

# Sprite wants to play catch ...

Vineet Srivastava

# In this lesson, we will learn ...

- \* How to create a version of the popular 'catch game'.
- \* In doing this, we will take a closer look at the concept and usage of variables in making games.

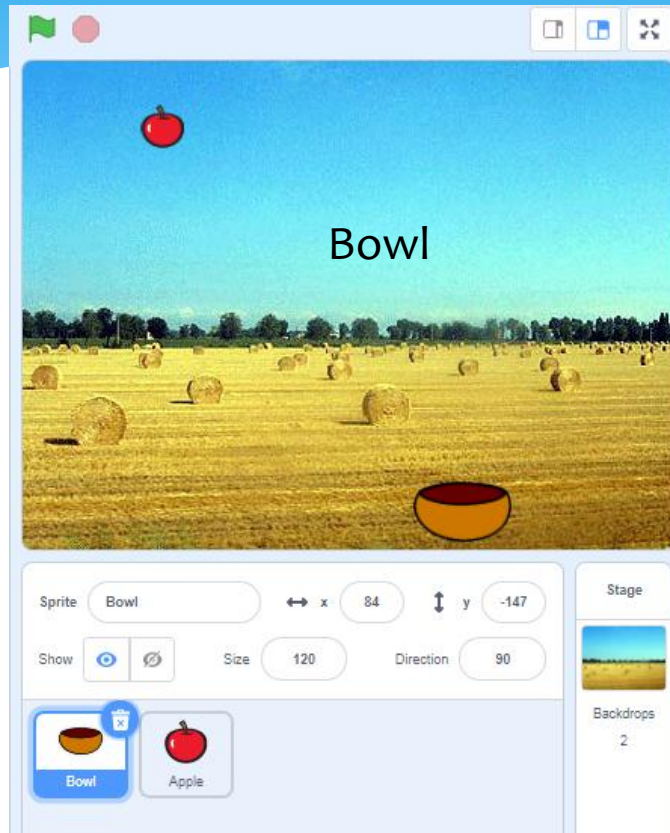
# Let's recall, what are variables

- \* Containers (or boxes) where sprites can keep bits and pieces of information and use them.
- \* Variables help us in creating interesting new features within games.
- \* Variables also help us to organize our code better.

# Let's make CATCH game

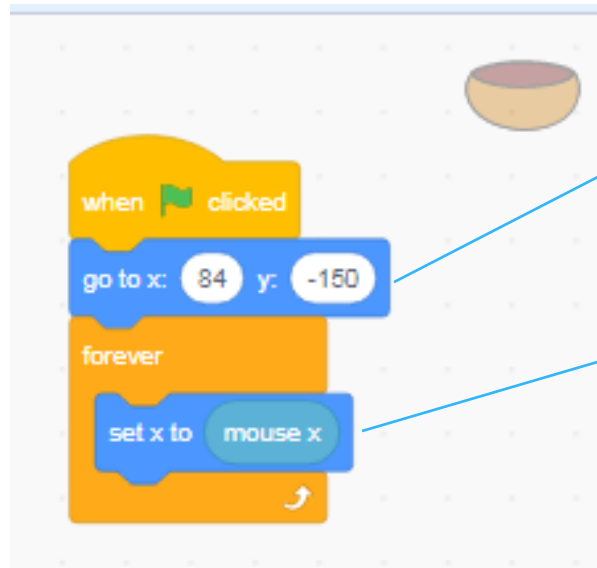
- \* A bowl on the ground is supposed to 'CATCH' fruits falling from the sky.
- \* For every apple caught, we score some points.
- \* For every orange caught, we score some points.
- \* But, if we catch a watermelon, we lose some points.
- \* Game lasts for a fixed number of seconds.
- \* You have to try and score as much as you can.

# Add backdrops and sprite



Note: We have intentionally added only the apple sprite so far. We will use a 'trick' to add other sprites later.

# Bowl code for movement



Send to the bottom of the stage

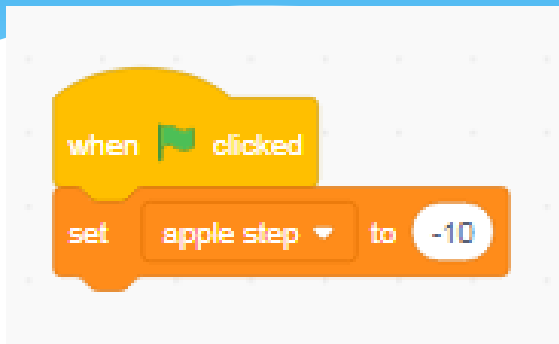
This code will ensure that the bowl moves left or right AS the mouse pointer is moved left or right.



block in the 'SENSING' blocks, which means the 'x value of mouse pointer.

Note: We can also use arrows for moving the bowl.

# Apple code (Initial Position)



```
when green flag clicked
  set apple step to -10
```

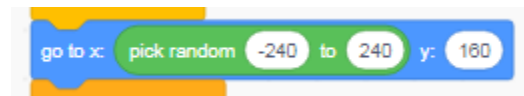
Use a variable to set APPLE SPEED.  
This will allow us to keep a clean code.

Apple Falling



```
when green flag clicked
  go to x: pick random -240 to 240 y: 160
  forever loop
    change y by apple step
```

Go to a 'random x position' on top of  
the stage,



```
go to x: pick random -240 to 240 y: 160
```

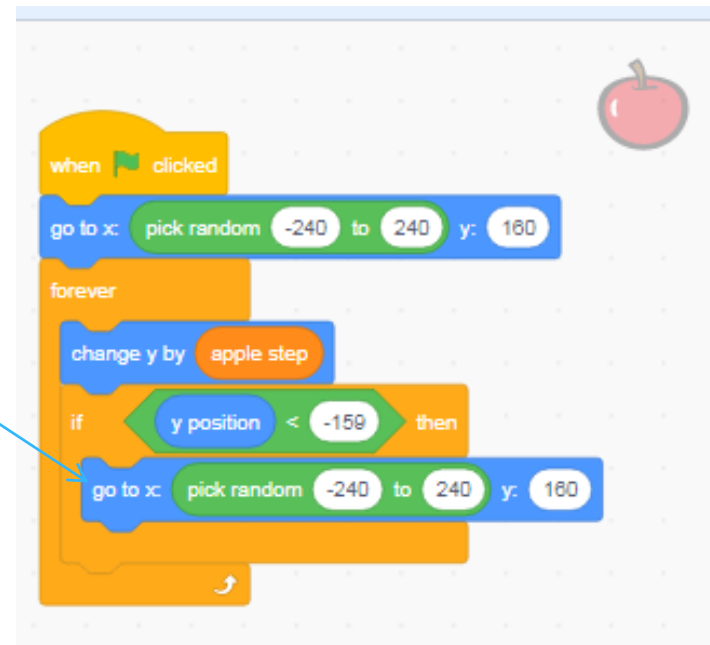
Pick a number between -240 and 240.

# Apple code (Go back to the top)

Once the apple has fallen to the ground, we want it to go back up, again to a 'random' x position.

If y position becomes smaller than -159, go back up

(recall, the bowl was at  $y = -150$ ).



```
when green flag clicked
  go to x: pick random -240 to 240 y: 160
  forever loop
    change y by apple step
    if y position < -159 then
      go to x: pick random -240 to 240 y: 160
```

The image shows a Scratch script for an apple. It starts with a 'when green flag clicked' event block. The first block is 'go to x: pick random -240 to 240 y: 160'. This is followed by a 'forever' loop containing three blocks: 'change y by apple step', an 'if' block with the condition 'y position < -159', and a 'then' block 'go to x: pick random -240 to 240 y: 160'. A blue arrow points from the text on the left to the 'if' block.



# Adding some more code to 'BOWL'

```
when green flag clicked
  set apple step to -10
  set score to 0
  set apple points to 10

when green flag clicked
  forever loop
    if touching Apple? then
      change score by apple points
      wait 0.4 seconds
```

Add variables

score – Total Score

Apple Points –

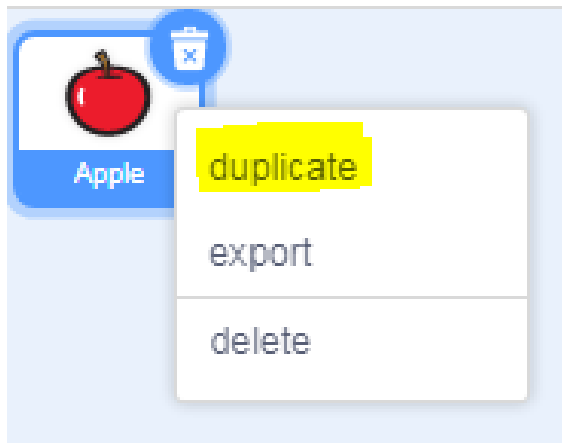
Points scored when I catch an apple

Add a small wait to avoid multiple 'touch' events from getting added!

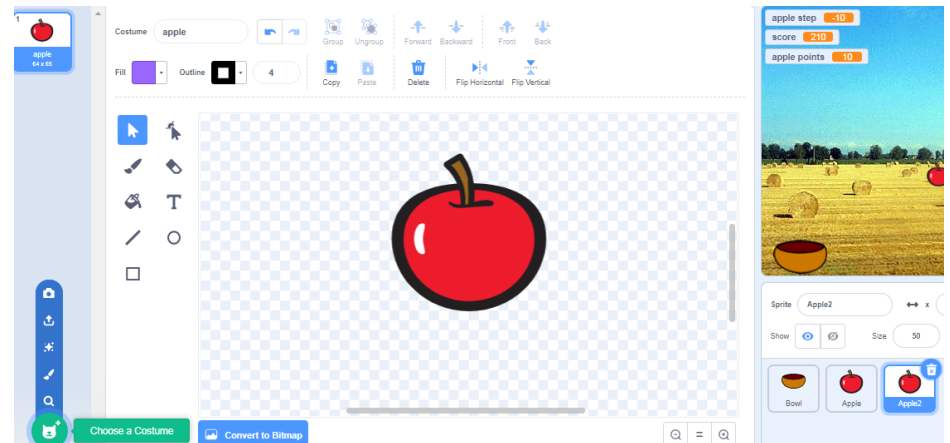
If I touch 'APPLE', change the score by 'apple score'. e.g, since 'apple points' is 10, every apple I catch increases my score by 10 points!

# Now, to add 'ORANGE' sprite

- \* Instead of adding all the code all over again, Select 'APPLE' sprite and use 'DUPLICATE code'.



Right Click



Change the sprite to Orange and re-name

# We have all the code ready

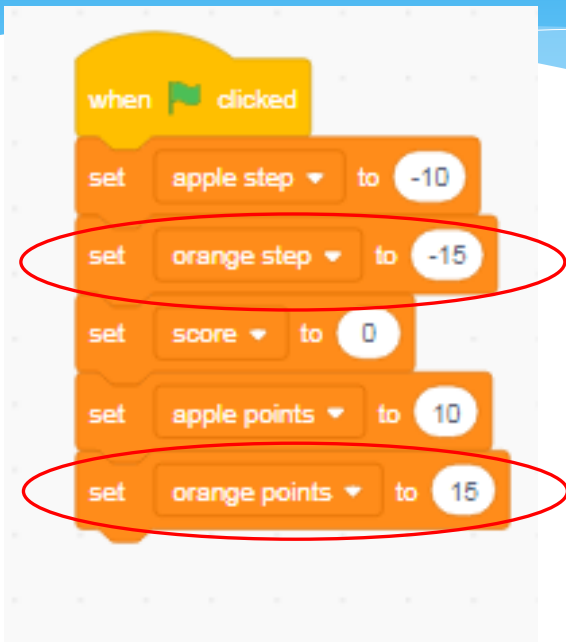
The image shows the Scratch code editor on the left and the stage on the right. The code editor contains the following blocks:

```
when clicked
  go to x: pick random -240 to 240 y: 180
  forever
    change y by apple step
    if y position < -159 then
      go to x: pick random -240 to 240 y: 180
```

The stage shows a background image of a field with hay bales. In the top right corner, there is a red apple sprite. In the bottom left corner, there is a brown bowl sprite. The stage has a score of 210 and an apple points value of 10. The 'apple step' variable is set to -10. The 'Sprite' dropdown is set to 'orange', with x and y coordinates of -3 and -45 respectively. The 'Show' dropdown is set to 'on', and the 'Size' and 'Direction' are set to 50 and 90 respectively. The 'Sprite' dropdown is currently set to 'orange', and the 'Bowl' and 'Apple' sprites are visible in the bottom right corner.

Just need to  
update this ...

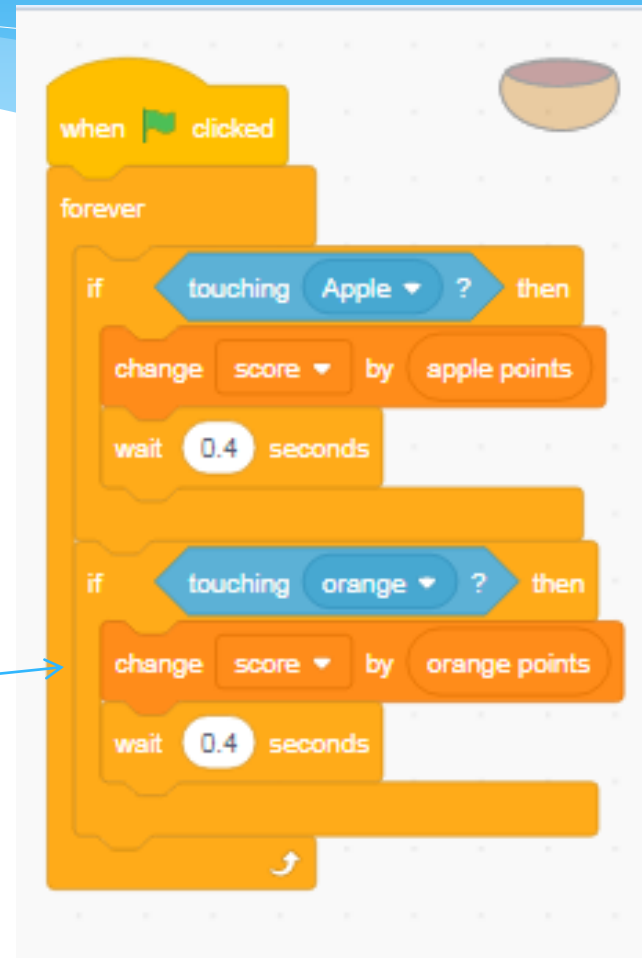
# Create a few more variables and update code



```
when clicked
  set apple step to -10
  set orange step to -15
  set score to 0
  set apple points to 10
  set orange points to 15
```

The image shows a Scratch code editor with a 'when clicked' event block followed by five 'set' blocks. The 'set orange step to -15' and 'set orange points to 15' blocks are circled in red.

Add code for detecting touching of orange.

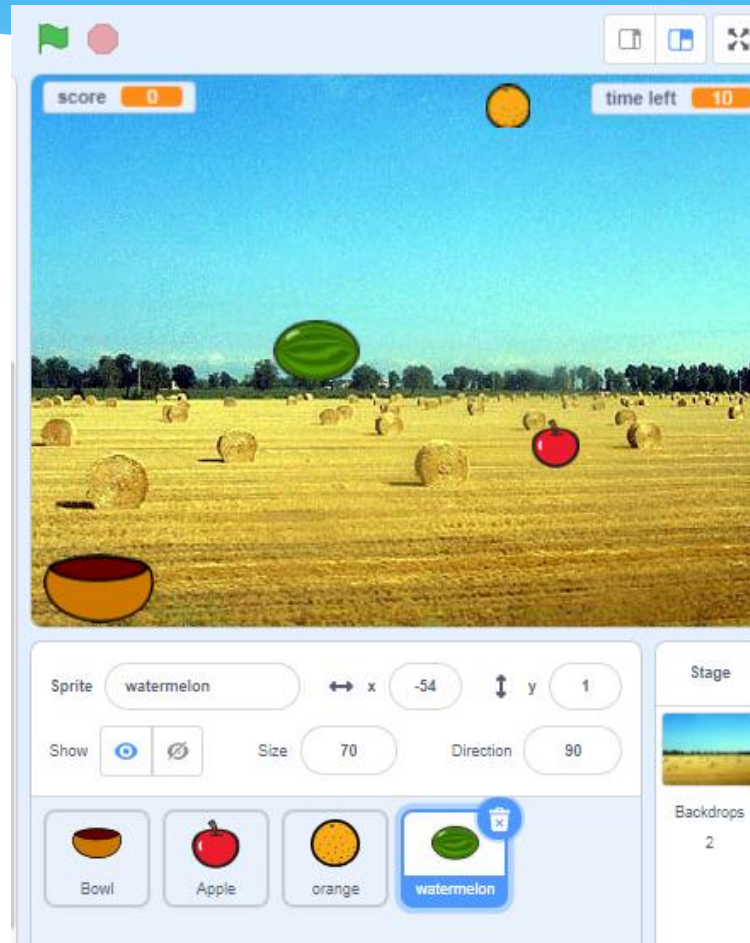


```
when clicked
  forever
    if touching Apple ? then
      change score by apple points
      wait 0.4 seconds
    if touching orange ? then
      change score by orange points
      wait 0.4 seconds
```

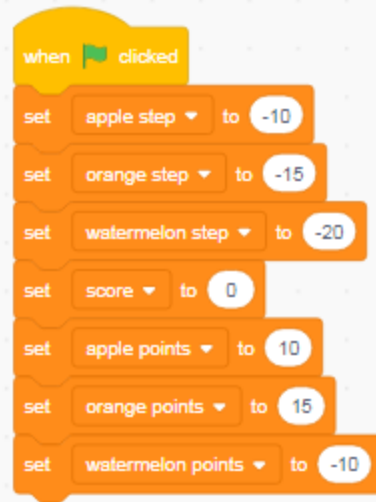
The image shows a Scratch code editor with a 'when clicked' event block followed by a 'forever' loop. Inside the loop, there are two 'if touching' blocks. The first 'if touching Apple ?' block contains a 'change score by apple points' block and a 'wait 0.4 seconds' block. The second 'if touching orange ?' block contains a 'change score by orange points' block and a 'wait 0.4 seconds' block. A blue arrow points from the text 'Add code for detecting touching of orange.' to the second 'if touching orange ?' block.

NOTE: Though we have here shown the orange code in the same loop, it is better to put it as a separate WHEN FLAG CLICKED loop. That way the processing for ORANGE and APPLE happens in parallel, not sequential.

# Similarly Add a Watermelon sprite



# And some variables ...



```
when clicked
  set apple step to -10
  set orange step to -15
  set watermelon step to -20
  set score to 0
  set apple points to 10
  set orange points to 15
  set watermelon points to -10
```

Add code for detecting touching of watermelon.



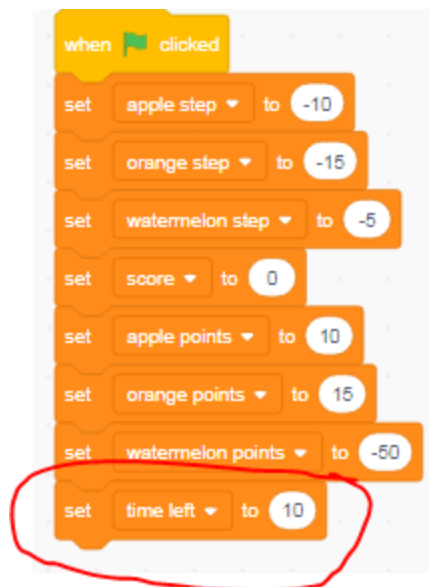
```
when clicked
  forever
    if touching Apple ? then
      change score by apple points
      wait 0.4 seconds
    if touching orange ? then
      change score by orange points
      wait 0.4 seconds
    if touching watermelon ? then
      change score by watermelon points
      wait 0.4 seconds
```

NOTE: Though we have here shown the orange code in the same loop, it is better to put it as a separate WHEN FLAG CLICKED loop. That way the processing for ORANGE, APPLE and watermelon happens in parallel, not sequential.

# Note


- \* We could also have placed the ‘if touching’ code in the individual sprites (Apple/Orange/Watermelon), and not the bowl sprite. That should also work just fine.

# Add a timer



```
when green flag clicked
  set apple step to -10
  set orange step to -15
  set watermelon step to -5
  set score to 0
  set apple points to 10
  set orange points to 15
  set watermelon points to -50
  set time left to 10
```

The image shows a vertical stack of Scratch code blocks. The first block is a yellow 'when green flag clicked' block. It is followed by seven orange 'set' blocks: 'apple step' to -10, 'orange step' to -15, 'watermelon step' to -5, 'score' to 0, 'apple points' to 10, 'orange points' to 15, and 'watermelon points' to -50. The final block is an orange 'set' block for 'time left' to 10, which is circled in red.



```
when green flag clicked
  repeat (time left)
    wait 1 seconds
    change time left by -1
  switch backdrop to Hay Field2
  stop all
```

The image shows a vertical stack of Scratch code blocks. It starts with a yellow 'when green flag clicked' block. This is followed by a loop structure: an orange 'repeat' block with 'time left' in the count field, containing a 'wait 1 seconds' block and a 'change time left by -1' block. Below the loop is a purple 'switch backdrop to' block set to 'Hay Field2', and finally a yellow 'stop all' block. Blue arrows point from the text on the right to the 'repeat' block, the 'wait 1 seconds' block, and the 'stop all' block.

Notice, the game will last for 'time left seconds'

Stop everything when the time is over.



# And you are all set ...

- \* We saw how we used concepts that we have learnt till now in making a very simple but engaging game.
- \* We also saw how variables helped us – in keeping track of time and score, and also in keeping the code clean.
- \* With this, you are all set for your independent activity – 7: A catch game. Enjoy!