The 6-Horse Race

https://scratch.mit.edu/projects/386217506

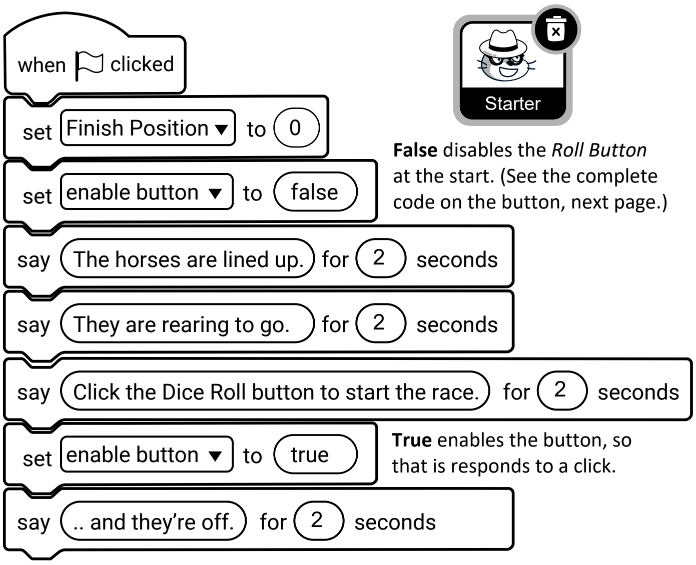
This is a race game with 6 horses on a six-lane track of 12 squares; ten race squares, plus start and finish squares. Each horse is identified by a number and colour that correspond to the six sides of a dice. Two similar dice are used. When a horse's number is 'rolled' the horse of that colour moves **one** square. If both dice roll the same number, the horse moves **two** squares.

1. the Backdrop

You may export the backdrop from this project to your computer and upload it to your own project. To draw your own backdrop, start with the 20px grid in the Backdrop library (by name *Xy-grid-20px*). Each track square is 40 pixels (that's 2 x 2, grid squares of 20 pixels). Accuracy will make it easier to get the coding right.

2. The Starter sprite's Code

The variable *enable button* is an ordinary variable except that you give it only two Boolean values, **true** and **false.** You use it to disable the button until the Starter has finished his instructions. Then you enable the button.



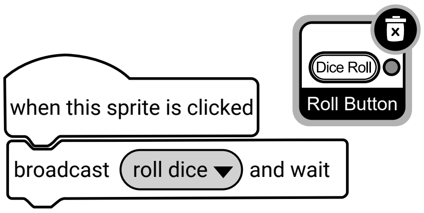
```
when ⚑ clicked
set [Finish Position ▾] to (0)
set [enable button ▾] to (false)
say (The horses are lined up.) for (2) seconds
say (They are rearing to go.) for (2) seconds
say (Click the Dice Roll button to start the race.) for (2) seconds
set [enable button ▾] to (true)
say (.. and they're off.) for (2) seconds
```

False disables the *Roll Button* at the start. (See the complete code on the button, next page.)

True enables the button, so that is responds to a click.

3. The Button sprite

Once the **Button** is enabled, when clicked it rolls two dice simultaneously, *Dice 1* and *Dice 2* like real dice. A single click produces two *random* numbers. Here is the main part of the code on the *Roll Button.*

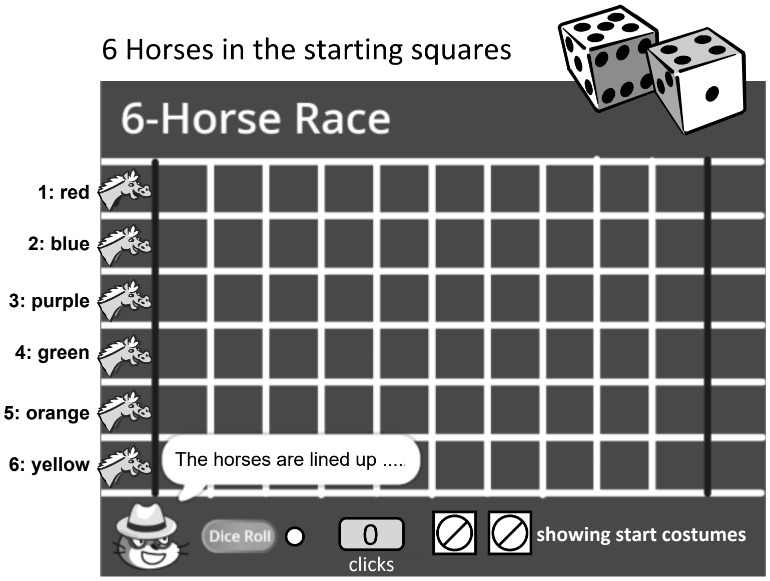


```
when this sprite is clicked
broadcast (roll dice ▾) and wait
```

This 'roll dice' message is sent to and received simultaneously by both *Dice 1* and *Dice 2*. Each dice will respond by supplying a number.

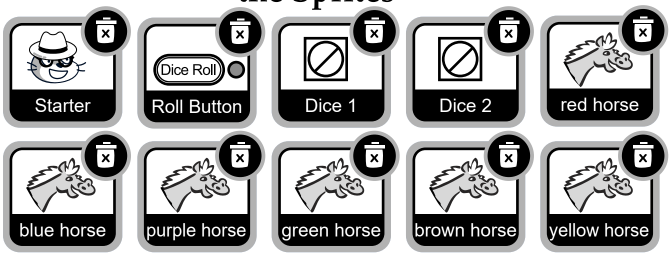
See next page for the complete code on the button.

6 Horses in the starting squares



Starter Button Dice 1 Dice 2

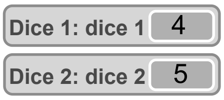
the Sprites



(The **Finish** line is also a sprite. Click *See page 3*

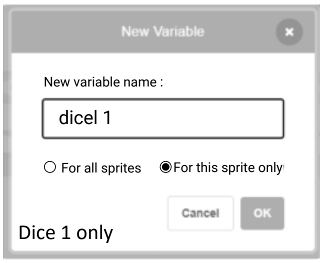
4. The 2 Dice sprites and their *local variables*

As there are no dice sprites in Scratch you may export them from this project or create your own similar dice. Code *Dice 1,* then duplicate it to create *Dice 2* and make the small changes. Each dice will need its own (local) variable to store its value after the **Roll Button** is clicked. The variable names are also *dice 1* and *dice 2*. When making the variable for each dice, choose *For this sprite only.* This is what makes it a **local** variable. Each dice will choose its own random number between 1 and 6.



Dice 1: dice 1 4

Dice 2: dice 2 5



New Variable ×

New variable name :

dicel 1

○ For all sprites ● For this sprite only

Cancel OK

Dice 1 only

A **local variable** allows for the roll of one dice to differ from the other. The *roll 1* variable is if Dice 1 only. The *roll 2* variable is for Dice 2 only.

A **local variable** is only visible to its own sprite. That's the reason why the name of the sprite is given before the name of the variable in the *readout monitor.*

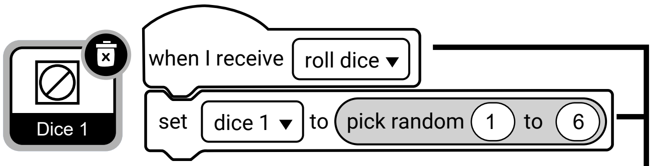
5. Other variables *can be 'for all'* (*global*)

Another important part of the set up, is to create the variable *enable* on the **Roll Button**. As explained, the *Starter* sprite uses it to disable and enable the *Button.* Make a sprite named *clicks,* to count the number of clicks. Make the variable *Finish Position* on the **Finish** sprite.

6. Summary of the Game Algorithm

When the Button is enabled and clicked, it broadcasts one message *roll dice* to *Dice 1* and *Dice 2*. What then?

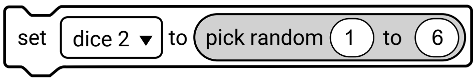
6. 1.



```
when I receive [roll dice ▾]
set [dice 1 ▾] to (pick random (1) to (6))
```

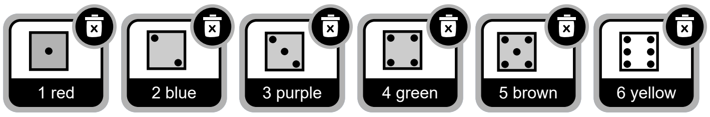
The *dice 1* variable is set to any number from 1 to 6.

Here is an illustration of what happens next. The *set variable to (pick random)* block on *Dice 1* sets a random value of 1 to 6 to the **local variable** named *dice 1*. *Dice 2* likewise, when it receives the broadcast, it sets its own random value of 1 to 6 to the **local variable** named *dice 2*.



```
set [dice 2 ▾] to (pick random (1) to (6))
```

***dice 2* is set to any number from 1 to 6**



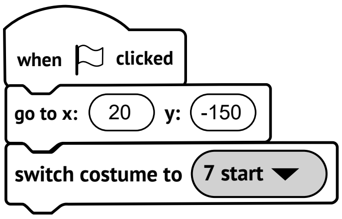
See opposite to complete the code on *Dice 1*.

6.2 Depending on the number that was rolled, *Dice 1* sends the *redMove1* broadcast to horse 1 (red), *blueMove1* to horse 2 (blue), *purpleMove1* to horse 3 (purple) and so on, when a 4 or 5 or 6 is rolled by *Dice 1*. Similarly *Dice 2* sends the *greenMove2* broadcast to green horse 4, *brownMove2* to brown horse 5, *yellowMove2* to yellow horse 6 and so on when a 1, 2 or 3 is rolled by *Dice 2*. So, if horse 6 (yellow) receives the broadcast *yellowMove1*, from *Dice 1*, the yellow horse moves 40 pixels (which is one track square) forward.

For example, if brown horse 5 receives the broadcast *brownMove1*, from *Dice 1* the brown horse moves 40 pixels (or one track square) forward. If on the same click, the brown horse receives the broadcast *brownMove2* from *Dice 2*, the brown horse moves 80 pixels (or two squares) forward. On any dice roll, two horses move one square forward or (if a pair of the same numbers is rolled) one horse of that number moves forward 2 squares.

8. Scripts on Dice 1 each dice has two scripts

Script 1 of 2



```
when ⚑ clicked
go to x: (20) y: (-150)
switch costume to [7 start ▾]
```

start view costume 7

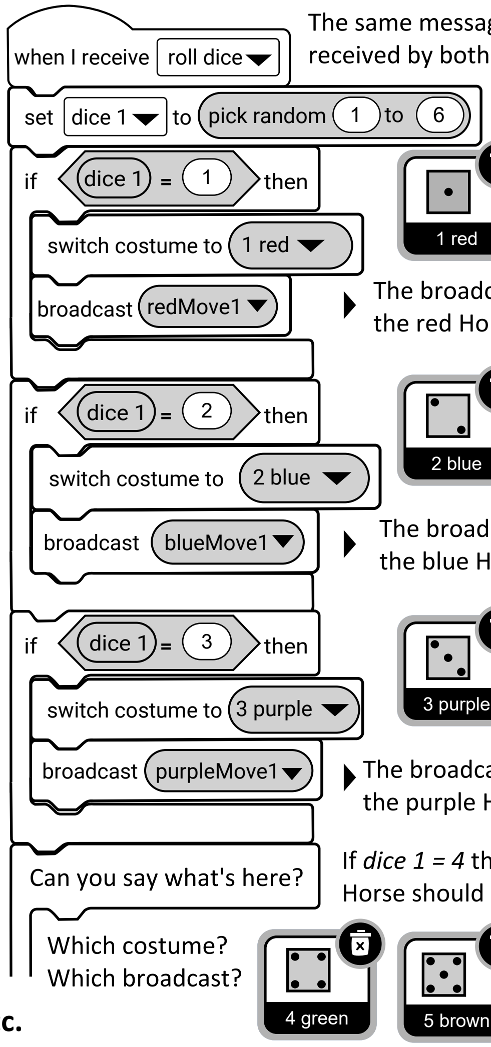
Horse scripts next page

Dice 2 is positioned at x: 60, y:-150

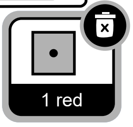
The two scripts on Dice 2 follow a similar pattern to these two scripts, apart from the variable name which is *dice 2* and broadcasts prefixed by the horse colour: *[colour]Move2*

Script 2 of 2

The same message is received by both dice

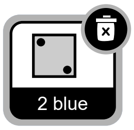


```
when I receive [roll dice ▾]
set [dice 1 ▾] to (pick random (1) to (6))
if <(dice 1) = (1)> then                      1
    switch costume to [1 red ▾]
    broadcast [redMove1 ▾]
if <(dice 1) = (2)> then                      2
    switch costume to [2 blue ▾]
    broadcast [blueMove1 ▾]
if <(dice 1) = (3)> then                      3
    switch costume to [3 purple ▾]
    broadcast [purpleMove1 ▾]
Can you say what's here?                      4
    Which costume?                            etc.
    Which broadcast?
```



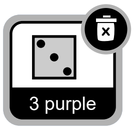
Switch costume to 1 red

▶ The broadcast instructs the red Horse to move



Switch costume to 2 blue

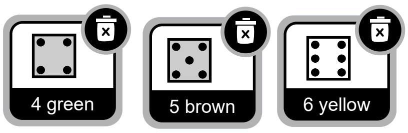
▶ The broadcast instructs the blue Horse to move



Switch costume to 3 purple

▶ The broadcast instructs the purple Horse to move

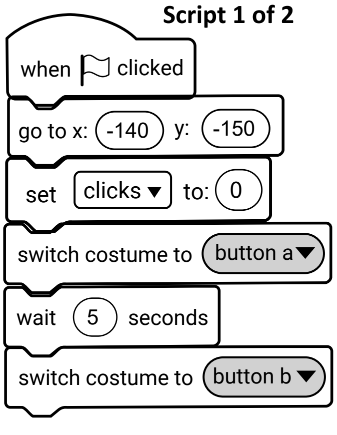
If *dice 1 = 4* then, which Horse should move?



Continue the pattern for *Dice 1* rolling 4, 5 or 6

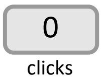
7. The Button Script in full

Script 1 of 2



```
when ⚑ clicked
go to x: (-140) y: (-150)
set [clicks ▾] to: (0)
switch costume to [button a ▾]
wait (5) seconds
switch costume to [button b ▾]
```

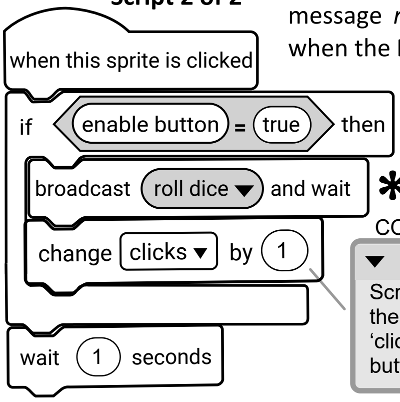
Includes a red dot that turns green



The button has two costumes, a small red dot before the button is *enabled* which turns green to show that it is enabled. The race can start. The *clicks* variable is set to zero in script 1. It increases by one, each time the button is clicked (as you can see in script 2)

Script 2 of 2

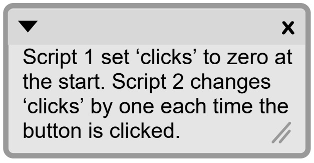
The second script **broadcasts** the message *roll dice to* each dice when the **Roll Button** is clicked.



```
when this sprite is clicked
if <(enable button) = (true)> then
    broadcast [roll dice ▾] and wait   *
    change [clicks ▾] by (1)
wait (1) seconds
```

It is **true** means. You can click the button.

CODE COMMENT



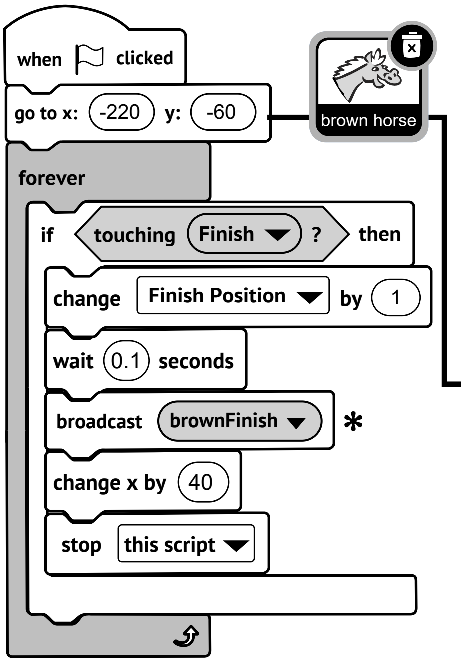
Script 1 set 'clicks' to zero at the start. Script 2 changes 'clicks' by one each time the button is clicked.

9. Code on each of the 6 Horse Sprites (Red, Blue, Purple, Green, **Brown**, Yellow)

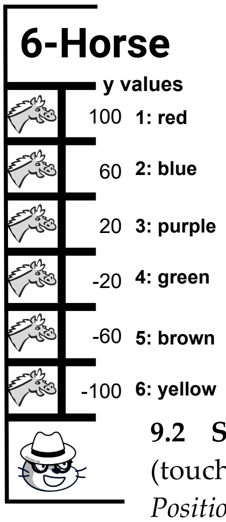
EXAMPLE: Full code on the brown horse (horse 5)

There are 4 scripts on each horse.

script 1 of 4 (Setup and Finish)



```
when [flag] clicked
go to x: (-220) y: (-60)
forever
    if <touching (Finish v) ?> then
        change [Finish Position v] by (1)
        wait (0.1) seconds
        broadcast (brownFinish v) *
        change x by (40)
        stop [this script v]
    end
end
```

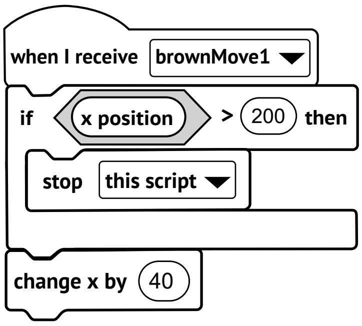


9.1 The 4 scripts on each horse follow a similar pattern, with changes to the **y** position of each horse and the name of each finishing **broadcast** (in scripts 1 and 4)**.** You only need to code one horse, then drag to copy the code to the other horses and make the small changes.

This Example: The block, **go to x: y:** sets the position of the *brown horse (5)* at the coordinates **x: -220 y: -60**. You can see that all the horses have the same **x:** coordinate but each horse is 40 pixels higher than the horse below it. So the **y:** coordinates differ by -40 from top to bottom for each horse.

9.2 Script 1. In the *forever* loop, each time a horse reaches (touches) the **Finish** line sprite, the variable named *Finish Position* increases by one. As this code is on the *brown horse (5)*, the **Finish** line sprite broadcasts the message *brownFinish* and the brown horse 'jumps' 40 pixels into the finishing square. If *Finish Position* happens to be 1, then it says "I'm first" (Script 4).

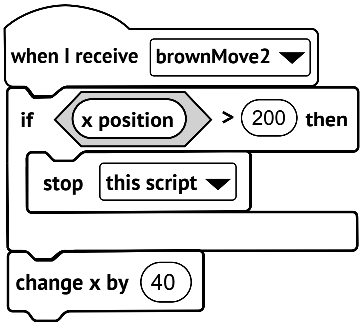
script 2 of 4 (Dice 1 RACE SCRIPT)



```
when I receive [brownMove1 v]
if <(x position) > (200)> then
    stop [this script v]
end
change x by (40)
```

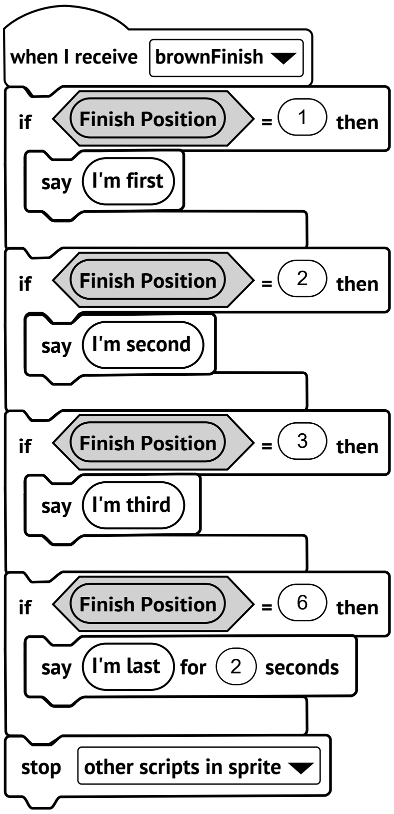
Script 2 is what happens when *Dice 1* is a 5. The brown horse moves 40 pixels (one square). **Script 3** is what happens when *Dice 2* is a 5. The brown horse moves 40 pixels (one square).

script 3 of 4 (Dice 2 RACE SCRIPT)



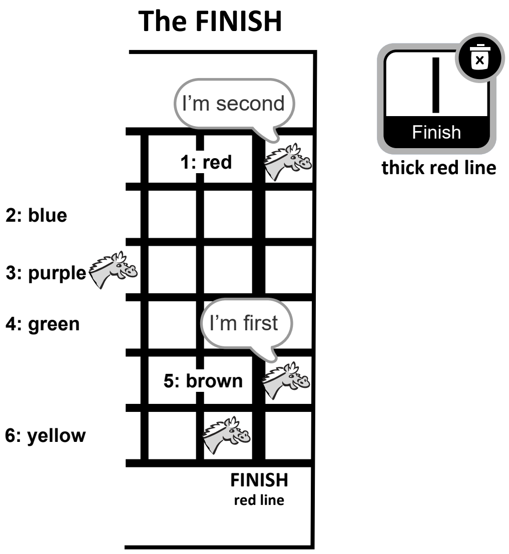
```
when I receive [brownMove2 v]
if <(x position) > (200)> then
    stop [this script v]
end
change x by (40)
```

script 4 of 4

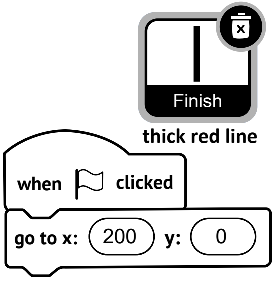


```
when I receive [brownFinish v]
if <(Finish Position) = (1)> then
    say (I'm first)
end
if <(Finish Position) = (2)> then
    say (I'm second)
end
if <(Finish Position) = (3)> then
    say (I'm third)
end
if <(Finish Position) = (6)> then
    say (I'm last) for (2) seconds
end
stop [other scripts in sprite v]
```

9.3 Scripts 2 and **3** are the RACE SCRIPTS which move the horses along until they reach the winning post at **x: 200**. Then they stop and the broadcast from Script 1 makes them declare their finishing position, 1st,2nd,3rd or last.



10. The Finish script



```
when [flag] clicked
go to x: (200) y: (0)
```

Summarising How the Race is Won

The *Finish* sprite is set at **x: 200 y: 0.** The first horse to touch the *Finish* declares itself the WINNER. The second horse declares itself second, and so on. You can figure out from the scripts how to do 4th , 5th . The last horse says 'I'm last'. The winning order is achieved by using a variable named **Finish Position**. To understand how this is used, click ***See inside***. You will see that *Finish Position* starts with a value of zero in the Starter's script. As each horse touches the *Finish line* the value of the variable increases by 1. The last horse takes the value of *Finish Position* to six, and says 'I'm last'.

■ See also the horse Derby replica project at **scratch.mit.edu/projects/386982734** and get the *Derby Maths Sheet* in the Files section of the Facebook Group *Teaching with Scratch*

6-Horse Race Fact-finding Sheet

https://scratch.mit.edu/projects/386217506

1. How Many Dice Roll Clicks to WIN and LOSE

There is a **click counter** built into the 6-Horse Race. Run the race online 3 times and write down the number of dice roll clicks it takes to WIN and how many to get the LAST horse to reach the finish. Over three races, what is the **least** number of clicks to WIN and the **highest** number to come in LAST? **Circle the two numbers:**

	RACE 1	RACE 2	RACE 3
CLICKS to WIN			
CLICKS to Come in LAST			

2. Collect Data from 5 Races

Run the 6-Horse Race 5 times online and write the **W**inner, **2**nd, **3**rd and **L**ast horse in each race using **W, 2, 3, L, T** (for a tie)

a. Did any horse win 3, 4 or 5 times? **Y N**

if Yes: Which horse ________

b. Was any horse last 3, 4 or 5 times? **Y N**

if Yes: Which horse ________

c. Which horse (or horses)
won most times? _______ _______

d. Which horse (or horses)
came last most times? ________ _________

e. Rate the horses (1 *best*, 2, 3, 4, 5 *worst*)

1_____ _2_____ ___3___ ____4___ _____5 ______ 6

		RACE 1	RACE 2	RACE 3	RACE 4	RACE 5
1 red						
2 blue						
3 purple						
4 green						
5 brown						
6 yellow						

3. A Race on Paper

Here are 20 clicks in order, 1 to 20. One horse wins before the 20th click and another comes second at the 20th. Mark each horse's progress with an **X** and find out which is the winner, which is second, which horse is coming 3rd and which is likely to be last. Write **W, 2, 3** and **L**. Write the number of clicks also.

Roll 1	Roll 2	Roll 3	Roll 4	Roll 5	Roll 6	Roll 7	Roll 8	Roll 9	Roll 10
1 3	2 3	4 5	2 1	1 3	3 1	1 1	5 2	5 6	5 5

Roll 11	Roll 12	Roll 13	Roll 14	Roll 15	Roll 16	Roll 17	Roll 18	Roll 19	Roll 20
2 1	6 3	1 6	5 5	4 1	3 6	1 5	3 5	1 6	5 3

The first and second rolls are marked X. Continue with remaining eighteen rolls.

													Number of Clicks
1 red		X_1										....	
2 blue		X_2										....	
3 purple		X_1	X_2									....	
4 green												....	
5 brown												....	
6 yellow												....	

4. Counting in 40s x coordinates:

Each horse starts at **x: -220**. It moves forward by 40 steps each time. Write the x value of each square of the race track. The start position and after-finish position are given. Can you see a symmetry in the number pattern?

x: -220											x: 220

I'm at the finish line

x: 220 (after the finish line)

Seamus O'Neill © 2020