Sprite whacks the mole …

Vineet Srivastava

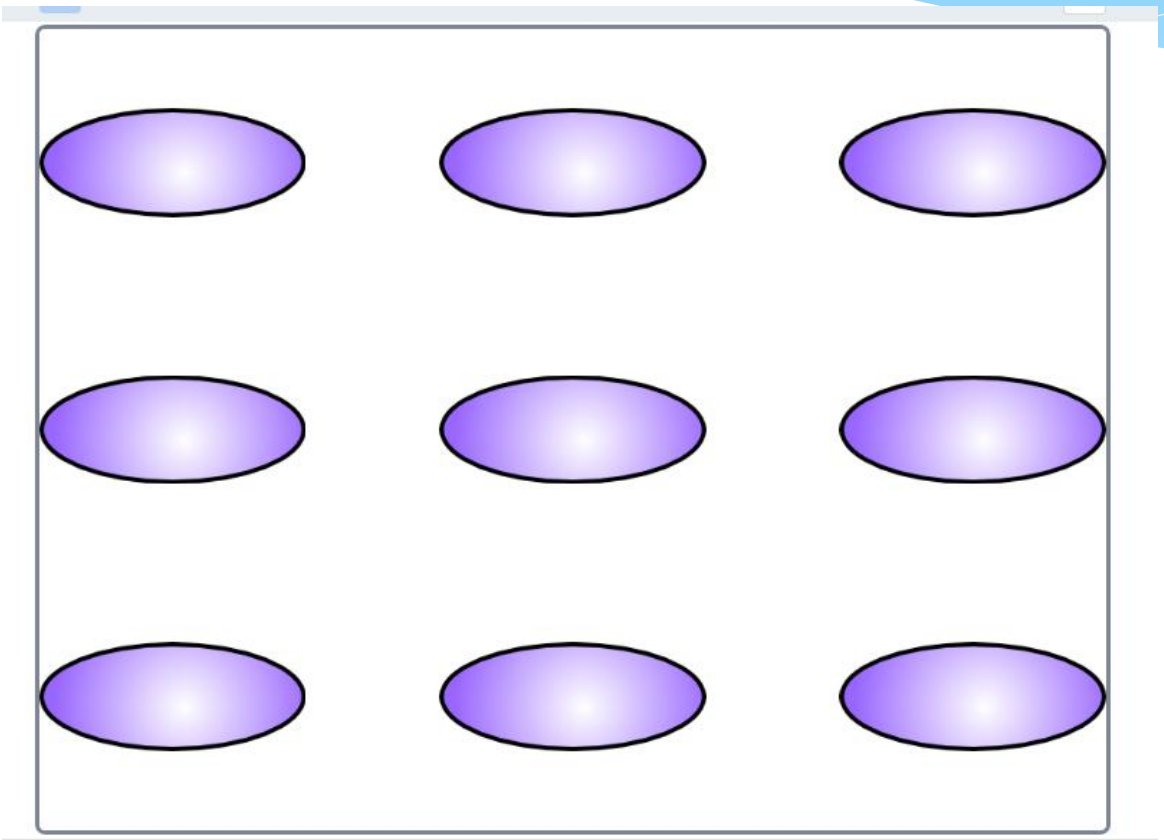
In this lesson, we will …

* Build a version of the popular ‘whack-a-mole’ game.
* In the process, we will use all the concepts that we have learnt previously, along with some logic.
* We will also, for the first time, use a version of NESTED loops – repeat inside a repeat – this is a powerful programming construct with a very wide applicability.

What is the ‘Whack-a-Mole’ game?

* A ‘MOLE’ keeps coming out of a hiding pit every now and then.
* We (controlling a hammer) have to ‘whack’ the MOLE when the it comes out of hiding.

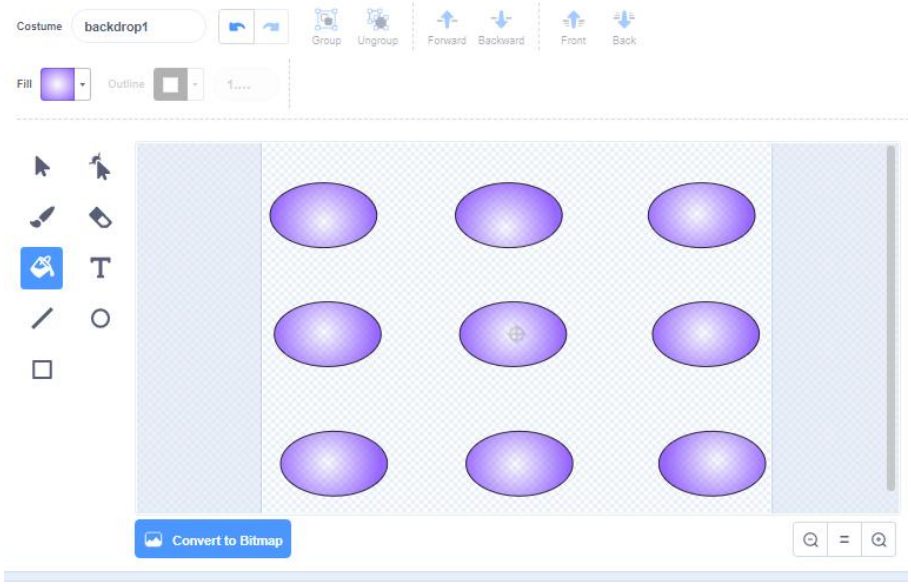
Background



* On the background, we need a regular pattern on hiding holes.

Option 1: Simpler Option

* Make the holes on the backdrop.
* This is easy, but not flexible. Hence we will not use this option.



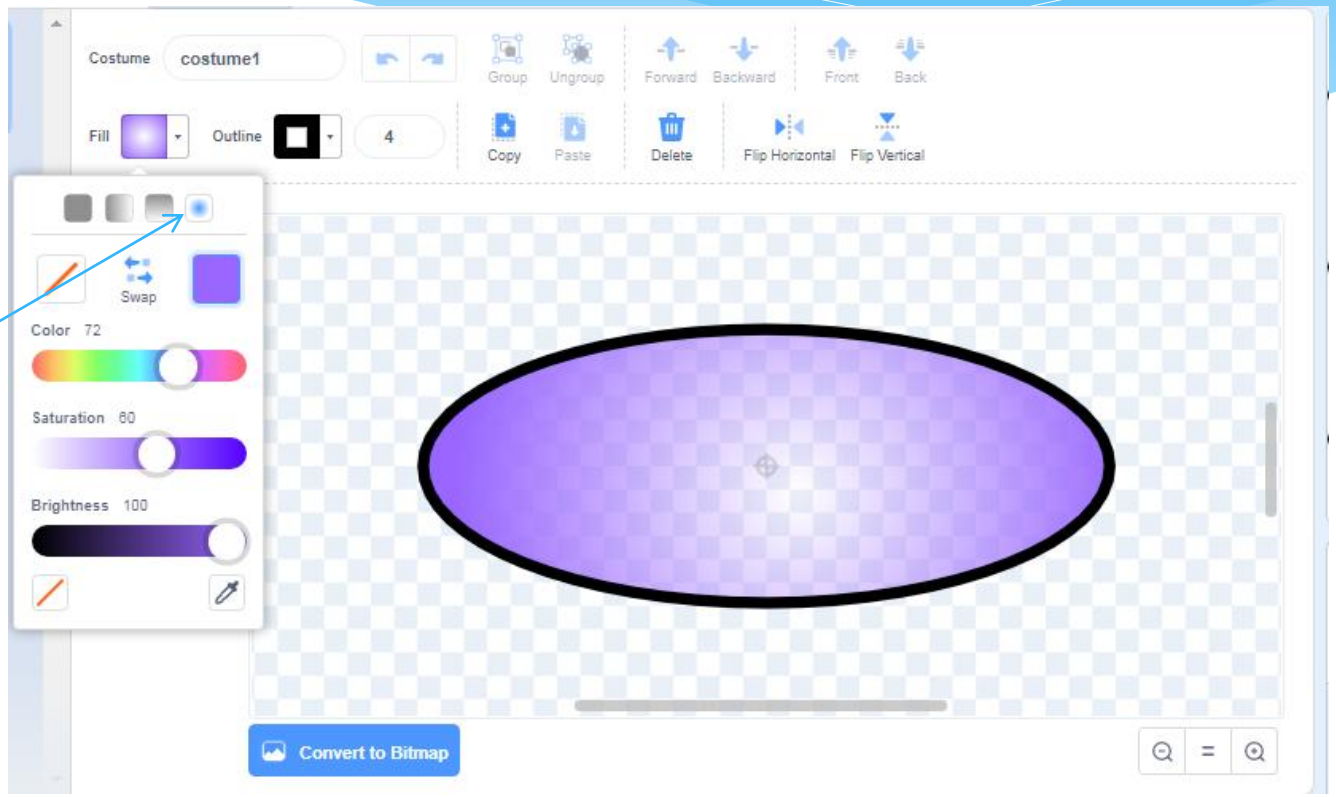
Option 2: Draw the background with CLONING

* Remember, cloning creates ‘copies’ of a sprite.
* Hence, if we can create copies of a ‘PIT’ sprite, we will have the backdrop ready.
* Using cloning this way is useful for many games and applications, hence good to learn.

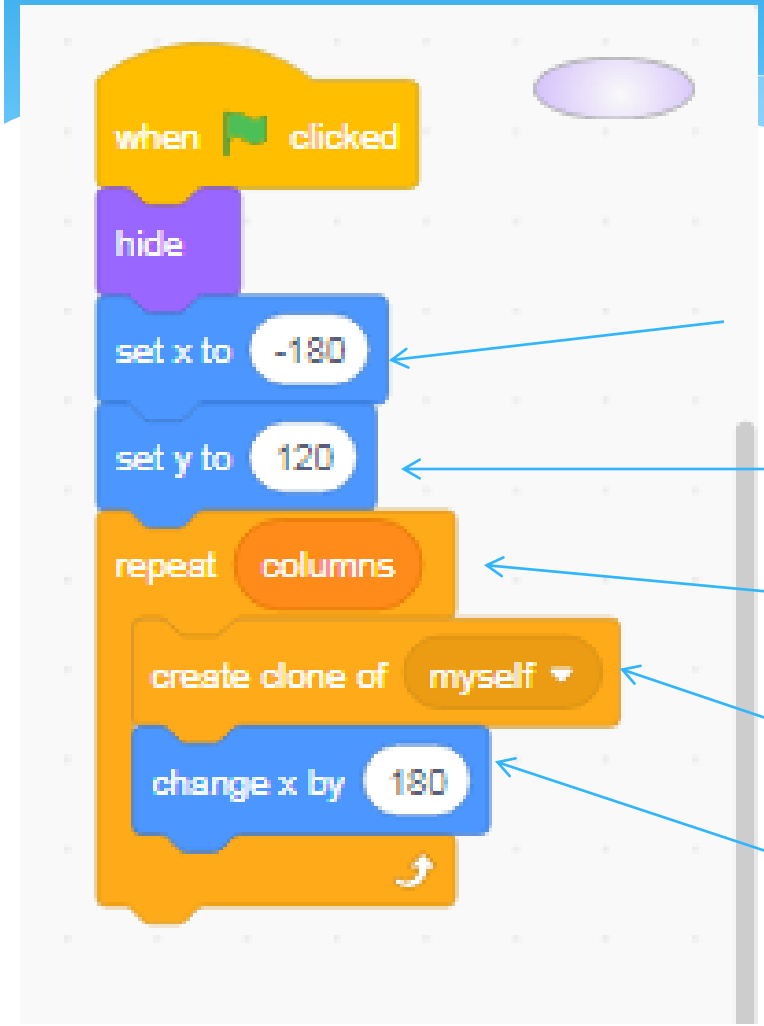
* Logic:
  * Create a single sprite, and make its copies at different locations.
  * Decide number of rows, number of columns.
  * First create 1 row.
  * Then create multiple rows.

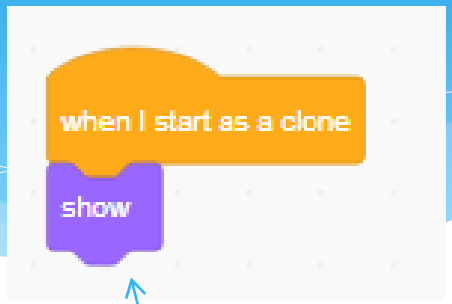
Create a Pit Sprite

Notice, we have used this effect here to get the two colour effect



Creating 1 row of Pits





Go to the left corner

Set the y- location

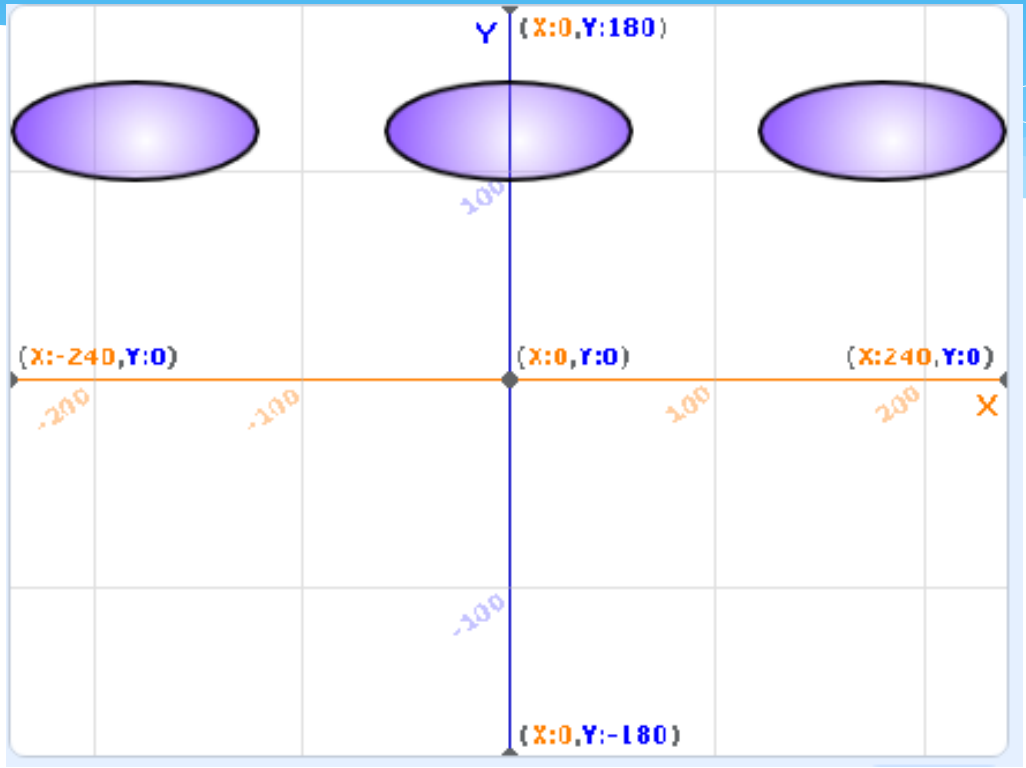
Repeat 'columns' times. E.g., in this case, repeat '3' times.

Create a CLONE

Change x by 180, that means, move RIGHT by 180 steps

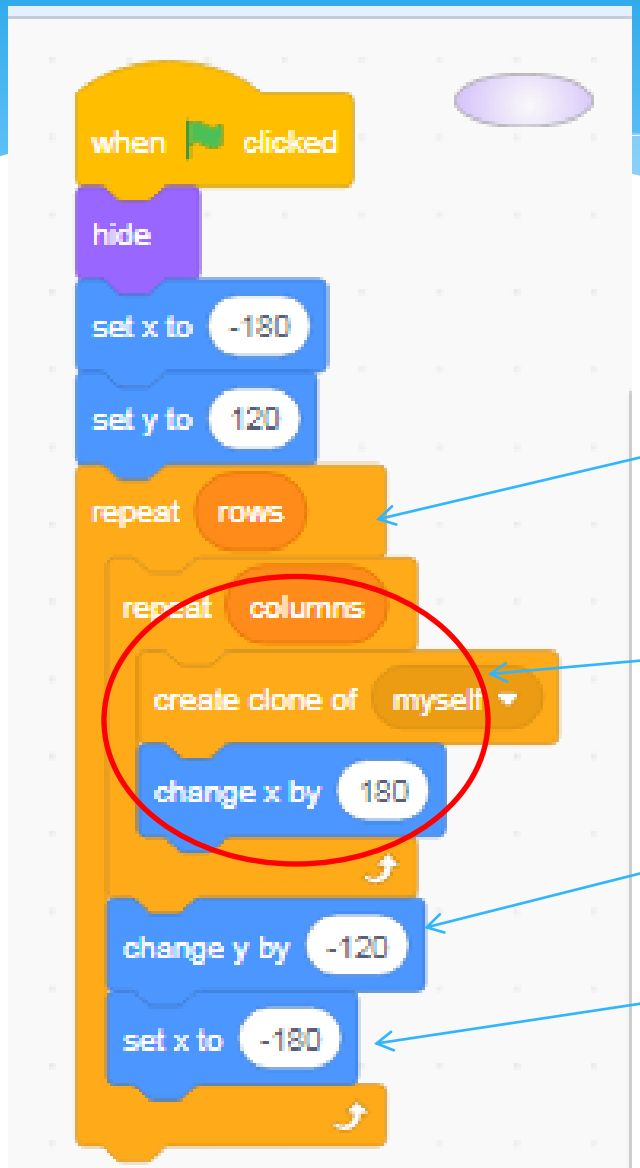
When the clone starts, just 'SHOW'

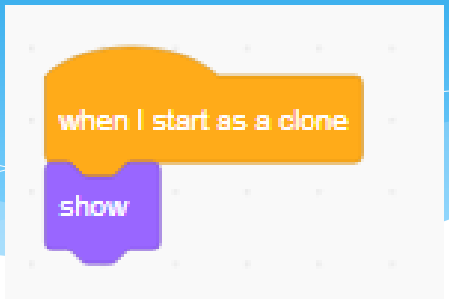
The top row is ready!



* Now, with the top row ready, we have to change ‘y’ and repeat this for different rows. For example, if we have three rows, then 3 times.

Code





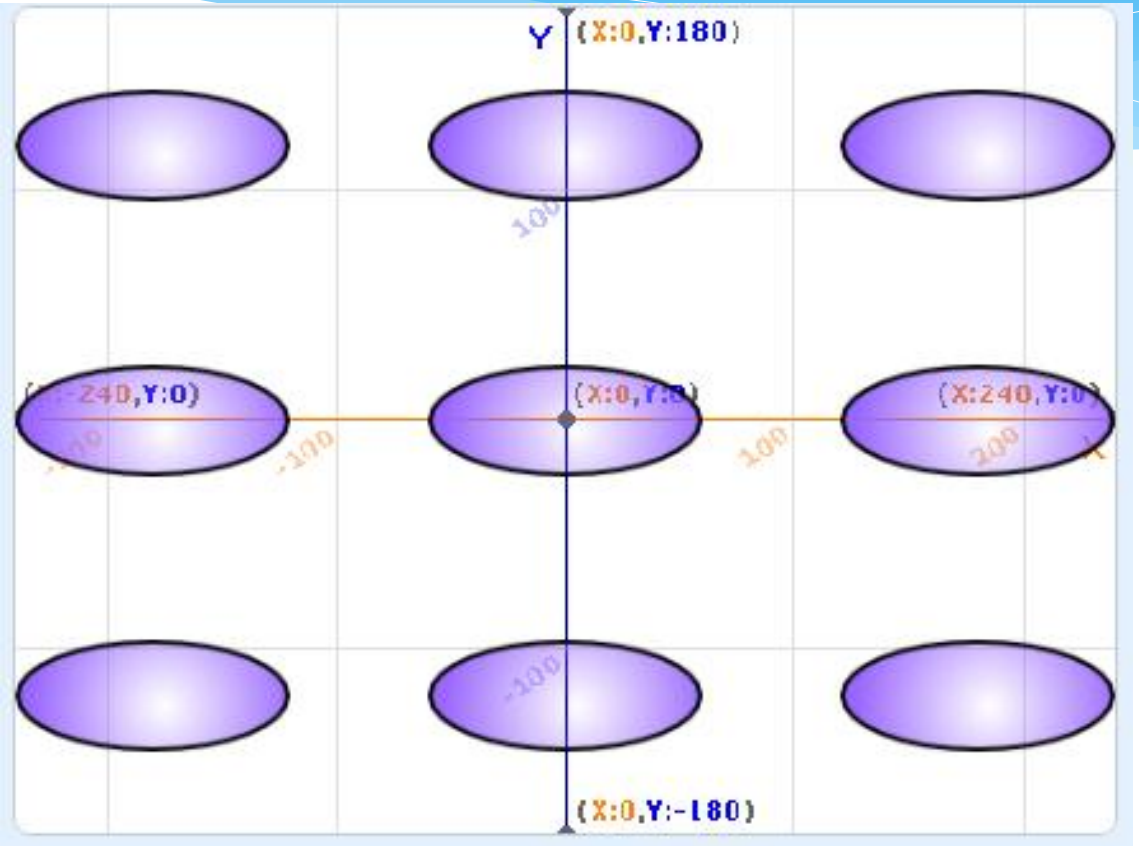
Repeat 'rows' times – that is 3 times.

This part of the code is exactly same as earlier – it creates 1 row of sprites

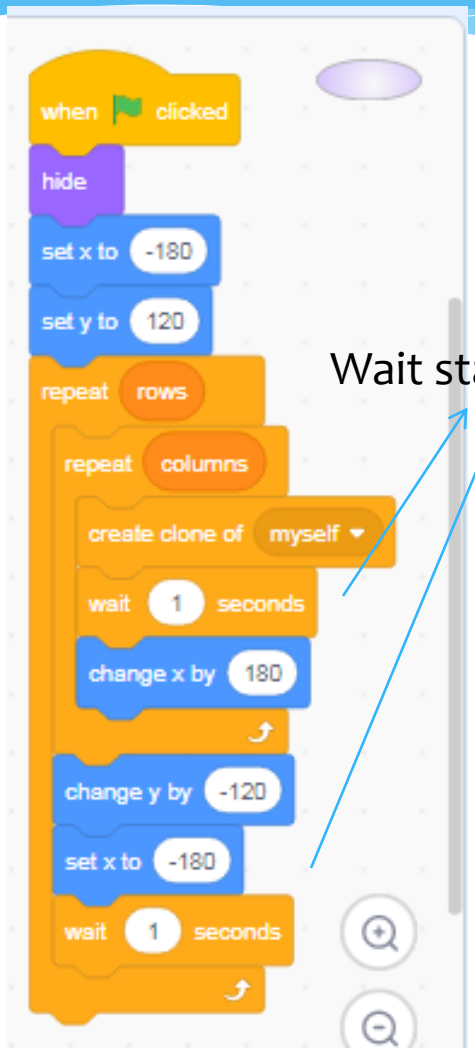
After 1 row is over, go to the new 'y' location. (Notice, we are lowering the 'y')

Go back to the original 'x'. This is important, so that the shape is maintained.

A grid of PIT sprites



Add WAIT to see and understand the process



Wait statements

Notice: The wait statements are not needed for the code, but adding these helps us to understand and visualize the process better.

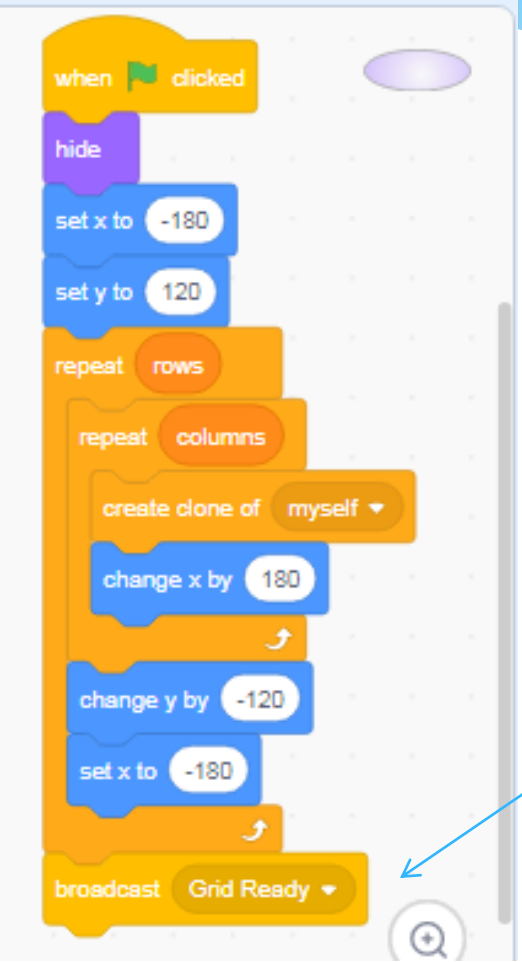
Also, add the 'x-', 'y-' backdrop to understand the entire process.

Another method to learn this logic could be to NOT use repeat. Instead use set x and set y for all the cloning locations and then make that code shorter with repeat. You will see the code is becoming quite repetitive. Hence it can be made shorter with this dual loop.

What is the benefit of using CLONING for grid?

* The benefits of this method are:
  * Flexible design – you can change the rows and columns to change the design.
  * We know ‘precisely’ what x- and y- locations are the holes located at.
  * We can just duplicate this for the mole sprite.
  * It is a very good practice of programming logic – we can represent the entire code like a flowchart.

After the grid is ready, broadcast



At the end of the grid getting ready, use broadcast to get other parts of the game started.

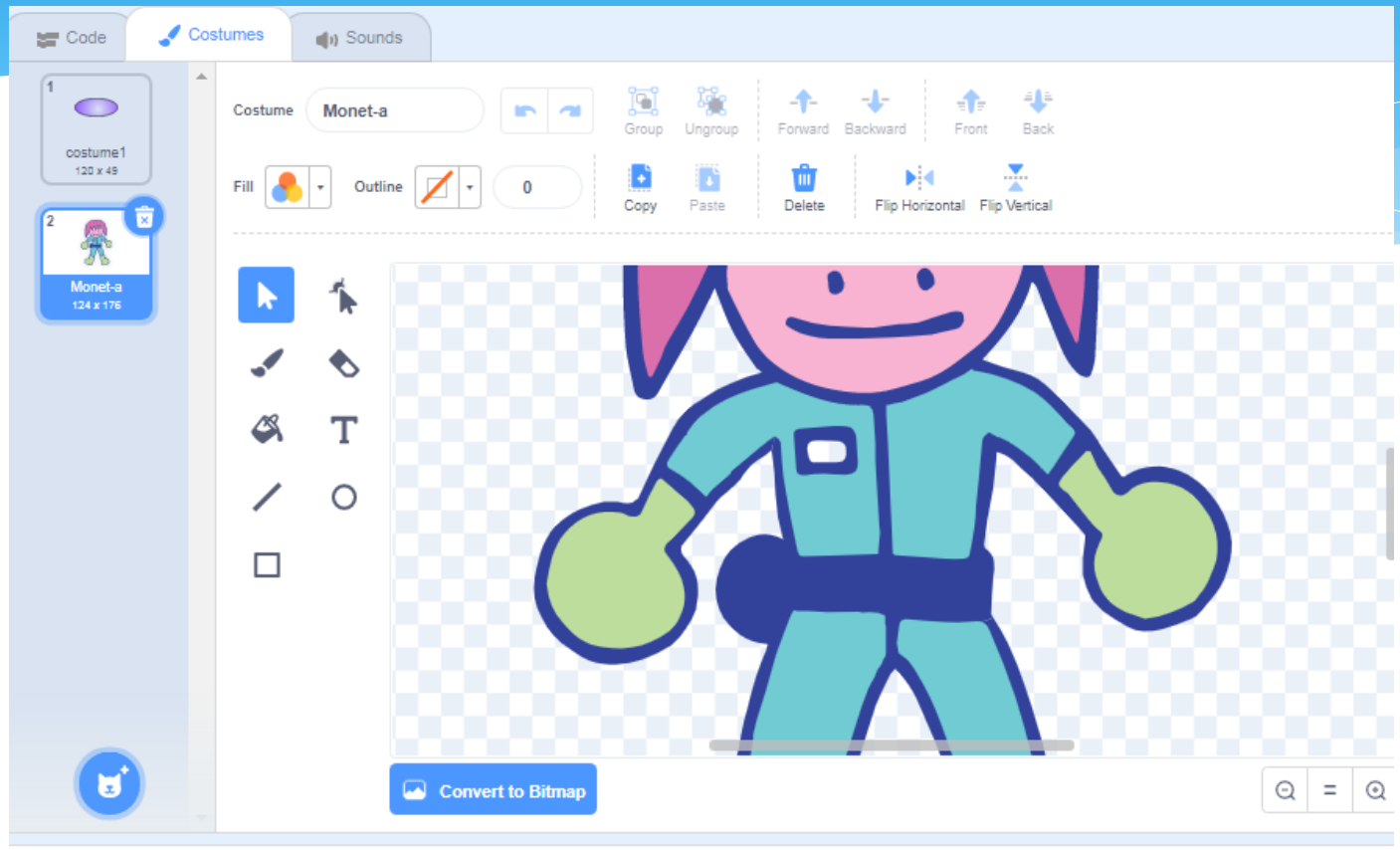
With this, we 'separate' the game preparation from the game play.

Now, over to the MOLE

* Let's think, what does the mole *really* do?
  * We have a mole in every pit.
* How do I get a mole inside each of the pits?
  * One answer is: Just like the pit, we can take a single mole and clone it. And since the 'x-' and 'y-' location of the pits is fixed (and known), we can in fact use the same code.
* We can do this by 'duplicating' the MOLE code.
  * Recall, duplicating sprites is a 'fast' way to get all the code to be carried over from one sprite to another.
* Once the code is duplicated, we have to only update the costumes.

Let's duplicate the PIT

Update the costume

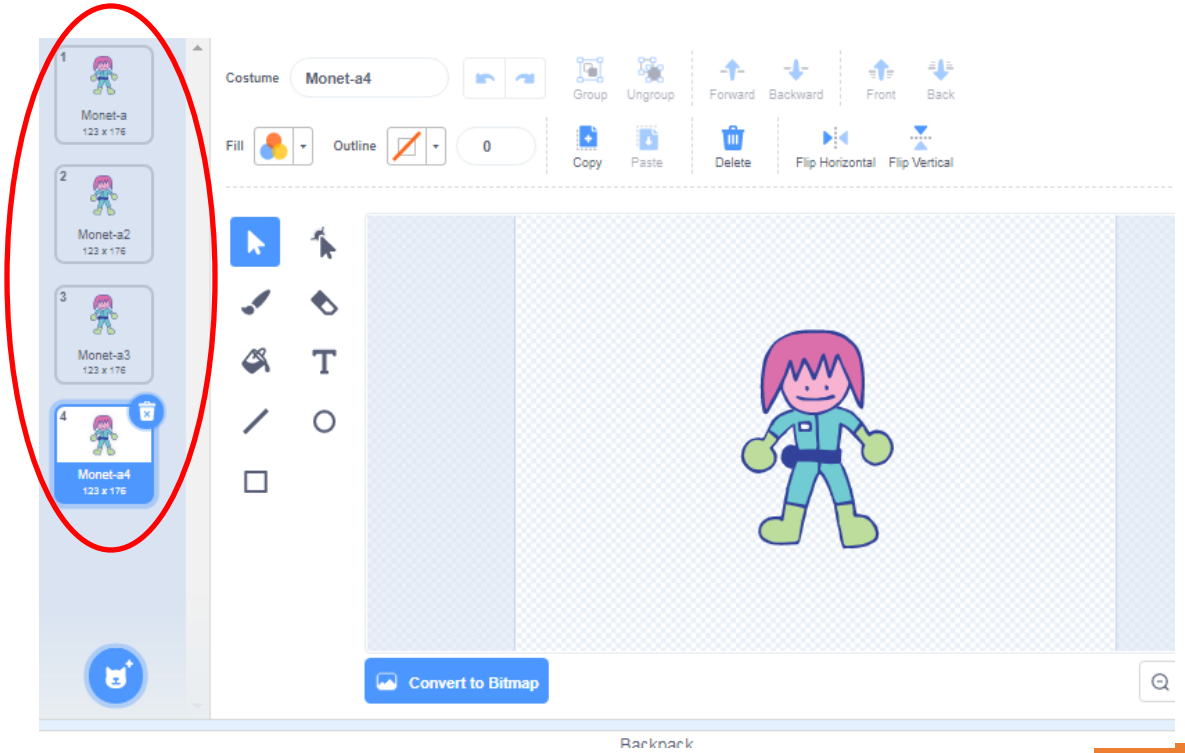


We will use COSTUMES to create the effect of MONET going up and down.

New Costumes to Monet!

* First, duplicate MONET's costume 4 times.

4
costumes



Now, edit each costume

* Costume 1 (Monet-a) – No change
* Costume 2 (Monet-a2) – Remove Lower Legs, *move the costume a bit down.*
* Costume 3 (Monet-a3) -- Remove Legs totally, *move the costume a bit more down.*
* Costume 3 (Monet-a4) -- Remove body, *move the costume even lower.*

* See next slide

Monet Costumes at a glance

Monet-a Monet-a2 Monet-a3 Monet-a4

Notice that the costumes are giving a feeling of ‘going down’. This way we can use the costumes to create the feeling that the sprites goes up an down, when in reality it stays at the exact same position.

Monet's Placement

* Remember, MONET has to go the exact same locations as the PIT. Hence, we can create its clones in the exact same way as the PIT.

Start this when the message 'GRID READY' is received.

Adjust the 'y' a little bit. (May need a bit of fine-tuning)

Monet Going up and down

When MONET starts as a clone …

Wait for a random amount of time

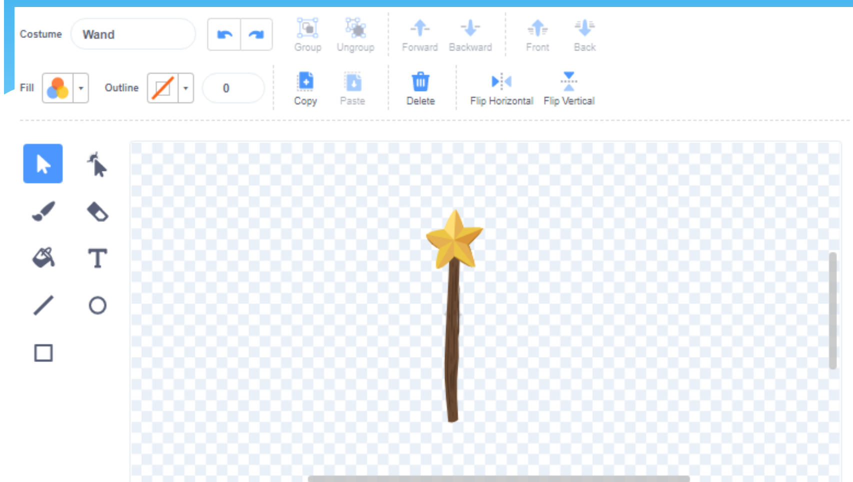
Show and take costume ‘Monet-a’ (The ‘complete’ Monet)

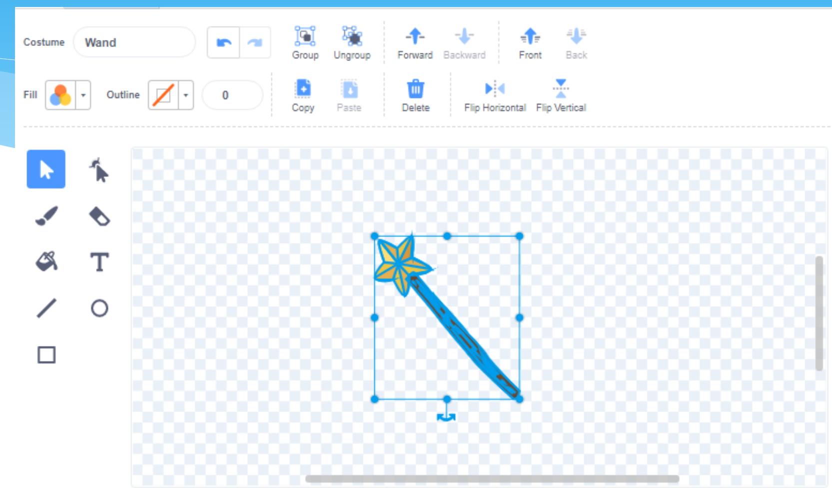
By changing costume, create a feeling that the Monet is going back down into the pit

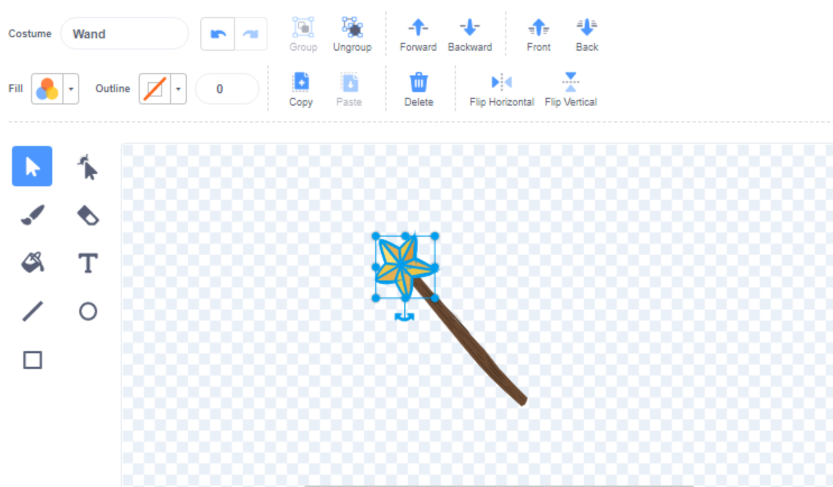
Now the Hammer

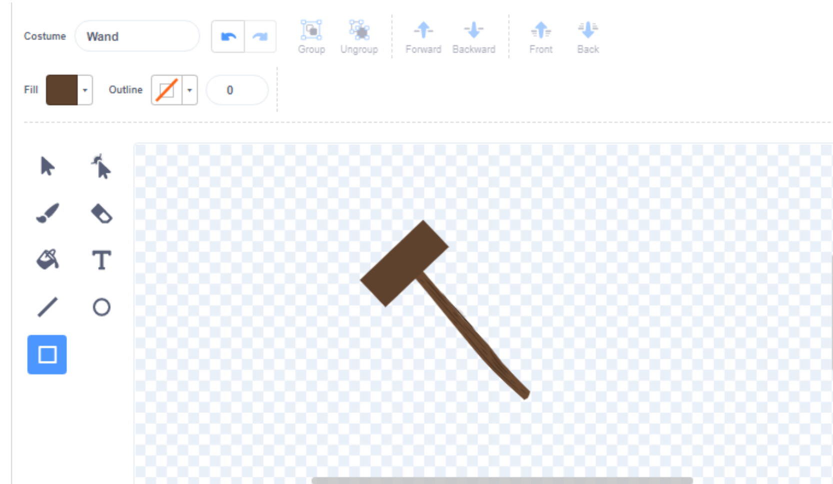
* You can create a new sprite and give it a hammer costume.
* As a simple hammer, we started with wand and replaced the ‘star’ with a rectangle.

Creating a Hammer Costume



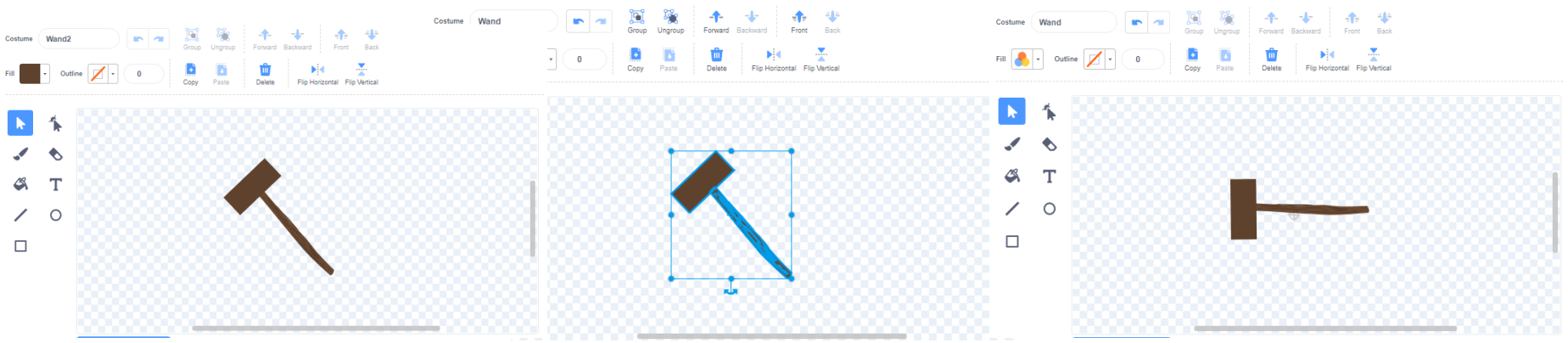




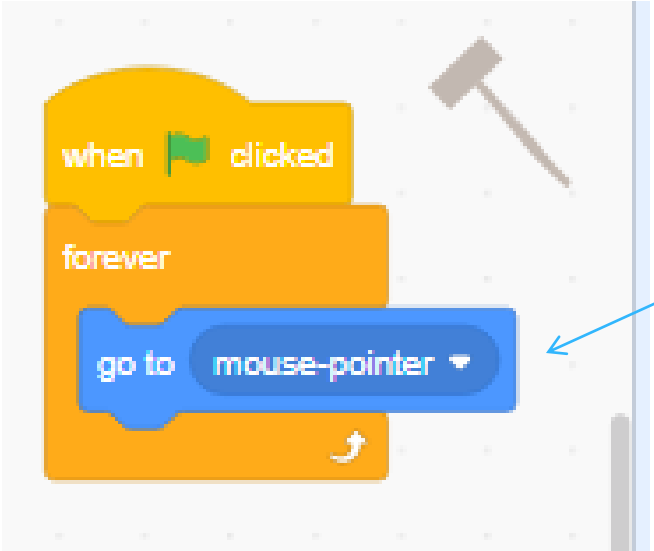


Creating a second Hammer costume

* Note: It is not a MUST to create a new costume here; we can instead use TURNING blocks here to create this effect.

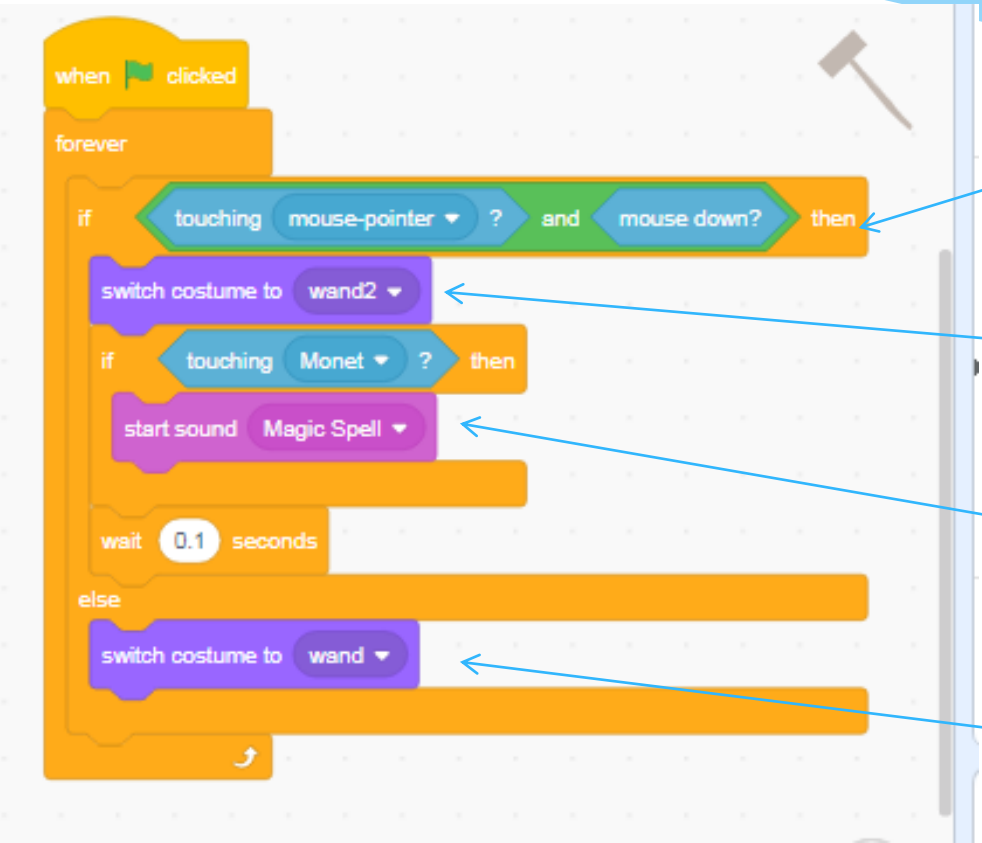


Making HAMMER move



Forever, go to the mouse pointer

Making HAMMER whack



If the MOUSE is CLICKED

Take on the next costume

If touching 'MONET', play a small sound. Tells that you have whacked the MONET

Go back to the regular costume

With this you are all set!

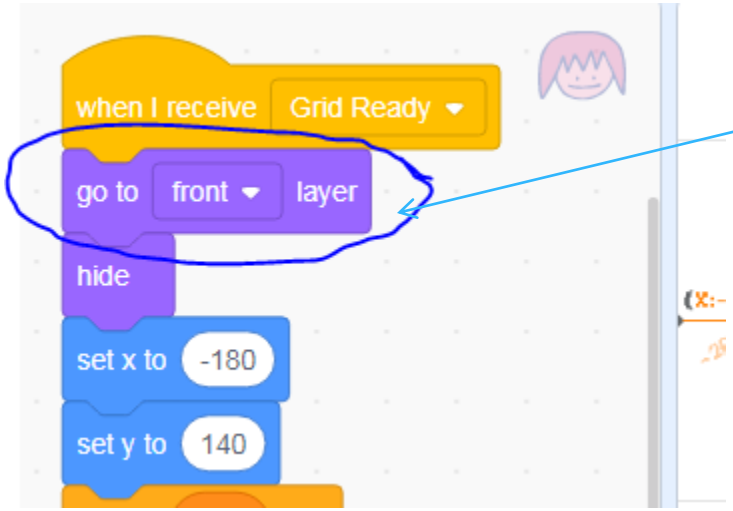
* ‘Whack-a-mole’ is a activity that allows us practice on concepts of cloning, broadcasting. We also learnt here the nested loop.
* There are many variations you can add, as we highlight in the extra innings.
* Try these and more in your independent activity.

Extra Innings

Food for thought!

* Why are we not DELETING the clones?
  * In this game, we know exactly how *many* clones we are creating.
  * Hence, we can choose to not delete the clones. They will anyway get deleted once the game is stopped.
  * This is unlike Shooting, Color wheel and Chrome Dino games where we do not know how many clones will get generated and hence keep deleting them as the game proceeds.

Making sure MONET is 'above' the Pits



Sometimes students may face issues with MONET – It may go behind the pits.

We want to make sure that MONET appears on TOP of the pits.

We can ensure this by using LAYERS and sending MONET to the TOP layer.

When two or more sprites overlap, we can determine their order using this concept of layers.
The idea is very simple, the sprite on a higher layer will go on top of the sprite on a lower layer.

Ideas for adding variety (And bonus)

* You can get attractive costumes for PITs and HAMMER and an attractive backdrop.
* You can give the MOLE a new costume after it is whacked.
* You can try placing the pits not in a grid, but say in a circular fashion.
* You can have exactly 1 MOLE that appears randomly on any of the PITS.
  * For this, you do not need to clone the MOLE, but send it to different x- and y- locations.
* You can do the game in 2 levels – in the first level, the grid is only 1 row, 3 columns. And then in the second level we have 3 rows, 3 columns.
  * Extensive usage of broadcasts. Plan very carefully.
* You can try creating more than 1 MOLE. Allocate them PITS at random. If both moles end up in the same pit, one of them will need to be deleted.