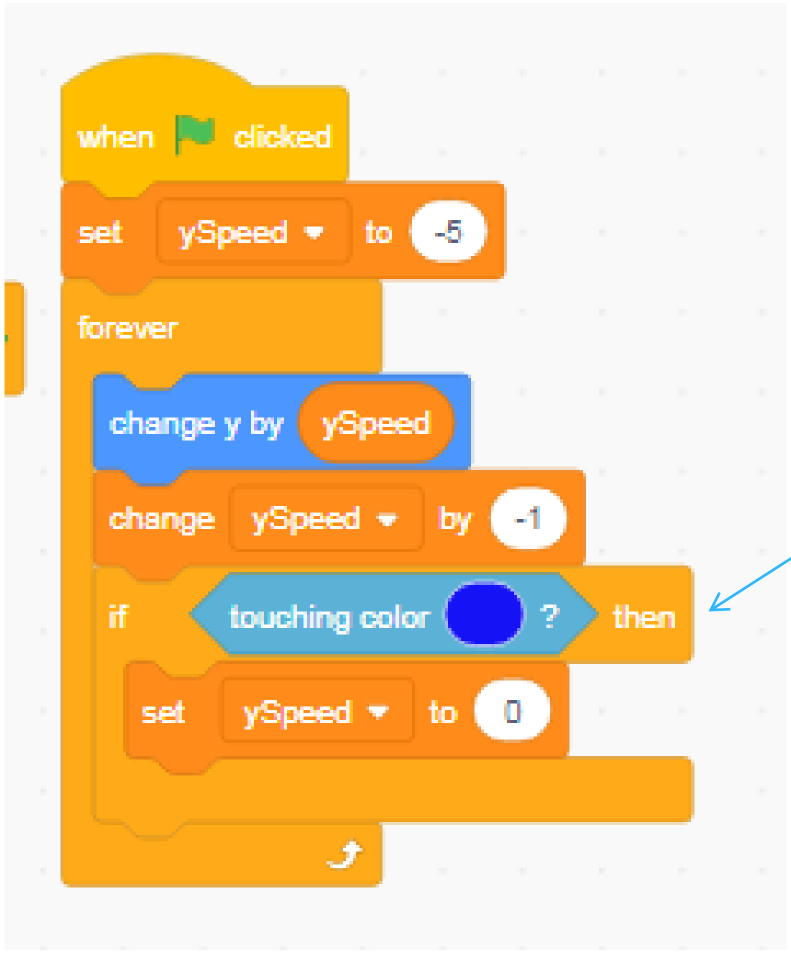
```
when [flag] clicked
set ySpeed to 0
forever
    change y by ySpeed
    change ySpeed by -1
```

Set ySpeed to 0 in the beginning, Thereafter it keeps changing by -1 every iteration

Stopping the FALL on touching the platform

- Recall, the platform is really just a block that we have drawn using the PEN EXTENSION.
- Hence, we can make use of the <when Touching COLOUR> block.
- But its takes more than just that.

Stopping the fall – A simple method

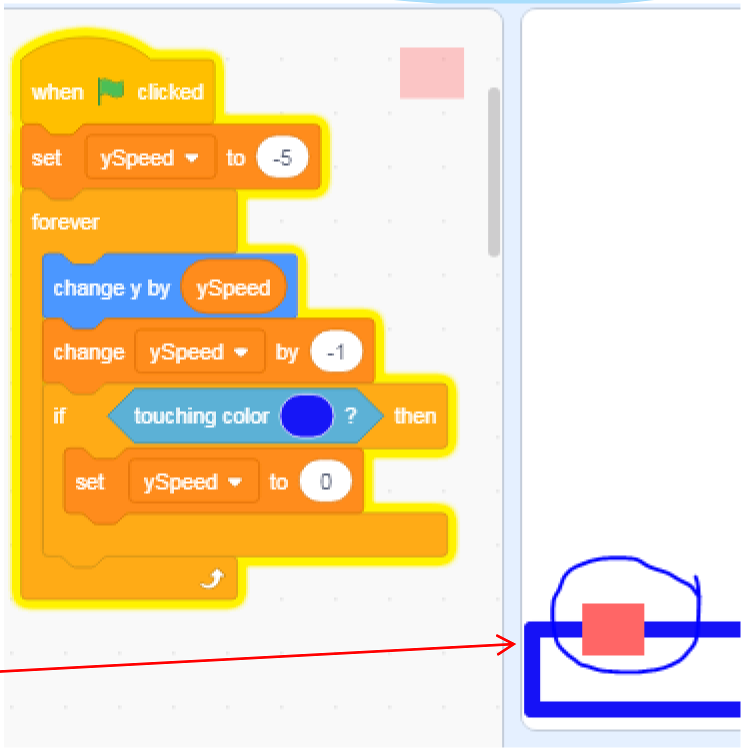


```
when 🏁 clicked
set ySpeed to -5
forever
    change y by ySpeed
    change ySpeed by -1
    if touching color ● ? then
        set ySpeed to 0
```

I can use 'TOUCHING COLOR' and set the ySpeed to 0.

Why does the block not 'STOP' at the first touch

- The player does not quite stop very cleanly.

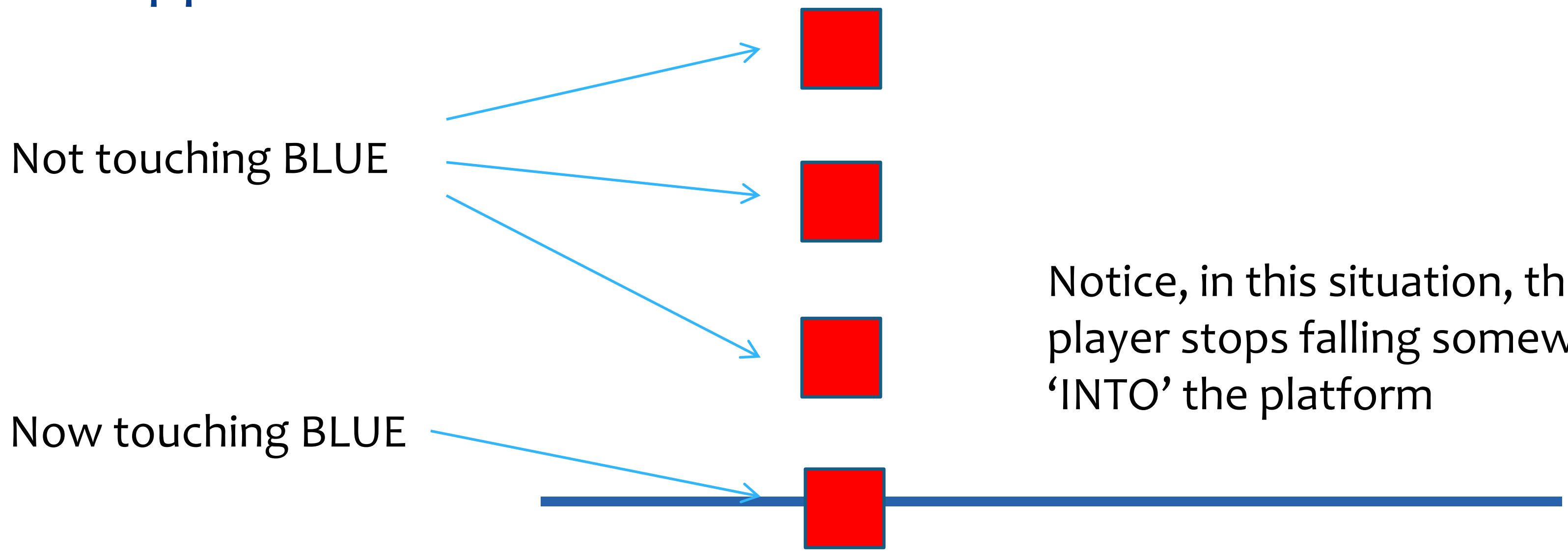


```
when 🏴 clicked
set ySpeed to -5
forever
    change y by ySpeed
    change ySpeed by -1
    if touching color ● ? then
        set ySpeed to 0
```

Notice, there are occasions where the player seems to stop somewhere 'inside' the platform

Why does this happen?

* This is happening because we are not able to 'perfectly' control where the first 'TOUCHING event' happens

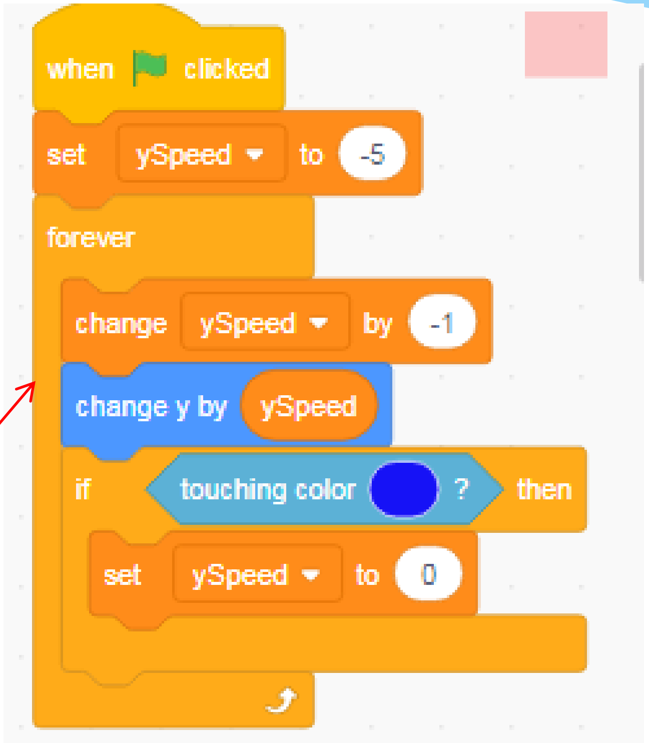


Notice, in this situation, the player stops falling somewhat 'INTO' the platform

To See this in the code, you can add a small WAIT in the previous FOREVER loop.

Food for thought

* Will this work?



```
when [flag] clicked
set ySpeed to -5
forever
    change ySpeed by -1
    change y by (ySpeed)
    if <touching color [●]?> then
        set ySpeed to 0
```

You will see that now the y position will keep decreasing – it will look like the object keeps falling – can you see why.

(See the sequence of events very carefully).

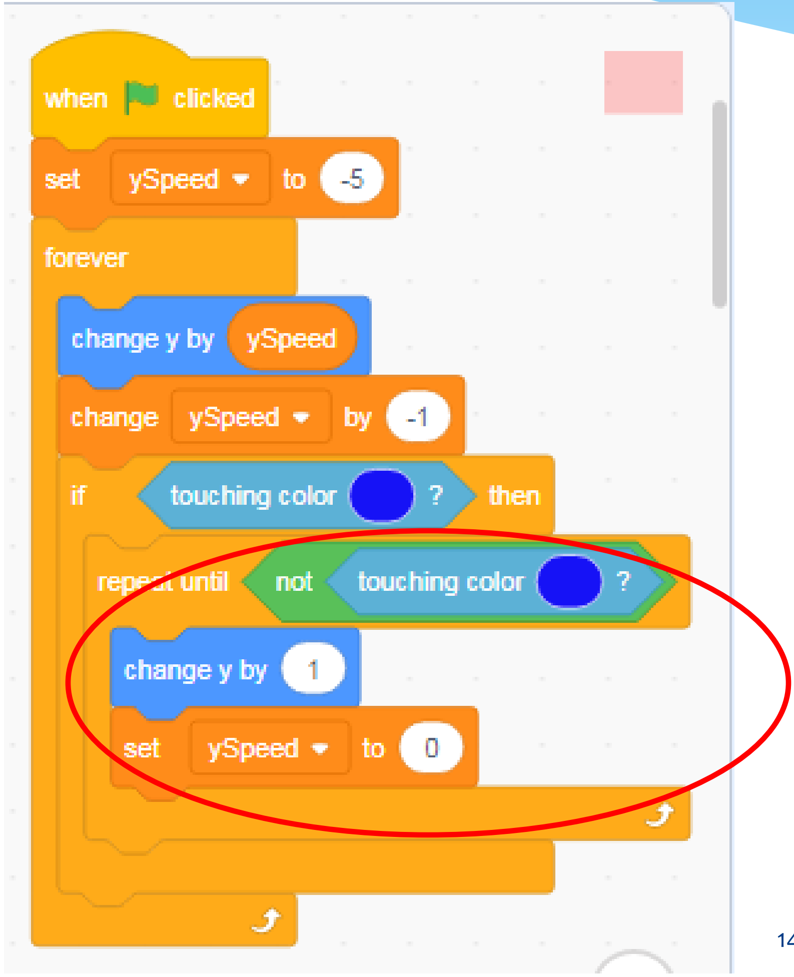
Notice, we have now interchanged these two

WIBYTE

Can I pull this back up?

- There is almost no way I can control what we just discussed, that is player going somewhat ‘INTO’ the platform.
- Hence all we can do is to ‘pull’ it out.
- One possibility is to keep increasing its y in small amounts until the player is **no longer** touching the platform.
- We can hence use REPEAT UNTIL.

Pulling out the sprite!



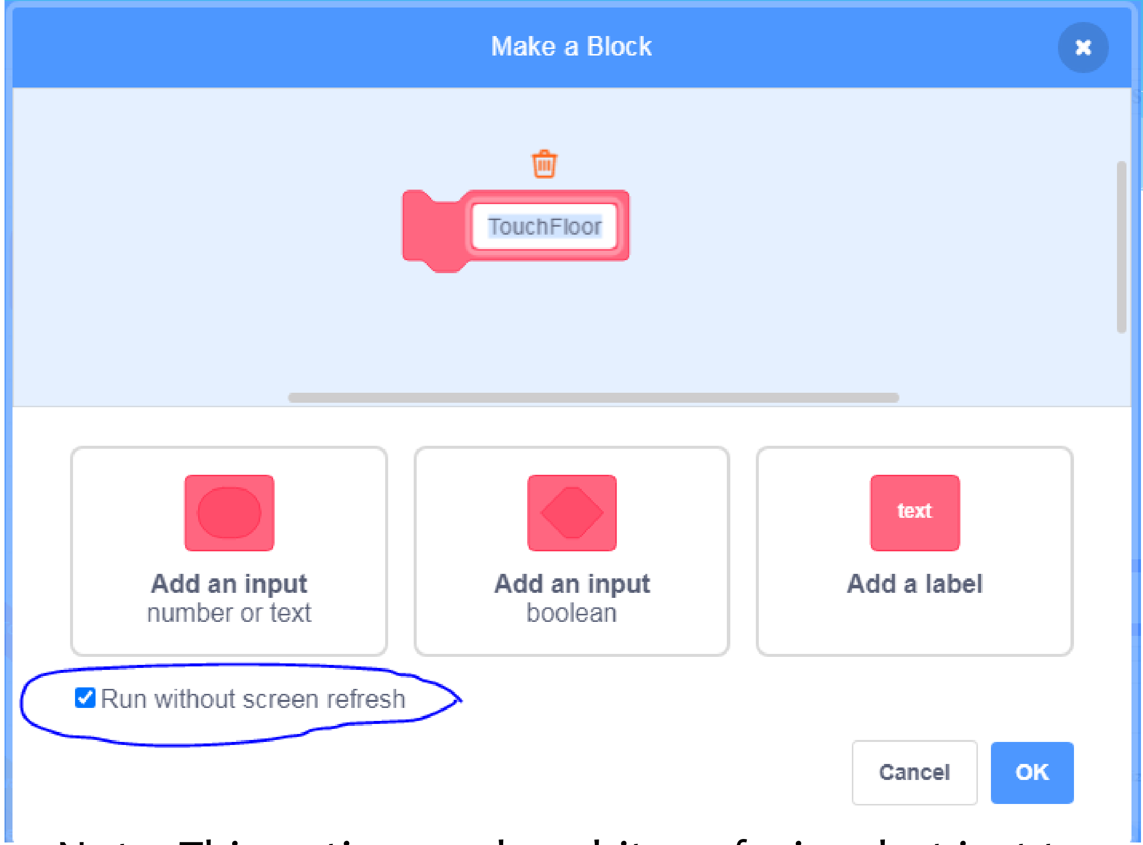
```
when [flag] clicked
set ySpeed to -5
forever
    change y by (ySpeed)
    change ySpeed by -1
    if <touching color ●?> then
        repeat until <not <touching color ●?>>
            change y by 1
            set ySpeed to 0
```

Notice, we keep pushing the sprite UP (increase its y value by 1) until it is no longer touching the platform.

But this causes a ‘bouncing’ effect

- We notice that while the code above does its job, we still see the sprite moving upwards.
- We can avoid this by using My Blocks and with the option “*RUN without screen refresh*”

Create a MyBlock


Make a Block

TouchFloor

☑ Run without screen refresh

We are not creating any inputs. Also, no labels.

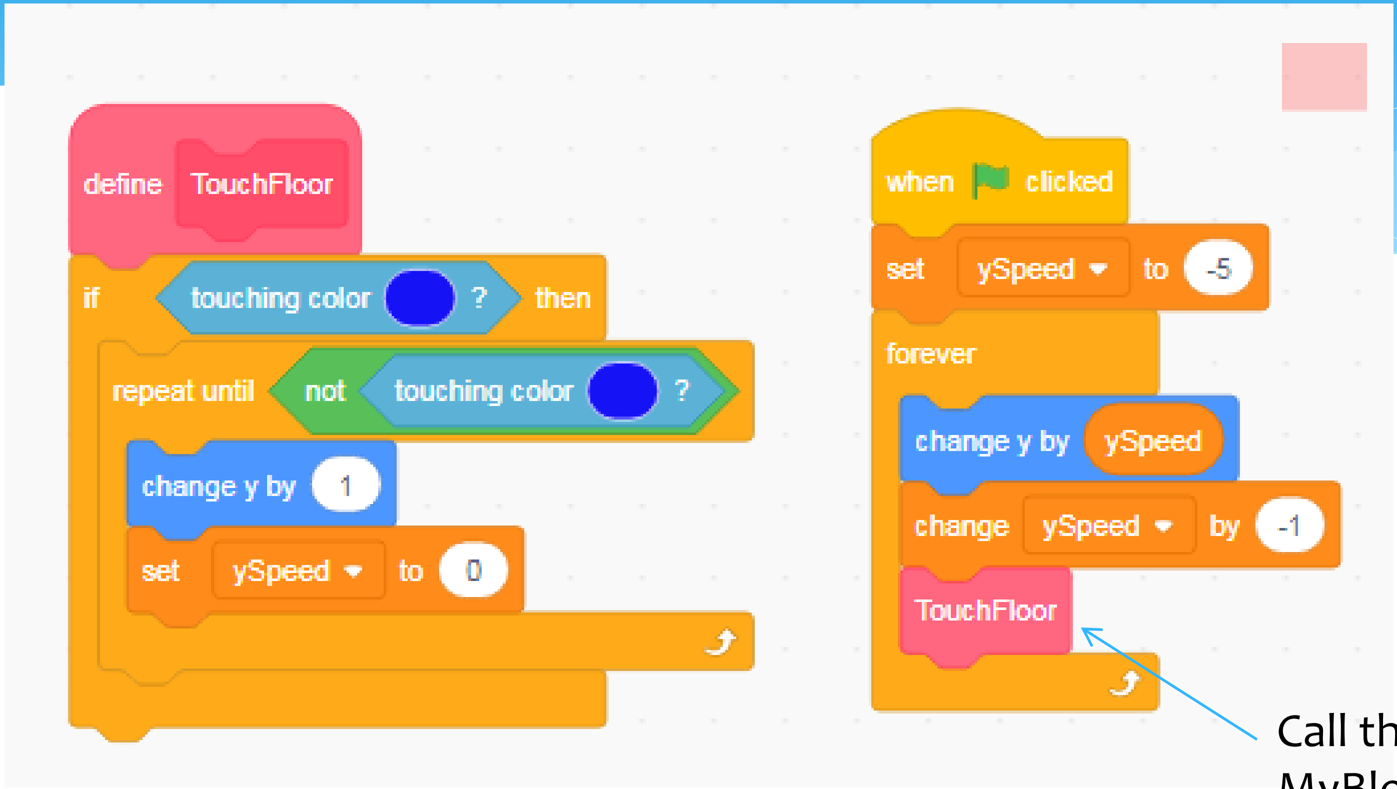
But (VERY IMPORTANTLY), we are using this option called ‘RUN WITHOUT SCREEN REFRESH’

What this does is to ‘kind of’ speed up the entire execution.

The best way to understand this is that by using this option, the Block will ‘FINISH’ its work – basically complete the REPEAT LOOP before returning the execution back.

Note: This option can be a bit confusing, but just try it. Try running the code with and without it and it will become clearer.

Define the touchFloor block



```
define TouchFloor
if <touching color (●) ?> then
    repeat until <not <touching color (●) ?>>
        change y by (1)
        set [ySpeed v] to (0)
    end
end
```

```
when 🏴 clicked
set [ySpeed v] to (-5)
forever
    change y by (ySpeed)
    change [ySpeed v] by (-1)
    TouchFloor
end
```

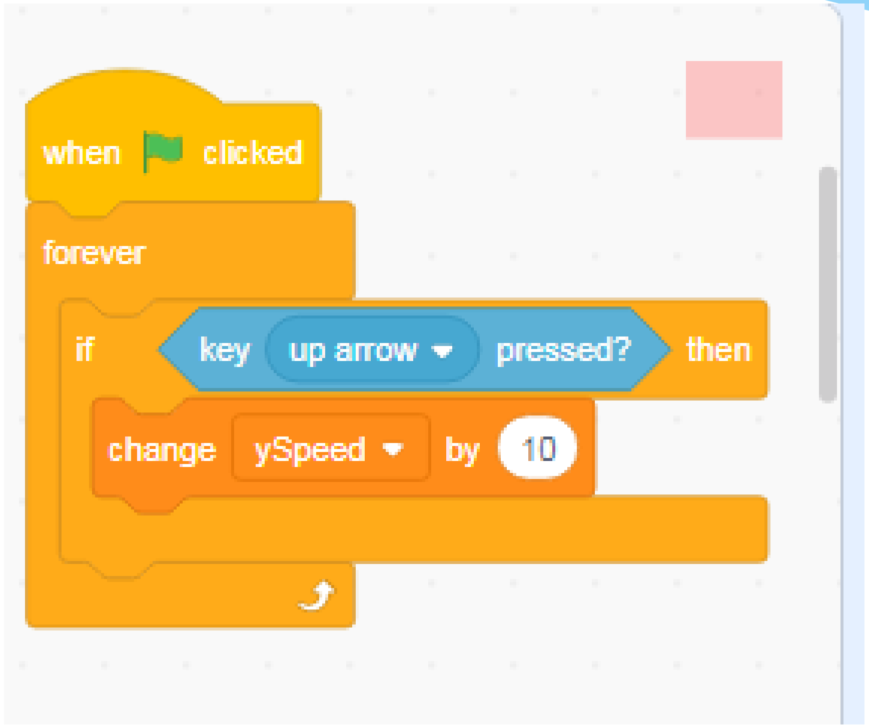
Call the MyBlock here

We basically moved the entire code into a MyBlock.
The value of the MyBlock is due to
'Run without Screen Refresh' – It basically make completes this REPEAT loop kind of one-shot! It does its job without showing us the inner steps.

Jumping

* Now that we have given our sprite the capability to respond to gravity, we will give it the capability to jump.
* Remember, the sprite naturally falls and stops on the ground.
* Thus, by jumping, we just have to change its ‘y’ position.

Jumping



```
when 🏁 clicked
forever
    if <key (up arrow ▾) pressed?> then
        change (ySpeed ▾) by (10)
```

Change ySpeed by 10 when UP ARROW is pressed?

Notice this very carefully. We are changing ySpeed and not y. (See next slide). Recall that in another loop, the y keeps changing by ySpeed and ySpeed keeps decrementing. Do you realize that this will cause the sprite to have 'natural' looking jump.

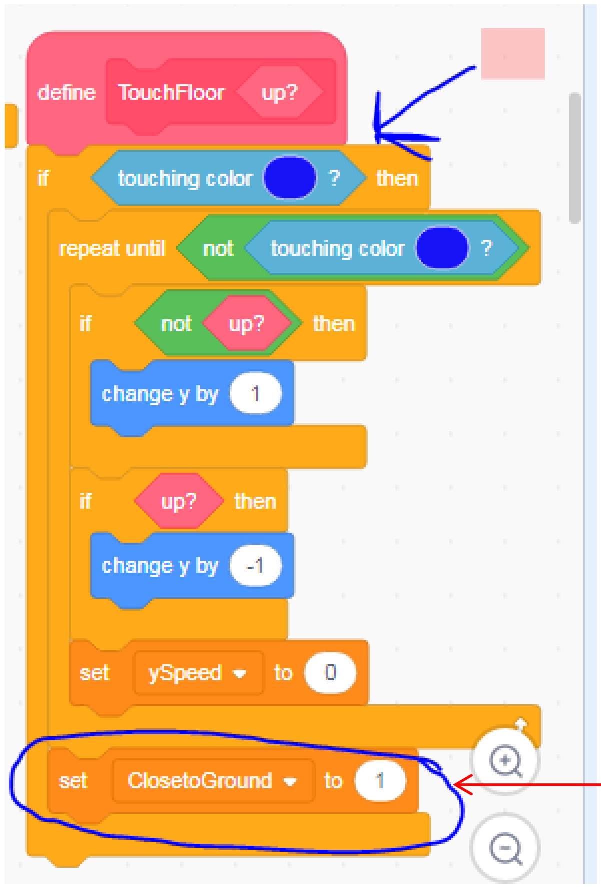
One problem in this code is that the jump is somewhat unpredictable. Every time we click the up arrow, **there may be multiple jumps created.**

Food for thought

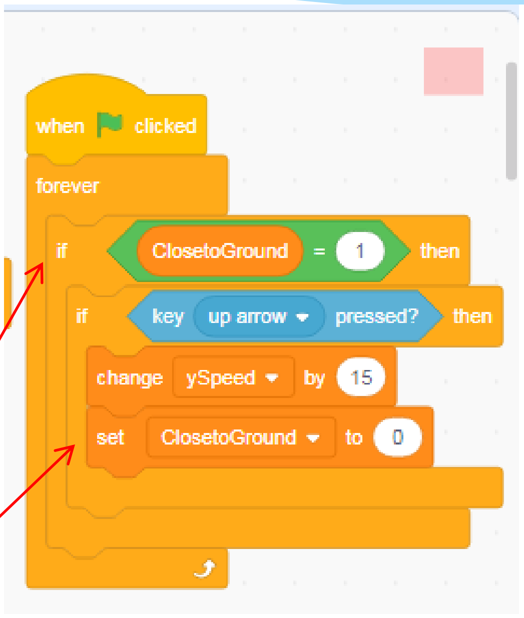
* In the previous slide, for jumping, we have changed ySpeed.
* What will happen if we changed y instead? Will that lead to a natural jump, with a symmetric looking jump and fall?

Enabling Jump only close to the ground

* We can avoid multiple jumps by allowing jumps only when the sprite is on the ground. We can use a variable for this purpose.



```
define TouchFloor (up?)
if <touching color ( ) ?> then
    repeat until <not <touching color ( ) ?>>
        if <not <up?>> then
            change y by (1)
        if <up?> then
            change y by (-1)
        set yspeed to (0)
    set ClosetoGround to (1)
```



```
when flag clicked
forever
    if <(ClosetoGround) = (1)> then
        if <key (up arrow) pressed?> then
            change ySpeed by (15)
            set ClosetoGround to (0)
```

When touchingFloor (blue colour) , set ClosetoGround to 1. As soon as the jump starts, set ClosetoGround to 0. And start the jump ONLY when ClosetoGround is 1.

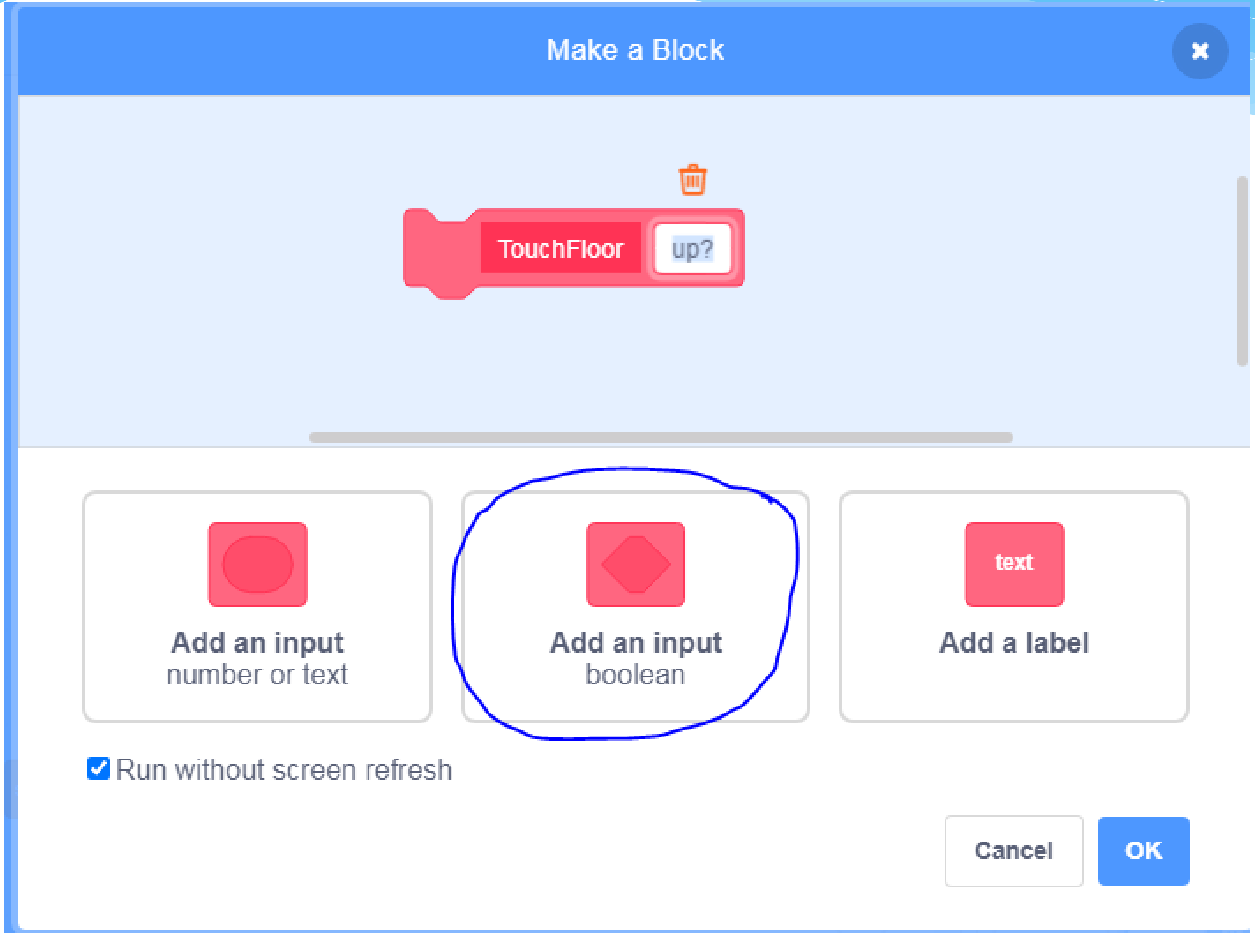
Food for thought

* A variation of the above idea is to enable jump not 'exactly' on the ground, but close to the ground too. This is important in case the sprite is coming down on a ramp.
* Can you think of a modification to the earlier code to do that?
* HINT – What if we keep track of 'how far' from the floor we are, rather than just tracking if we are on the floor or not.

Detecting Ceiling

- Notice that if we hit the platform from below also, the sprite just 'goes through'.
- This is because the 'touch floor' code – what we used to prevent the sprite falling all the way – tends to push the sprite upwards. (It assumes that the sprite is falling from top).
- To prevent this, we can modify the 'touchFloor' block, and give it one more input.

Add a BOOLEAN input to the TouchFloor

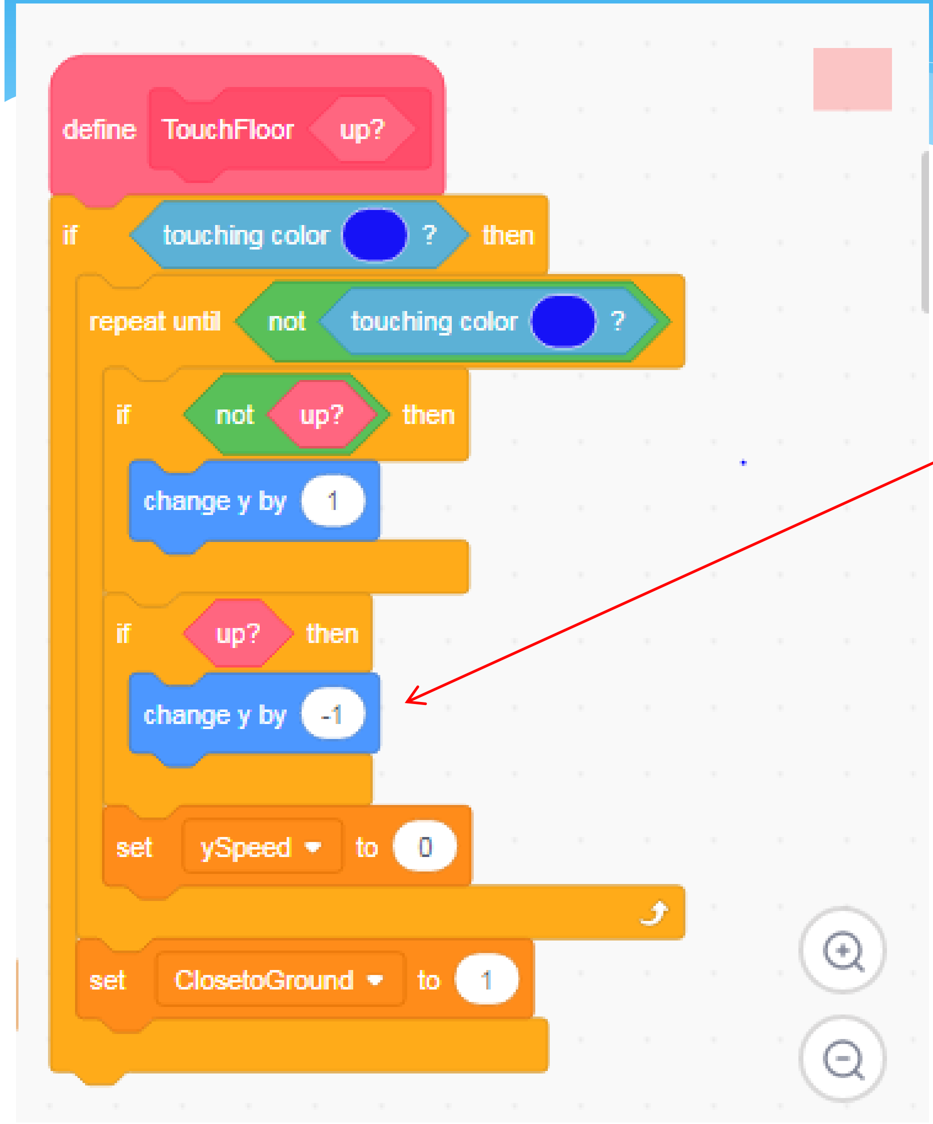


☑ Run without screen refresh

BOOLEANs are special inputs, which can take on ONLY two values – TRUE or FALSE.

We will give an input to the touchfloor, to determine if the sprite is going upwards or downwards.

Updated Definition of TouchFloor

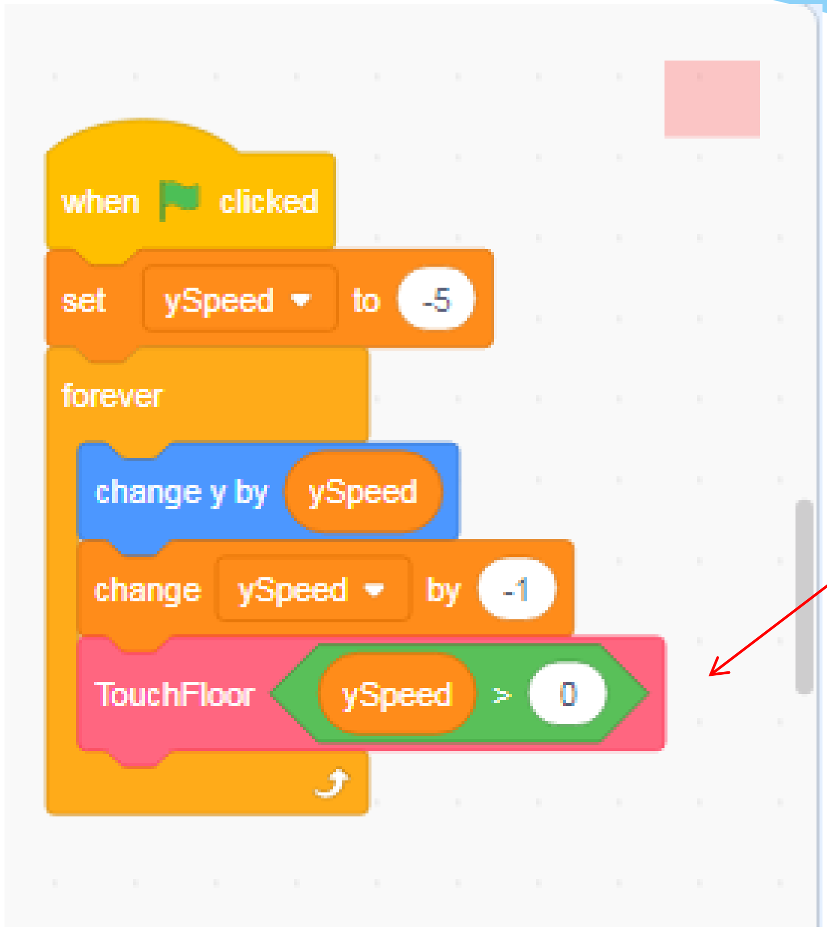


```
define TouchFloor (up?)
if <touching color (●) ?> then
    repeat until <not <touching color (●) ?>>
        if <not <up?>> then
            change y by (1)
        if <up?> then
            change y by (-1)
        set [ySpeed] to (0)
    set [ClosetoGround] to (1)
```

If the sprite is traveling 'upwards', pull it down until it is no longer touching the floor.

Think this through, this helps us distinguishing a falling object from a rising object!

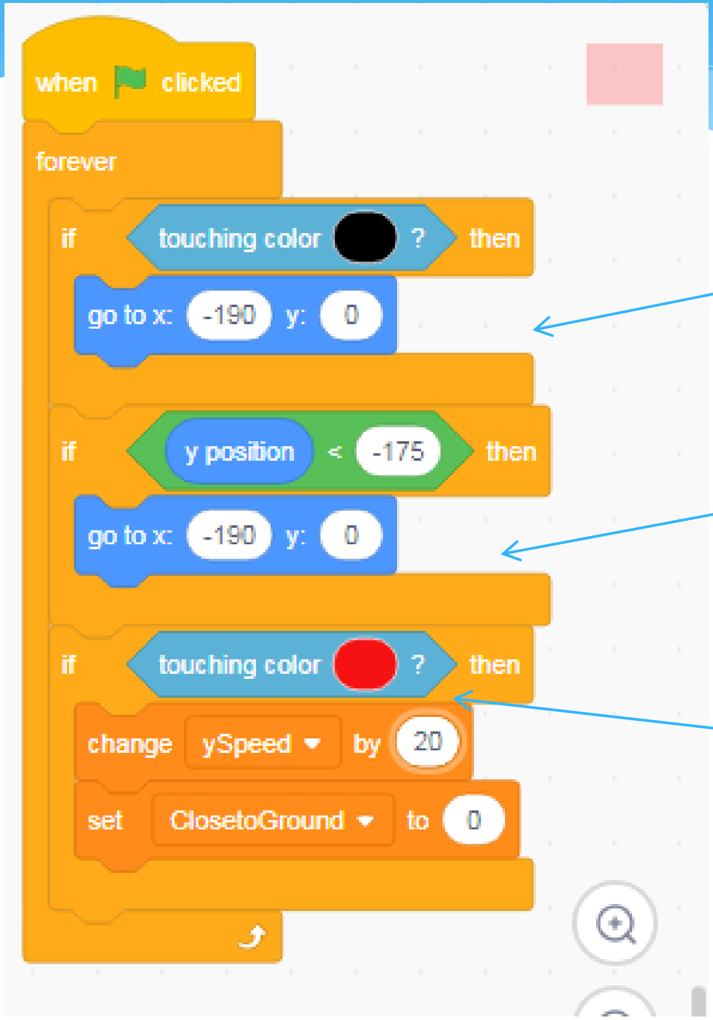
Updated Calling of TouchFloor



```
when [flag] clicked
set ySpeed to -5
forever
    change y by (ySpeed)
    change ySpeed by -1
    TouchFloor <(ySpeed) > 0>
```

We pass an input to the TouchFloor which is 1 when the ySpeed is more than 0, which means that the sprite is moving upwards!

What happens when we hit obstacles



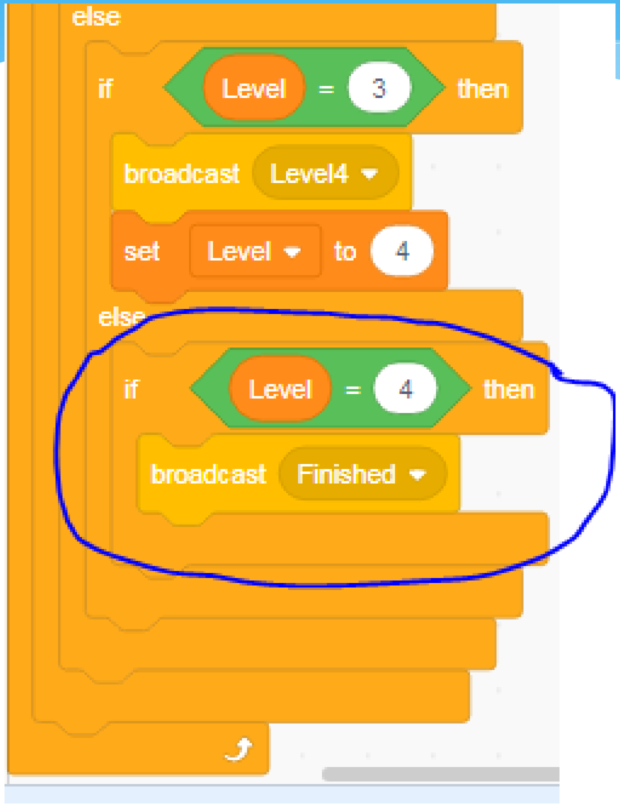
```
when [flag] clicked
forever
    if touching color ● ? then
        go to x: -190 y: 0
    if y position < -175 then
        go to x: -190 y: 0
    if touching color ● ? then
        change ySpeed by 20
        set CloseToGround to 0
```

Touching BLACK sends us back to the LEFT corner

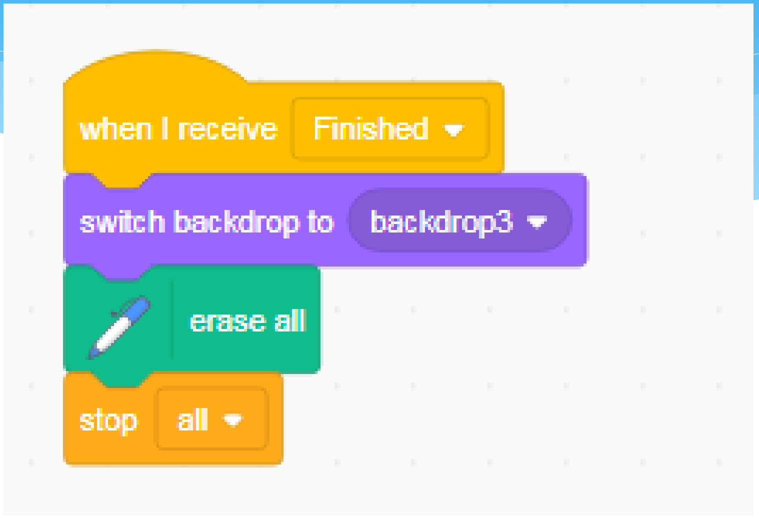
Falling down sends the sprite back to the LEFT corner

Stepping on RED colour makes the sprite jump

Ending the game



```
else
  if <(Level) = (3)> then
    broadcast (Level4 ▾)
    set [Level ▾] to (4)
  else
    if <(Level) = (4)> then
      broadcast (Finished ▾)
```



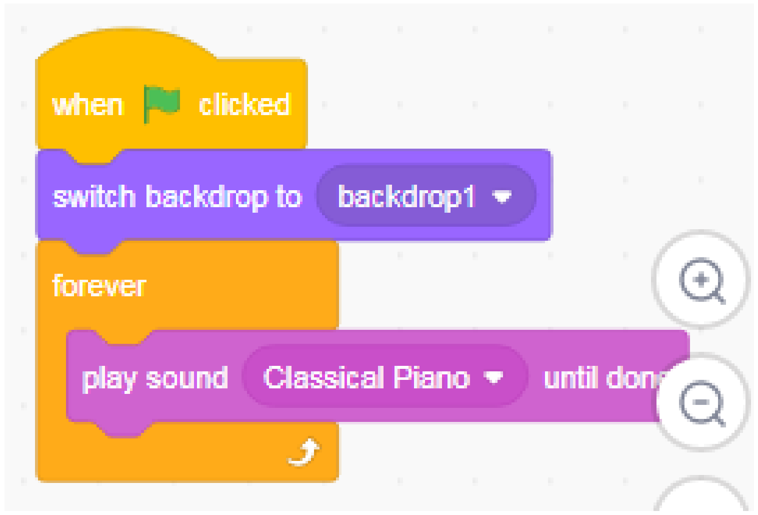
```
when I receive [Finished ▾]
switch backdrop to (backdrop3 ▾)
erase all
stop [all ▾]
```

Add ONE more condition to the LEVELS loop. After the last LEVEL, the game gets over

WIBYTE

Add some background Music!

* Play music in a separate forever loop.



```
when 🏴 clicked
switch backdrop to backdrop1 ▾
forever
    play sound Classical Piano ▾ until don
```

WIBYTE

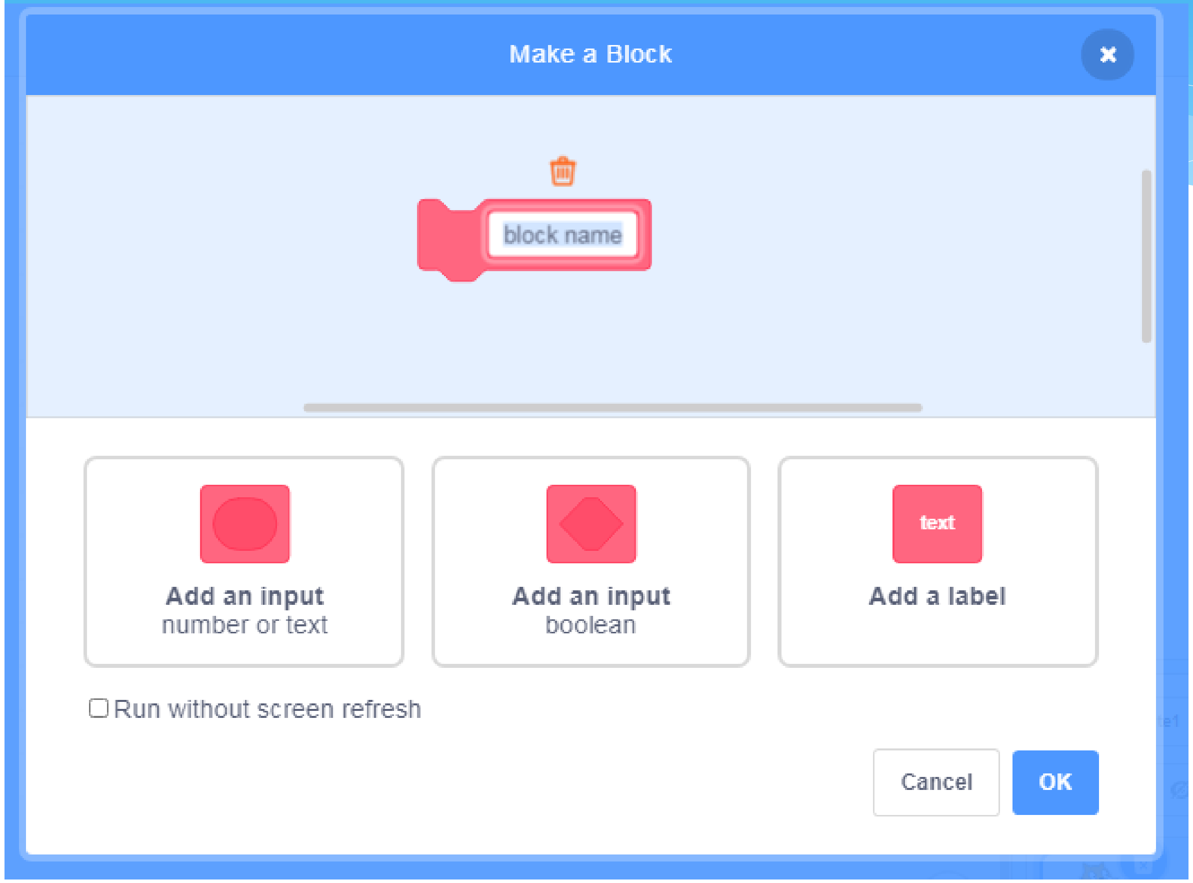
And you are all set!

* With this, you are all set for your Independent Activity -16 : Pen Platformer Part 2.
* In this activity, you will use concepts of gravity, jumping, ceiling detection to give life to your own Pen Platformer.
* Use this opportunity to learn more about and gain confidence with MyBlocks! Of course, in the process, have fun ☺.

WIBYTE

Extra Innings

Running without screen refresh



☐ Run without screen refresh

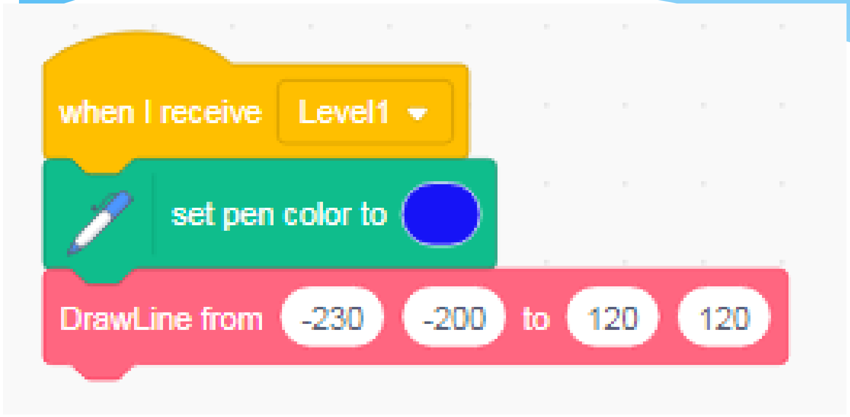
Recall, there is an option called 'RUN WITHOUT SCREEN REFRESH'.

This is useful if there are loops like 'repeat' inside your block. With this option, the loops run faster – without the added delay caused by screen getting refreshed.

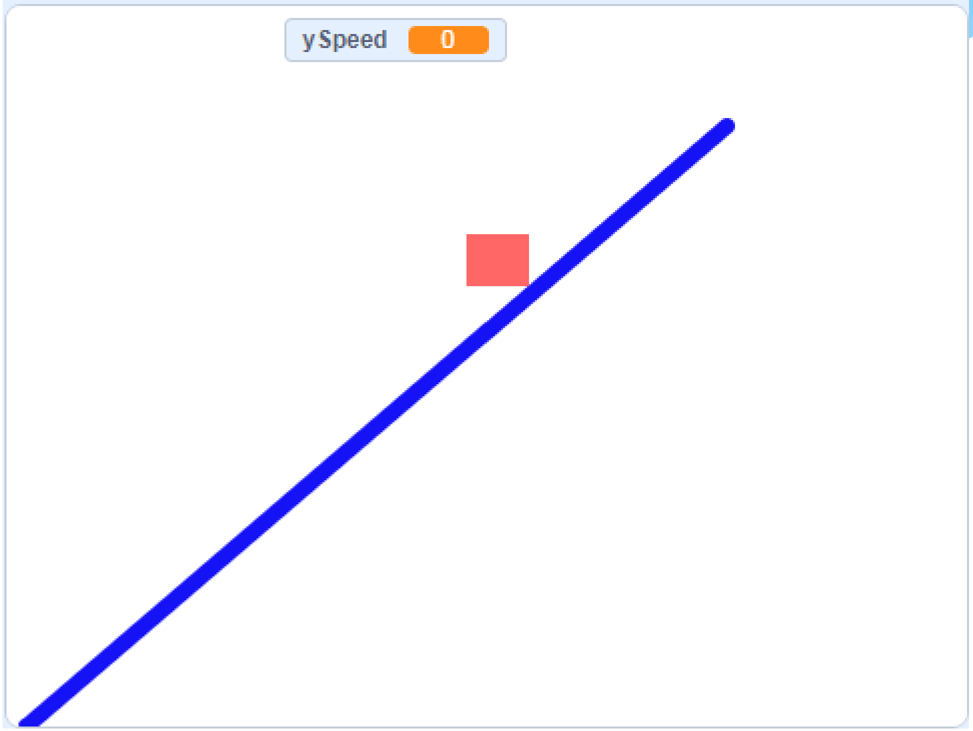
Hence, if you are making a circle, using repeat block, this may be helpful.

See these examples:
https://scratch.mit.edu/projects/88005579/editor
https://scratch.mit.edu/projects/10567475/editor

Notice something interesting – Sprite going up an incline



```
when I receive Level1
set pen color to ●
DrawLine from -230 -200 to 120 120
```



The touchFloor function will also enable the Sprite to move up on an incline. Try this.

With the code we have done previously, we will see that the sprite will climb up with it!

WIBYTE

Preventing the sprite from going up on a wall

* We have seen that the sprite can go up on an incline.
* But what if the incline is very very steep.
* We can actually stop the sprite going up the steep line – but it requires more coding.
* Basically we have to determine the ‘slope’.
* Again a usage of MyBlocks.

Ideas to spice up the game!

- This game can be enhanced in many ways.
- For a start, we can add LIVES, and score.
- Then, we can have interesting arenas and many levels.
- We can define a few types of jumps.
- We can have different types of obstacles.
- We can have some ‘boosters’ that make the sprite immediately go to the next level etc.
- Use your imagination! But this project is coding intensive – so attempt the bonus activities only after you are through with the basic stuff.

WIBYTE