Sprite breaks the bricks …

Vineet Srivastava

In this lesson, we will …

* Explore the concept of direction in more detail.
* Create 'reflection'.
* Create a version of the popular 'BRICK BREAKING' game.

WIBYTE

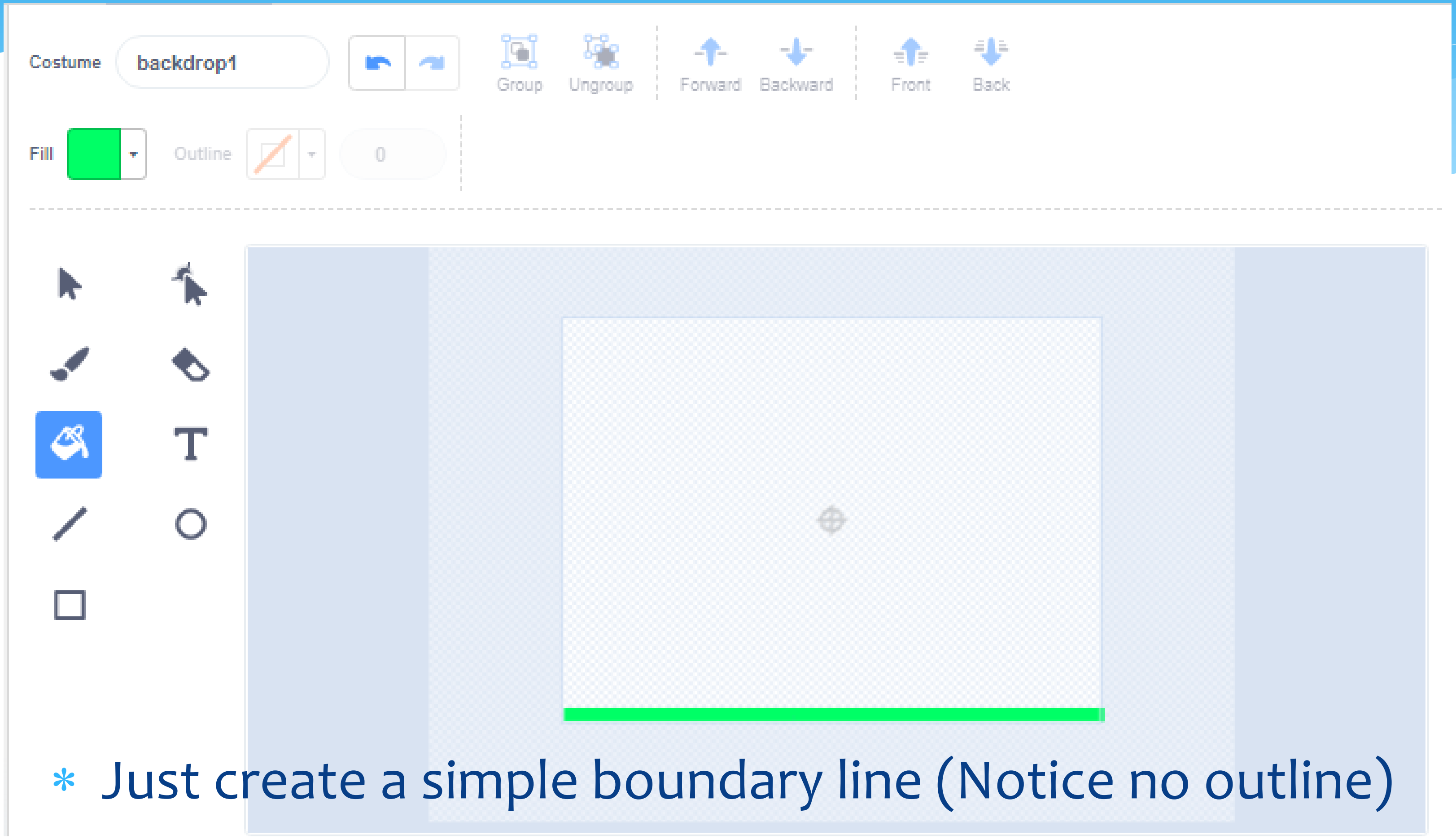
What is the brick breaking game?

- A ball must break an array of ‘bricks’.
- The player has a pedal to bounce the ball on.
- The player must prevent the ball from falling down.
- Every time the ball hits a brick, the brick disappears.

The concept of direction

- During our beginner session, we said that direction can be somewhat confusing in case of scratch.
- However, it is a very important concept that we have to understand to make SPRITES move in somewhat advanced ways.

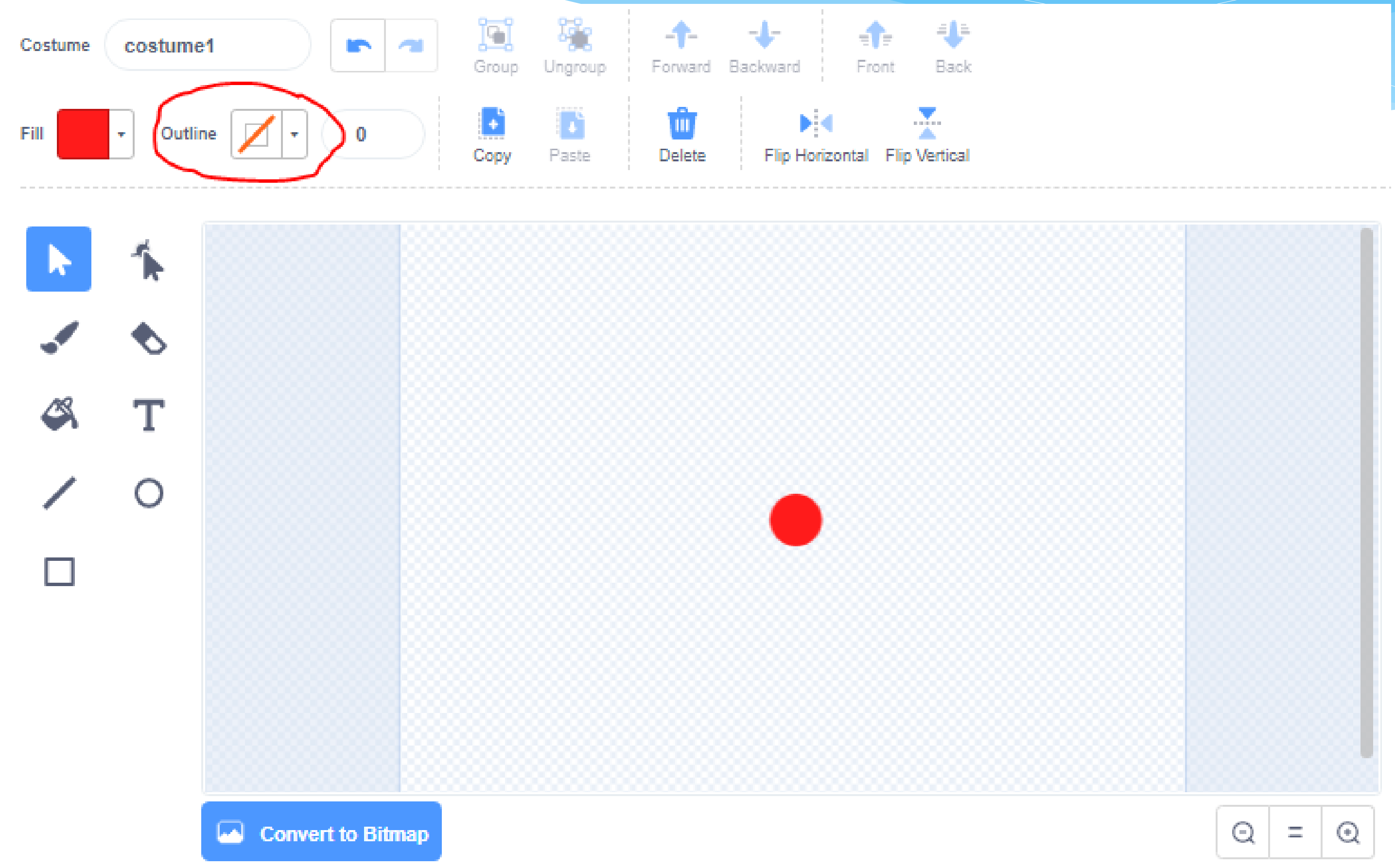
Create the backdrop!



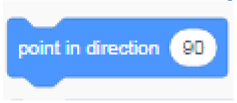
* Just create a simple boundary line (Notice no outline)

WIBYTE

Let's bring in the BALL sprite

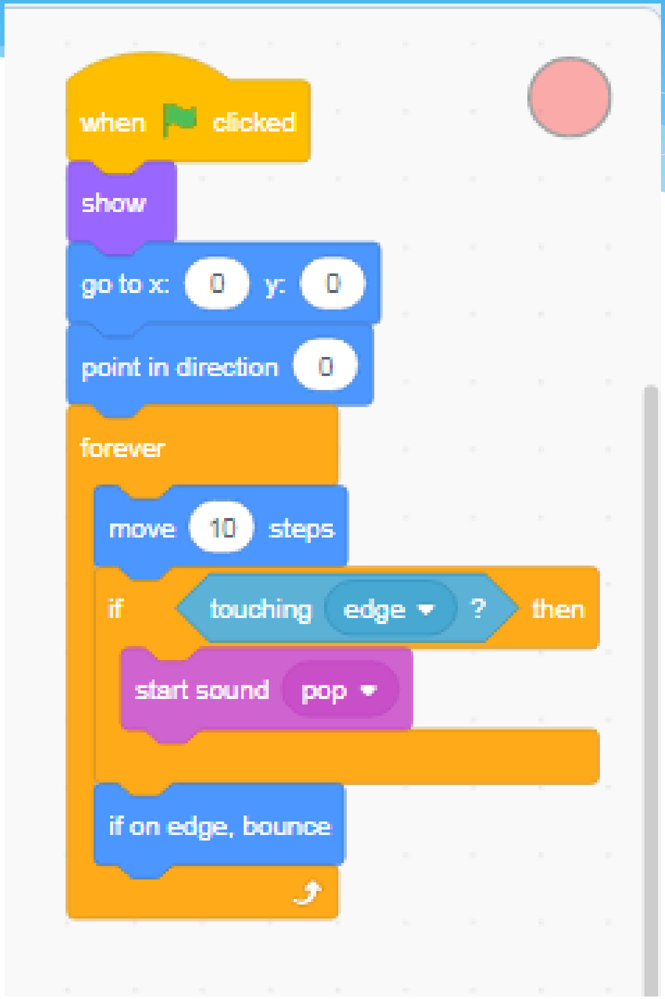


Point in direction Block

- In the MOTION section, there is a block called POINT in DIRECTION, 

- We can change the ‘value’ to control which direction the sprite points in.
- Direction is measured in ‘degrees’, which is itself a mathematics concept that students learn in secondary school. Fortunately scratch gives us a simple, visual way, to set direction.

Making ball sprite bounce off the edges



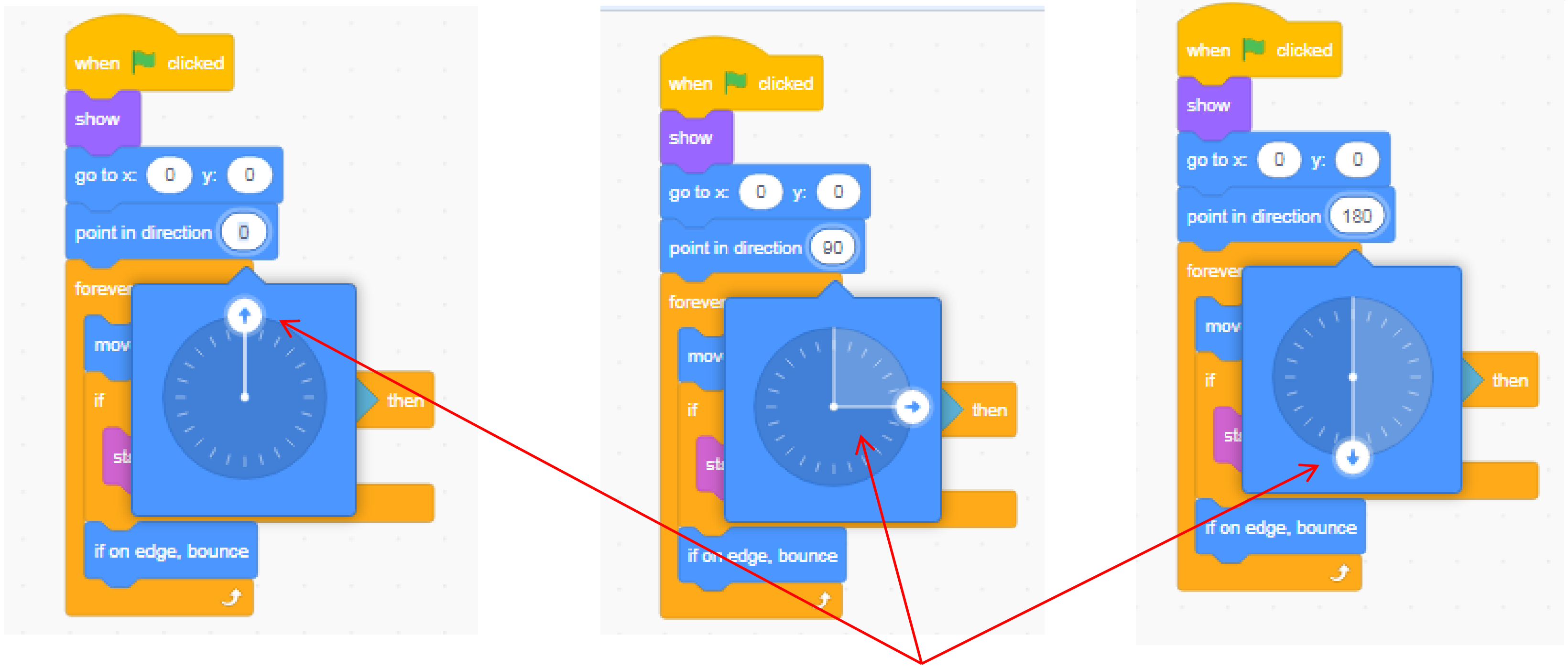
```
when [flag] clicked
show
go to x: 0 y: 0
point in direction 0
forever
    move 10 steps
    if touching edge ? then
        start sound pop
    if on edge, bounce
```

If the sprite touches the 'EDGE', a small sound clip is played and the sprite bounces back.

WIBYTE

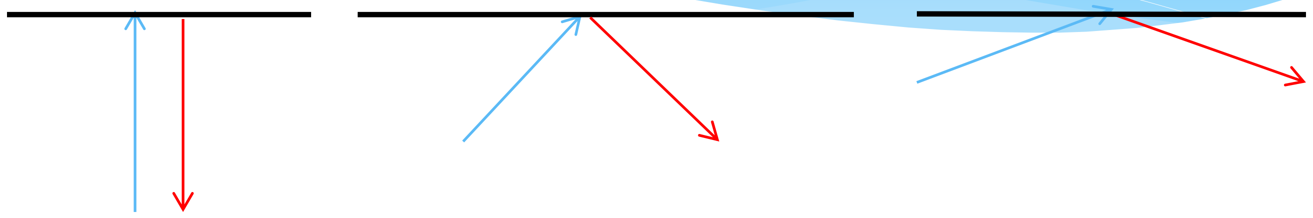
Sprite’s Direction

* In simple words, upward direction is 0, right direction is 90 and downward direction is 180.



Do NOT get scared or confused by the numbers, just use the arrow on the dial and **experiment for yourself**.

Let’s study sprite’s direction carefully (before and after bounce)



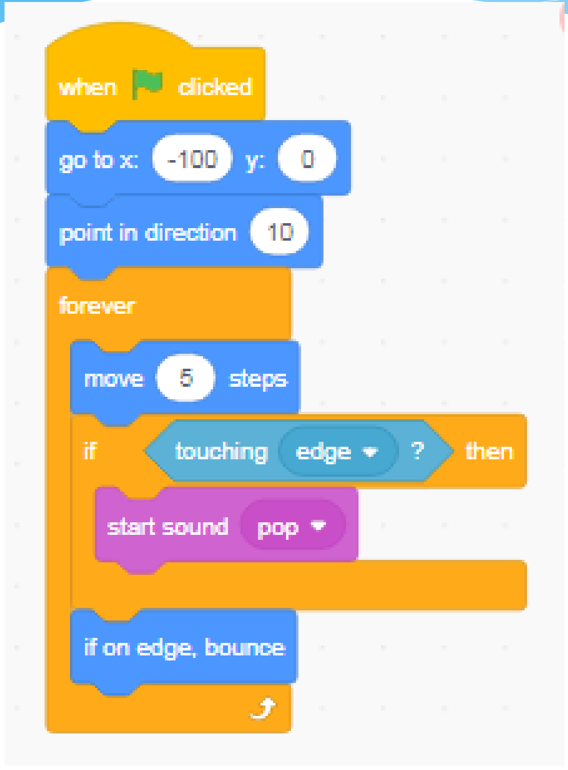
Direction Before Bounce	0	45	15
Direction After Bounce	180	135	165

Case 1

Do NOT get scared or confused by the numbers, just use the arrow on the dial.

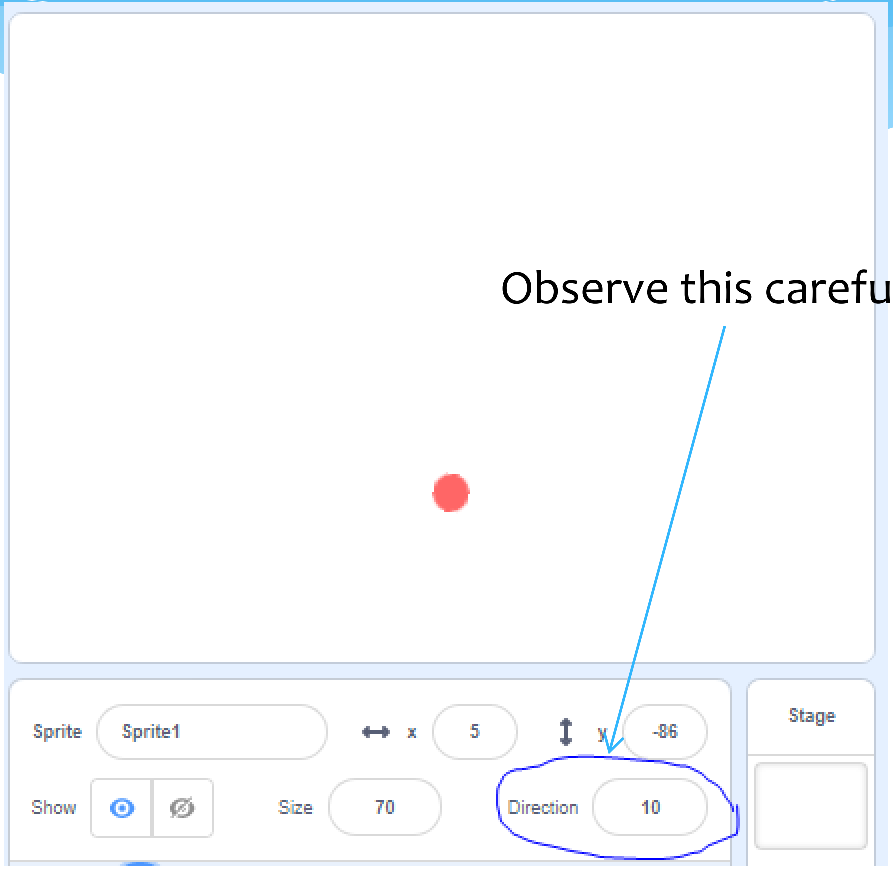
WIBYTE

How to observe this?



```
when 🏴 clicked
go to x: -100 y: 0
point in direction 10
forever
    move 5 steps
    if touching edge ▾ ? then
        start sound pop ▾
    if on edge, bounce
```

Run this code, this basically makes the ball bounce off the edges.



Observe this carefully

Sprite Sprite1 ↔ x 5 ↕ y -86

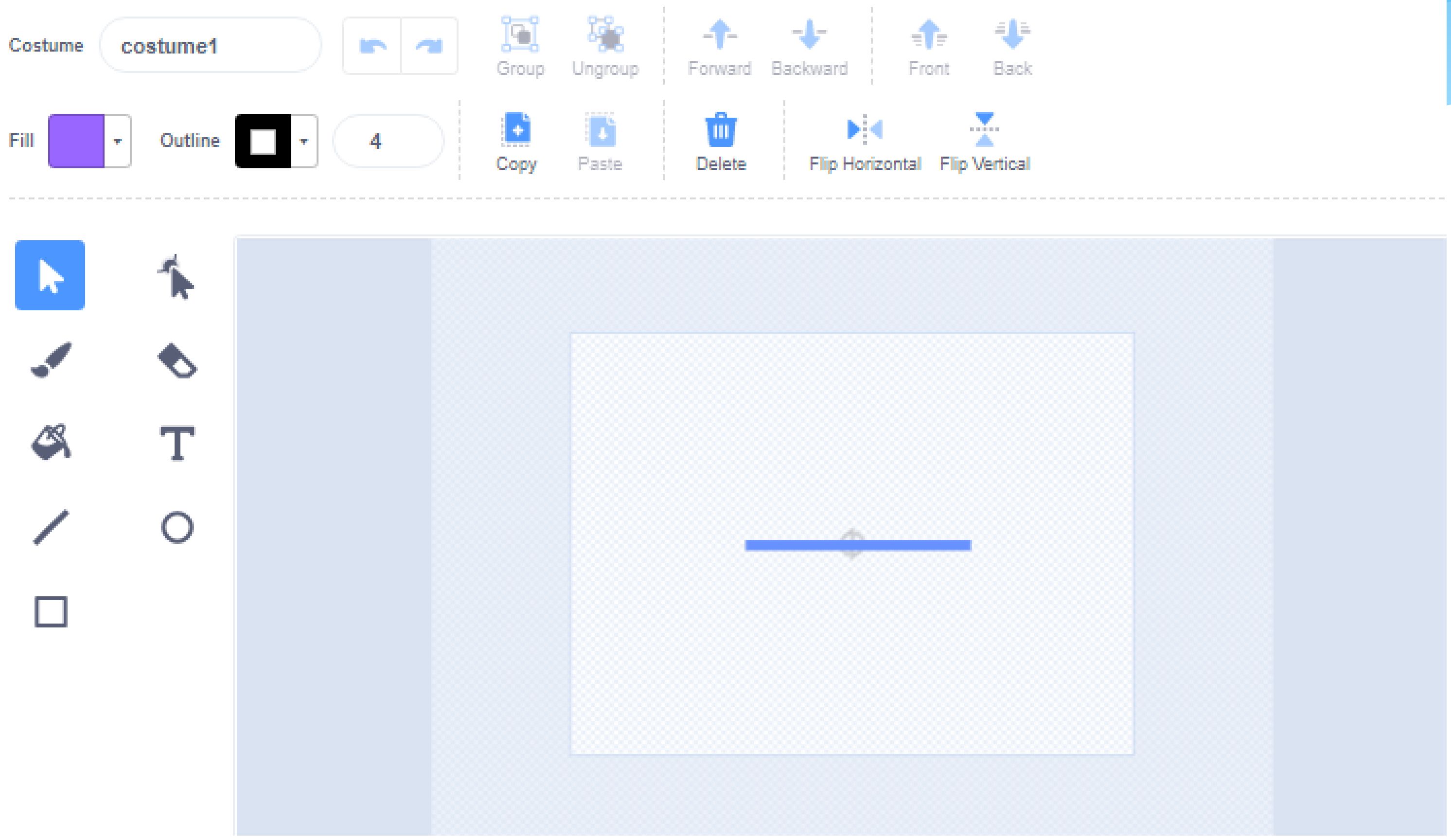
Show Size 70 Direction 10

Ok, so what?

- The point is, ‘we can use this principle to make sprite bounce of ANY other sprites. Let’s see how.
- All we have to do is to change its direction from the present direction to **<180 – direction>** using the point in direction block.

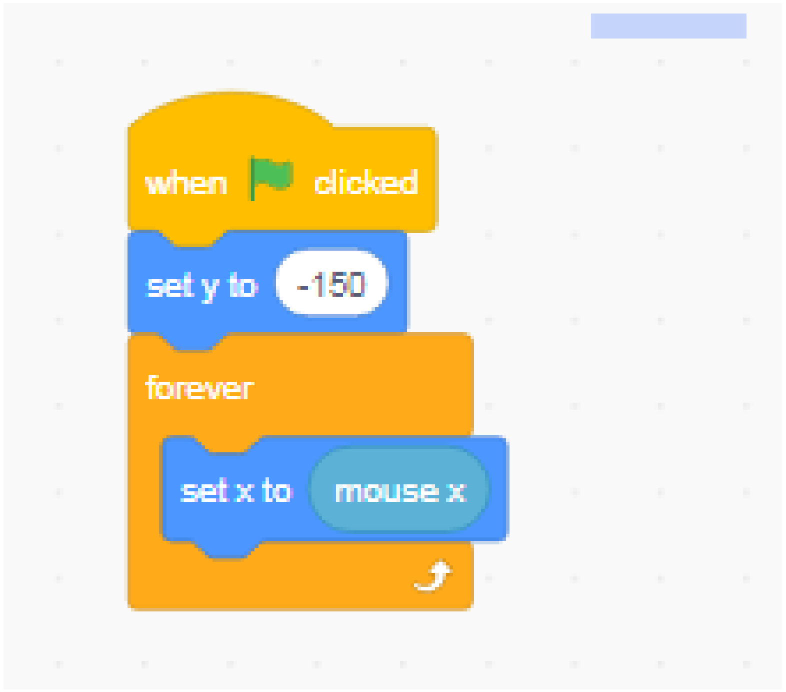
- NOTE:
  - What we covered here is a well-known principle of science called LAW OF REFLECTION. You will learn this in science.
  - You can prove this using geometry – as you will see in mathematics.
  - This applies to reflection off HORIZONTAL surfaces. You can work out the formulas for reflection off vertical or ‘tilted’ surfaces.

Let's bring in the PEDAL sprite



WIBYTE

Pedal’s code for movement



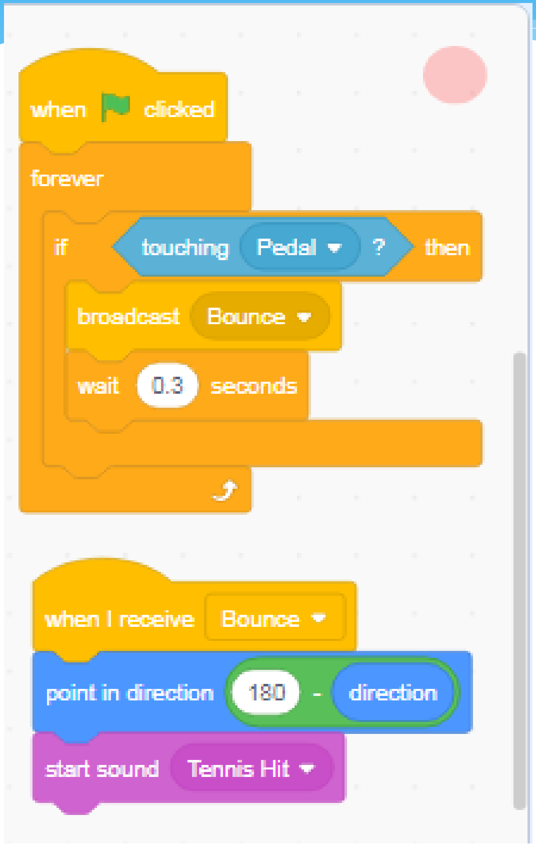
```
when [flag] clicked
set y to -150
forever
    set x to (mouse x)
```

Send the plate sprite somewhere in the lower part of the stage (hence y = -150)

Make it move horizontally with the mouse position. (Exactly same as what we did in the catch game)

WIBYTE

Making Ball bounce off the pedal



```
when [flag] clicked
forever
    if <touching [Pedal ▾] ?> then
        broadcast [Bounce ▾]
        wait (0.3) seconds

when I receive [Bounce ▾]
point in direction ((180) - (direction))
start sound [Tennis Hit ▾]
```

When the ball touches the plate, change the direction of the ball to
(180 – direction).

Example:

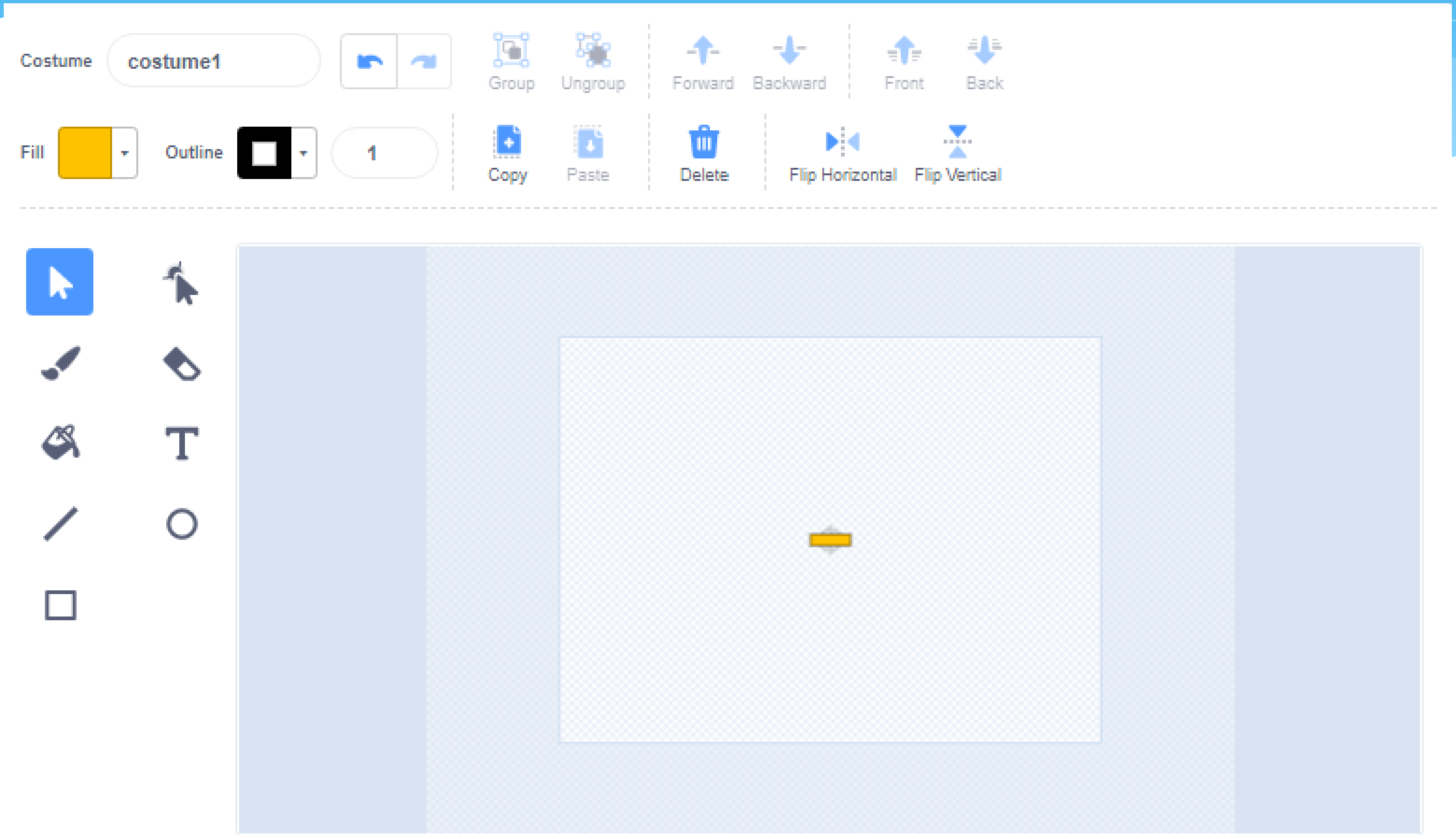
Say ball was traveling in direction 150. Once it touches the plate, it will point in direction 180-150 = 30. And so on.

(Observe the 'direction' of the ball before and after it hits the plate).

NOTE: A lot is going on here. Do NOT rush through this. Go very slow, try to understand each piece individually. Why we used Broadcast will get more clear later

WIBYTE

Let's create the bricks



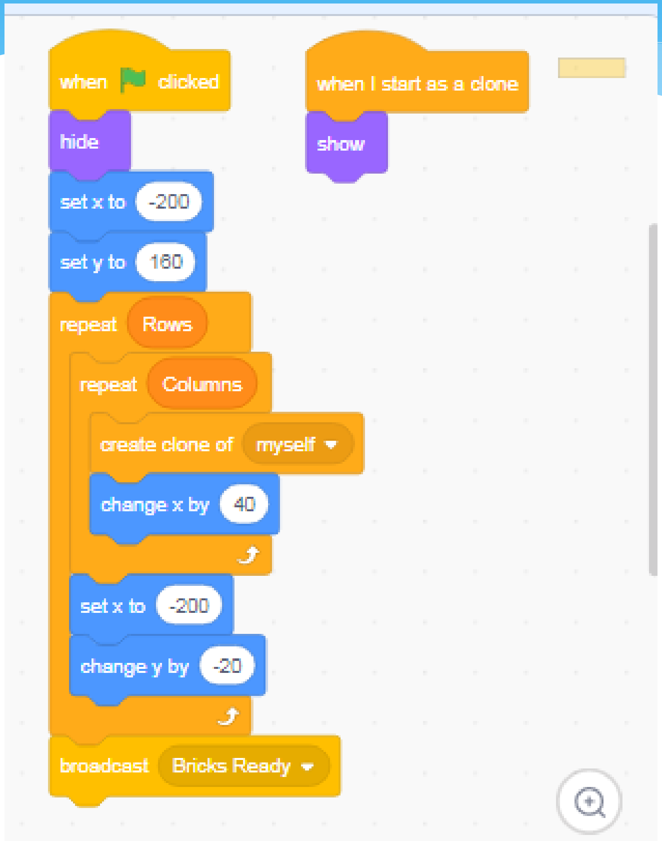
Create a new sprite – as before, ensure the costume is centred correctly.

WIBYTE

Placing the bricks in a grid

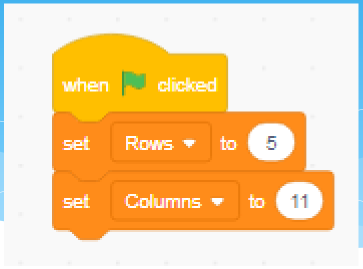
- Remember, from our game, we have to create multiple bricks. We will place these sprites in a grid.
- We will use the same method as we used for the pits in the 'Whack-a-mole' game.

Placing the bricks



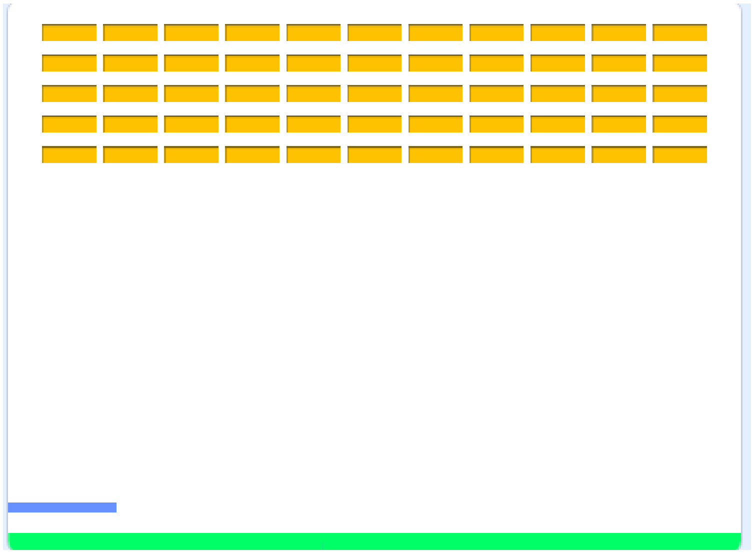
```
when [flag] clicked
hide
set x to -200
set y to 160
repeat Rows
    repeat Columns
        create clone of myself
        change x by 40
    set x to -200
    change y by -20
broadcast Bricks Ready

when I start as a clone
show
```



```
when [flag] clicked
set Rows to 5
set Columns to 11
```

Notice that this logic is exactly the same as the 'Whack-a-mole' game.

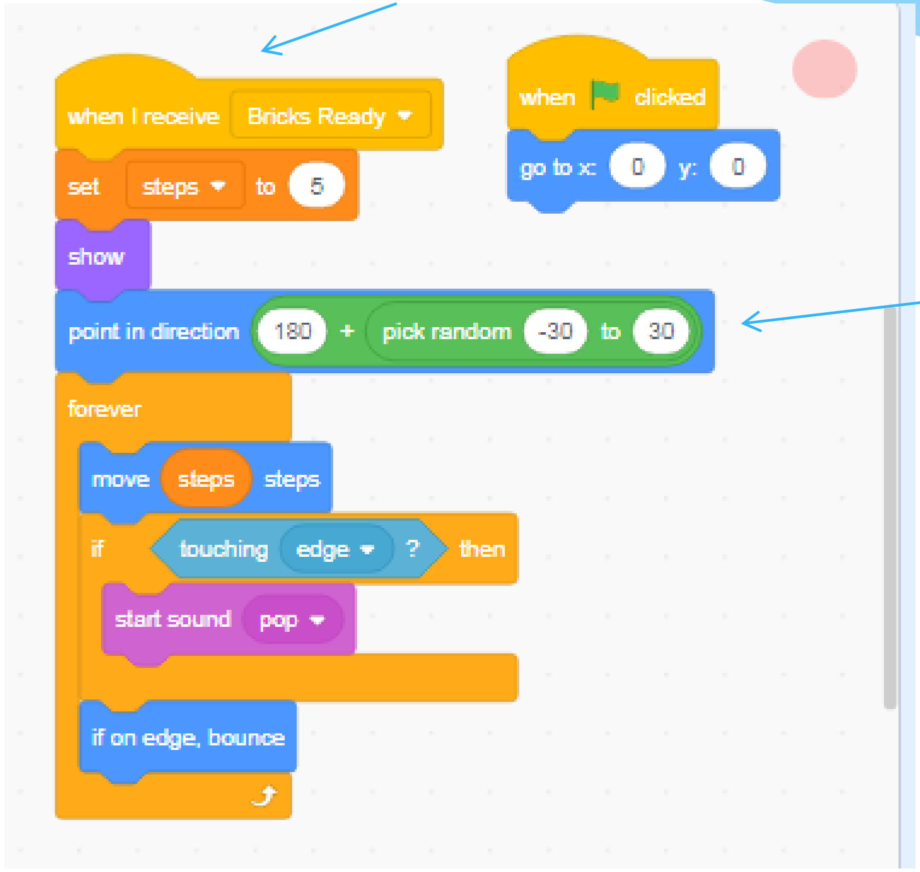


We placed a 5 Row, 11 Column grid.

WIBYTE

Getting the game started

Receiving Broadcast Message



```
when I receive [Bricks Ready ▾]
set [steps ▾] to (5)
show
point in direction ((180) + (pick random (-30) to (30)))
forever
    move (steps) steps
    if <touching [edge ▾] ?> then
        start sound [pop ▾]
    end
    if on edge, bounce
end

when ⚑ clicked
go to x: (0) y: (0)
```

Notice this:

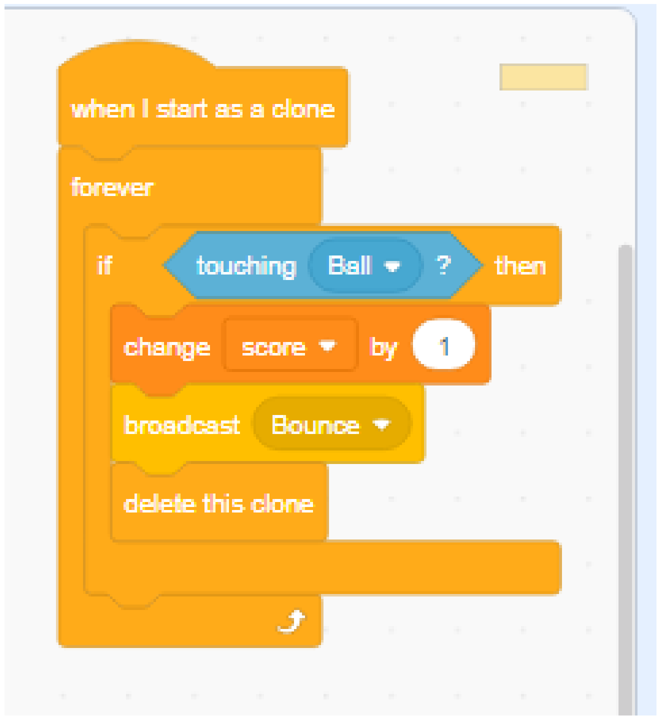
This will create a random direction between 150 and 210.

(That is pointing downwards, so that the pedal can be alert.)

WIBYTE

Keeping track of the score

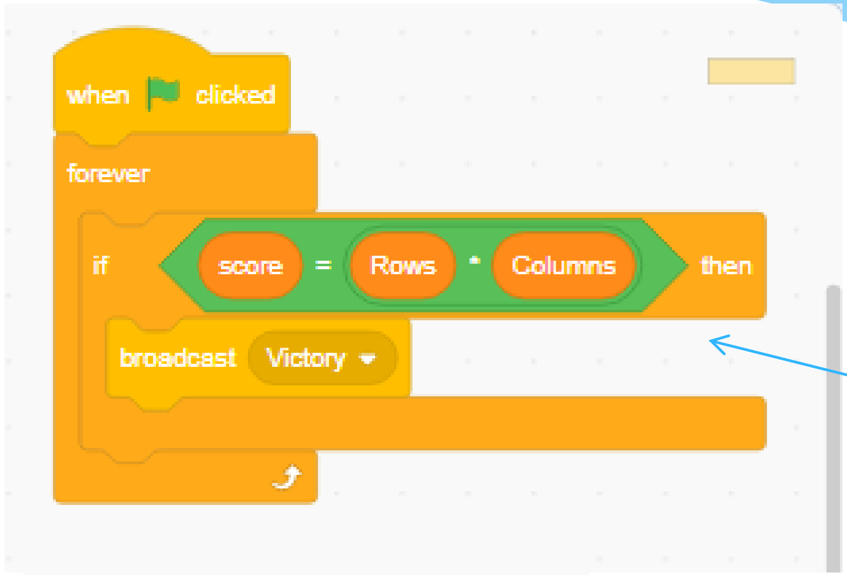
- The score is basically just the number of bricks that have been broken.



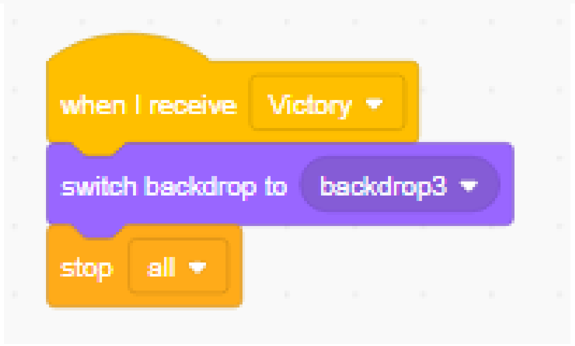
```
when I start as a clone
forever
    if touching Ball ? then
        change score by 1
        broadcast Bounce
        delete this clone
```

WIBYTE

Getting the game to end (Victory)



```
when [flag] clicked
forever
    if <(score) = ((Rows) * (Columns))> then
        broadcast [Victory ▾]
```



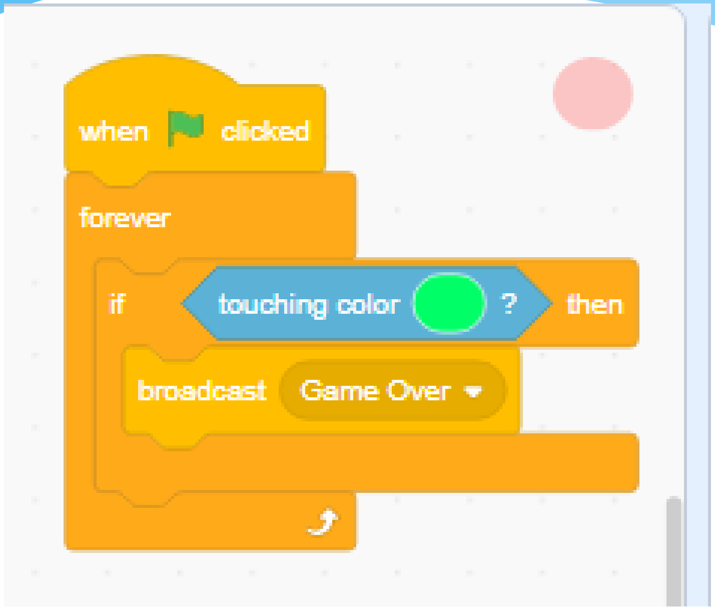
```
when I receive [Victory ▾]
switch backdrop to [backdrop3 ▾]
stop [all ▾]
```

Once you manage to clear all the bricks, you win!

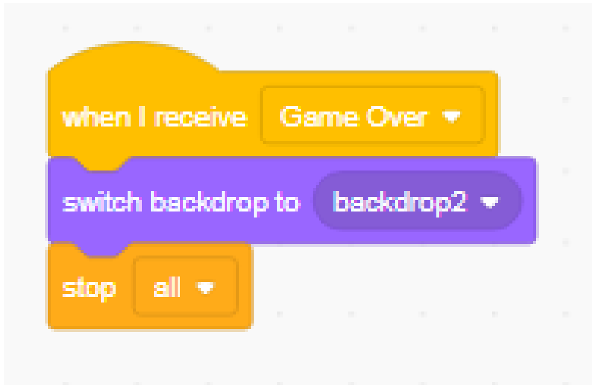
Notice the use of operators!

WIBYTE

Getting the game to end (Defeat)



```
when [flag] clicked
forever
    if <touching color [green] ?> then
        broadcast [Game Over]
```



```
when I receive [Game Over]
switch backdrop to [backdrop2]
stop [all]
```

Notice, strictly speaking the game starts after the bricks are ready. But since we force the 'BALL' to go to (0 0) at that point, we are sure that the ball is not touching the green line when the flag is clicked. Hence we can use WHEN FLAG CLICKED here.

You are all set!

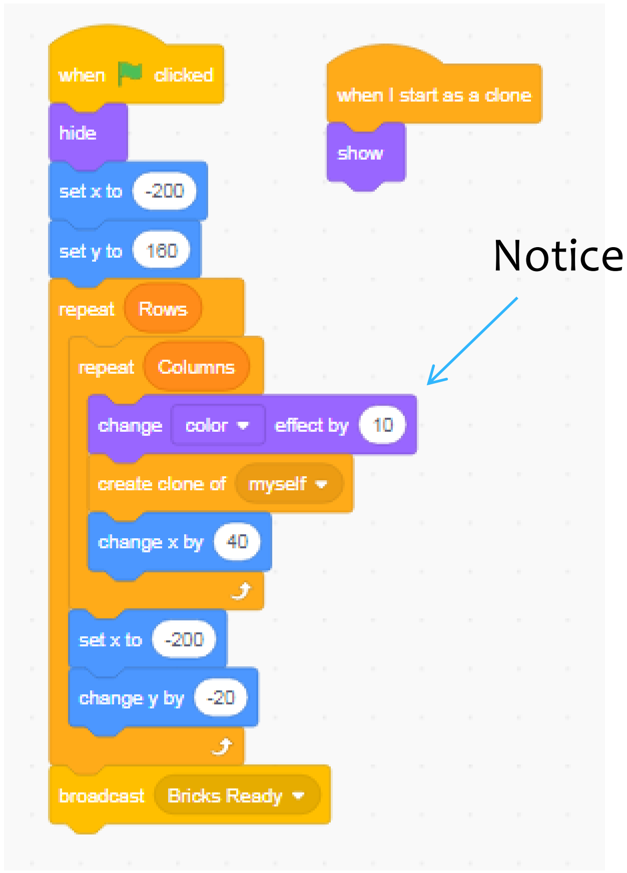
* The key ‘new’ concept we learnt in the class is of direction.
* This is a very useful concept but needs practice to understand.
* Try out this activity. Apart from concept of direction, this will also be a good practice of the concepts of cloning/broadcasting etc.

Extra Innings

Ideas to spice up the game!

- Attractive graphical effects
  - For example, before the cloning, we can use color effects to get colourful bricks.
- Increase the speed of the ball as the game progresses.
  - However, be careful. If the speed is too high the game will become unplayable and also the bounce won't work properly.
- CRAZY Ball
  - Change the bounce code o that the ball does not follow the laws of reflection. Instead, it reflects in a random direction.
- Bricks appear in formations
- Bricks give different points (This will be better done without cloning).
- Two or more type of ball sprites (Use different sprites for balls – do not clone the balls).
- Some obstacles.
- Attacking bricks.

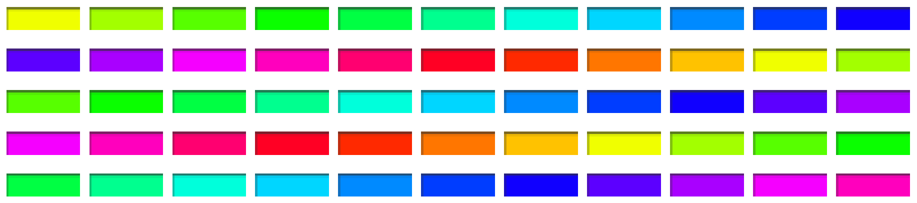
Color effects before cloning



```
when [flag] clicked
hide
set x to -200
set y to 160
repeat (Rows)
    repeat (Columns)
        change [color v] effect by 10
        create clone of [myself v]
        change x by 40
    end
    set x to -200
    change y by -20
end
broadcast [Bricks Ready v]

when I start as a clone
show
```

Notice



WIBYTE