Sprite tests your memory!

Vineet Srivastava

In this lesson, we will …

* Use lists to create a simple memory game!
* This game can act as a basic memory game which can be extended in many ways.

Rules of our memory game

* We have a background of buildings, each with their own numbers.
* Our hero, Noor, goes to a few of these buildings.
* After visiting these buildings, she asks us a few questions on where all she has been.
* We have to answer these questions – these test how attentive we have been.

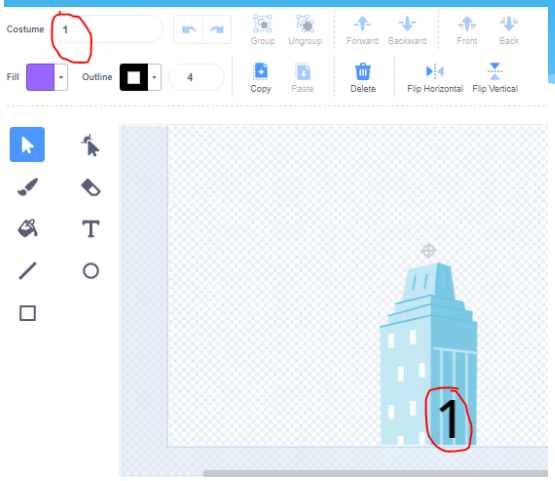
Why are lists useful here?

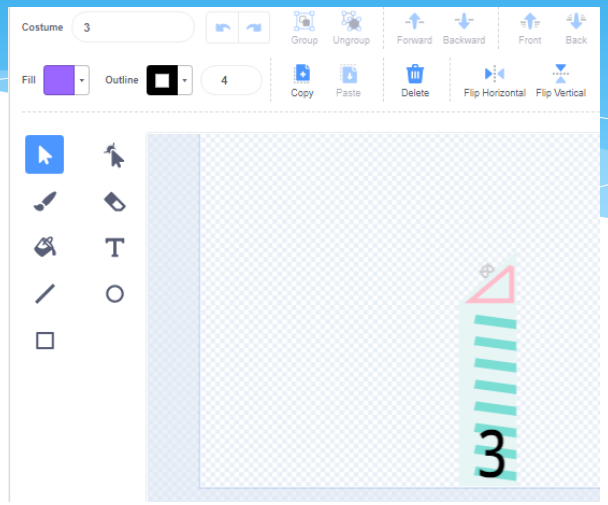
* In this game, we will use lists to keep track of the
  * different building costumes
  * The locations of the different buildings
  * The buildings that Noor visits.
* In this process, we will use several functions related to Lists.

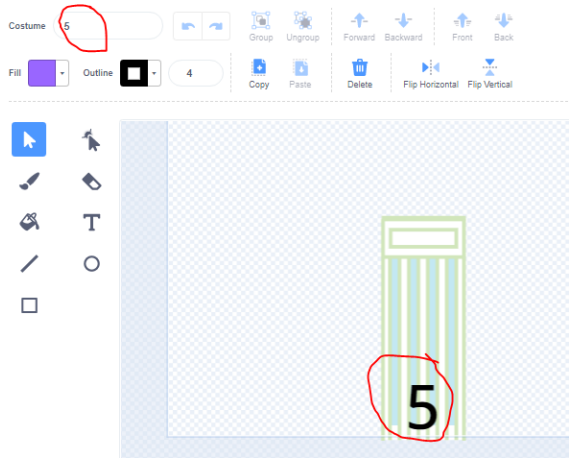
Building Costumes

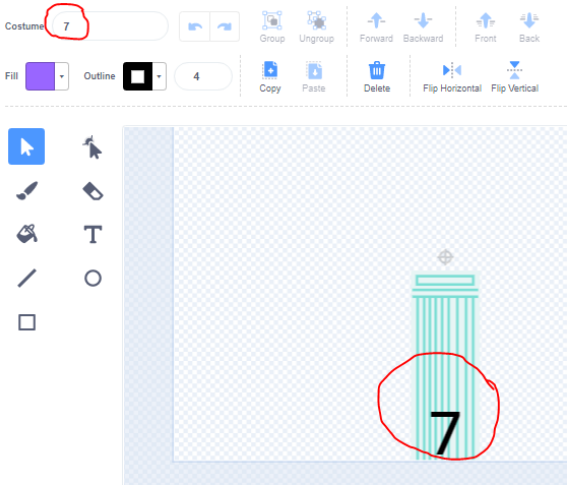
* In this game, we will use buildings with numbers. We have done this by editing the building costumes and adding text on these.
* We have used a sprite called ‘BUILDINGS’
* We have retained only those costumes that are somewhat narrow.
* Thereafter we have added a ‘TEXT’ on each of the costume.
* For the ease of reference, we have also named the costumes as ‘1’, ‘2’, …, ‘7’, corresponding to the number mentioned on the building.

Some example Building Costumes



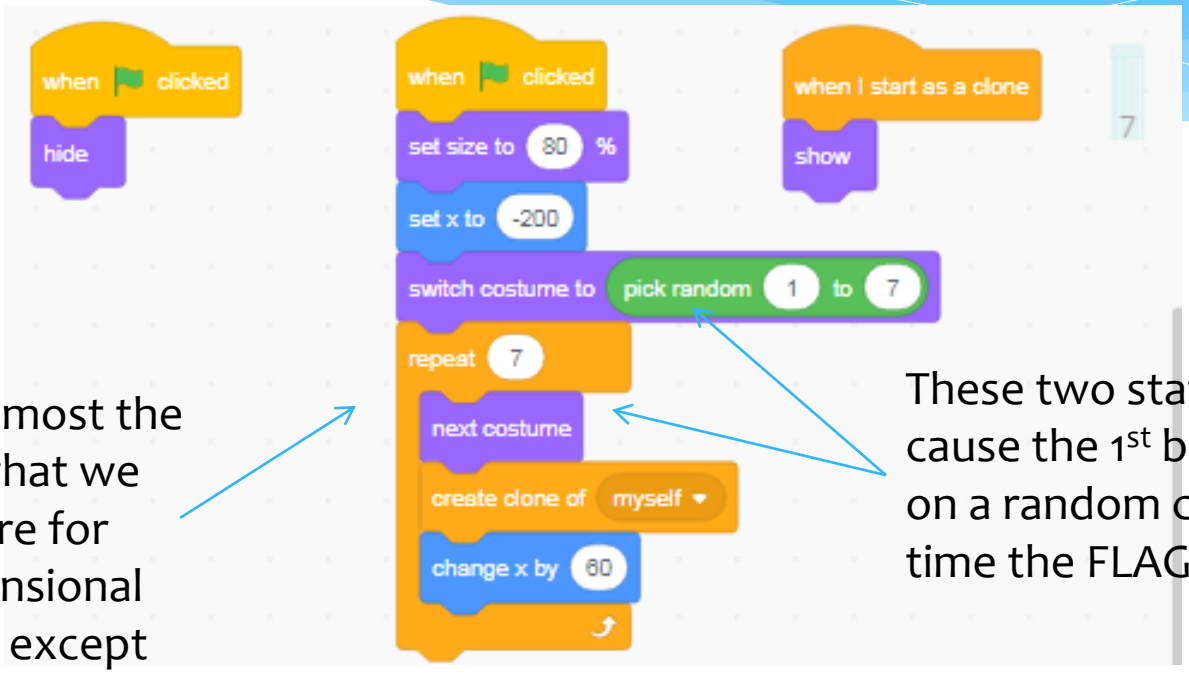






Showing here some example building costumes.

Cloning to create buildings



Notice, this is almost the same logic as what we have used before for creating 2-dimensional grids of sprites, except that here we are creating only 1 row of sprites.

These two statements will cause the 1st building to take on a random costume every time the FLAG is clicked.

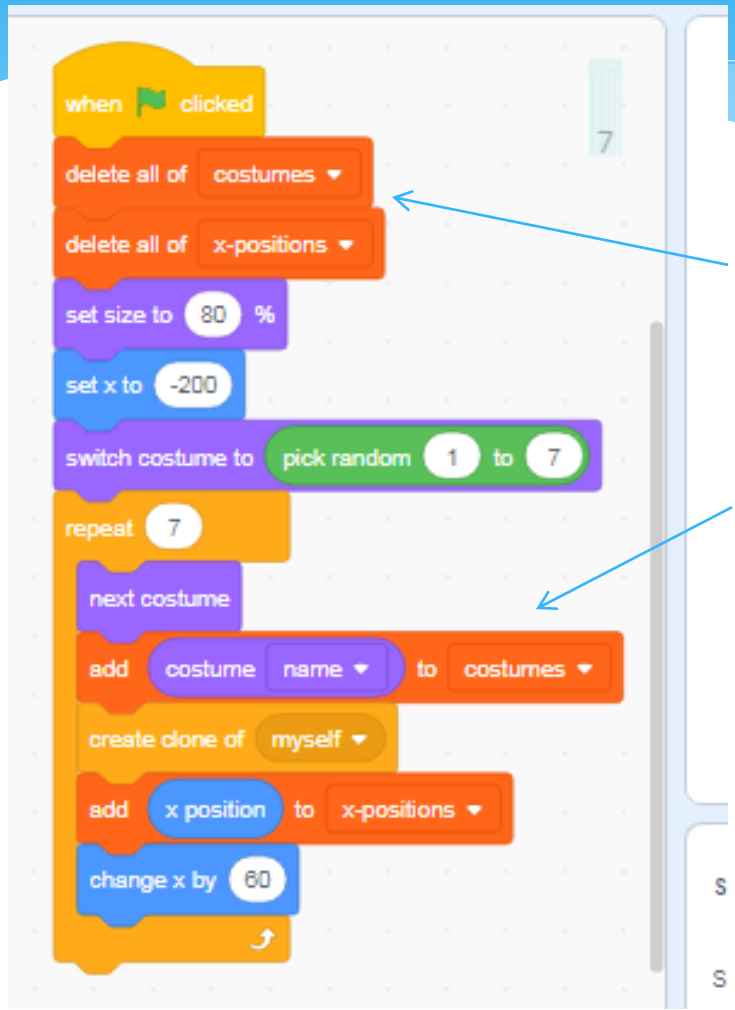
Notice that since we have named the costumes to be named as 1, 2, 3, …, 7, we can use a random number to set the costumes.

However, every time the flag is clicked, we do not know 'where' exactly a certain building costume has ended up.

Food for thought

* In our code, the starting costume is RANDOM between 1 and 7. However, since we are using 'NEXT COSTUME' after that, the costumes always appear in an order.
* How will you modify this code so that all the 7 buildings are always placed in a random order.
* (This is more than just picking up random numbers, since we have to avoid duplicates – same building appearing twice.)

Introduce two lists for Buildings



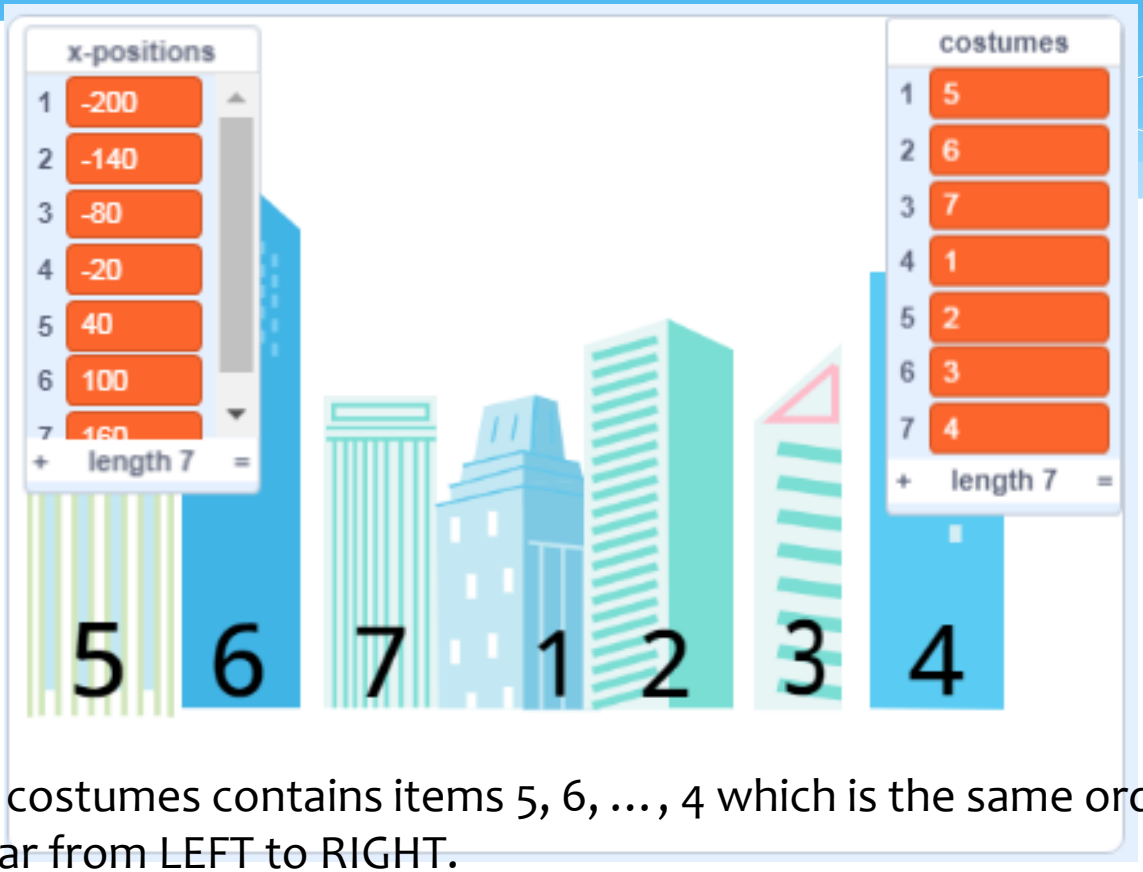
We create two lists : costumes and x-positions.

In the beginning, delete all the items of both these lists.

In the list called costumes, keep adding the name of the costume. Notice that the list costumes will contain the names of all costumes from LEFT to RIGHT.

In the list called x-positions, keep track of the x-location of the specific costume. Notice, this musy be done BEFORE the change x by 60.

Example



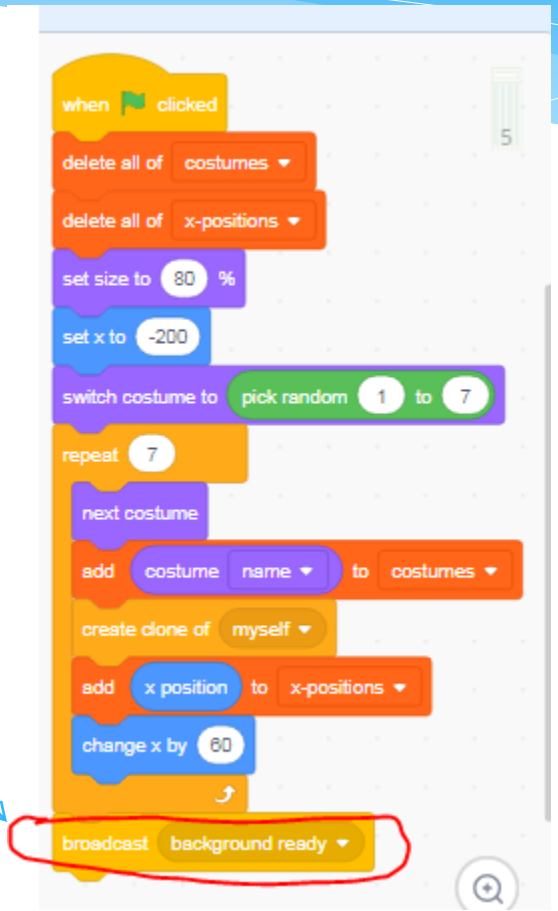
Notice, the list costumes contains items 5, 6, …, 4 which is the same order as the buildings appear from LEFT to RIGHT.
Also, the list called x-positions contains the x-location of each of these buildings.

Food for thought

* Why do we need these two lists:
  * The two lists that we have created can be thought of as ‘Building Names’ and their addresses.
  * Noor goes to the different buildings – as captured in the ‘costume names’ – and stored in the list called ‘Costumes’.
  * However, Noor needs to know the address of these buildings, in this case their x- positions, in order to be able to go there.

After the stage is set up, BROADCAST

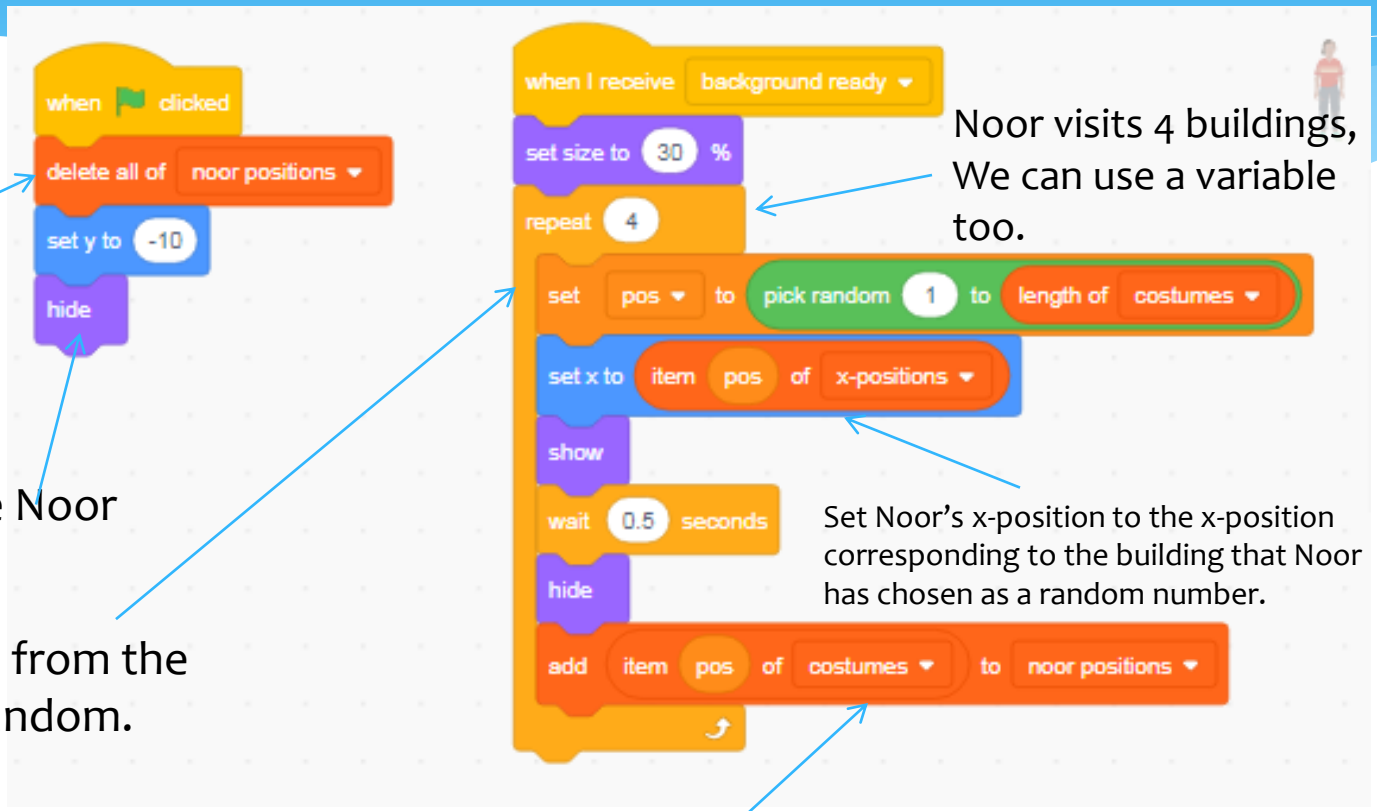
Notice this helps us to sequence events.



Code for Noor Movement

* We want Noor to be hidden in the beginning.
* Once it receives the BACKGROUND READY broadcast, it should visit 4 randomly chosen buildings.
* It should keep track of the names of the buildings it has visited in a list. We will call this list Noor Positions.

Code for Noor's Movement



Delete all items of 'Noor Positions' list in the beginning

Hide Noor

Noor visits 4 buildings, We can use a variable too.

Set Noor's x-position to the x-position corresponding to the building that Noor has chosen as a random number.

Pick up one costume from the list of costumes at random.

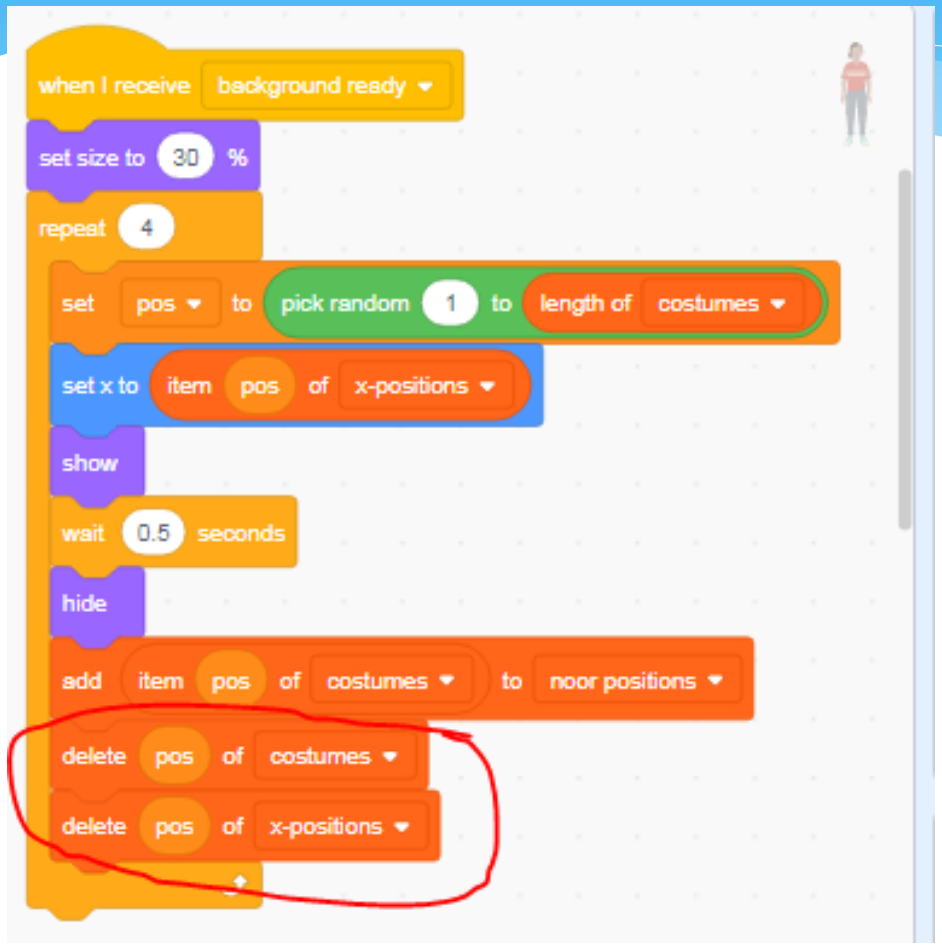
Notice, using 'length of COSTUMES' keeps the code flexible. (Rather than using 7.)

Add the chosen costume to the list called NOOR POSITIONS

Noor visiting same building more than once

* In our code right now, there is no mechanism that ensures that Noor does not visit the same building twice.
* This will happen, for example, if the same random number is picked up more than once.
* One method to avoid this could be to keep ‘deleting’ the buildings that Noor has once visited from the List costumes.
* Correspondingly, we will remove the item from the x-positions list too.

Avoiding repeat visits



Once Noor visits a certain building, we delete that item from the list costumes.

This ensures that Noor does not select the same building twice.

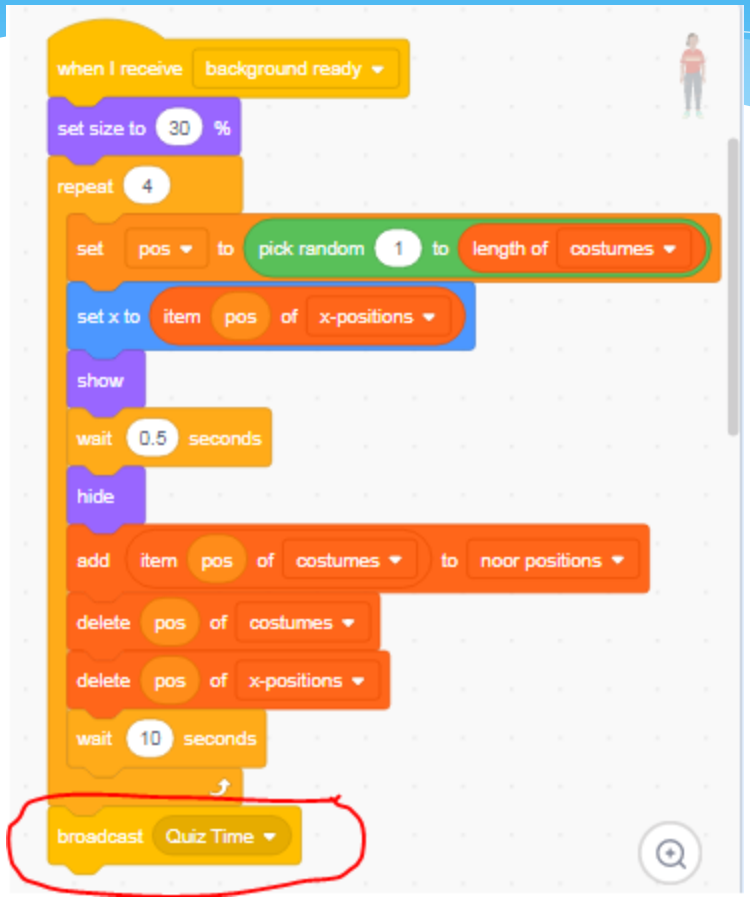
(See example on the next slide).

Avoiding repeat Visits (Example)

Noor went to Building 4. Notice, the Building 4 is deleted from the List costumes. This ensures that Noor will not visit Building 4 again.

Next, Noor went to Building 7. We delete 7 also from costumes and also the corresponding x-positions from x-positions. This ensures Noor does not visit 7 too again.

At the end of Noor's visits, BROADCAST

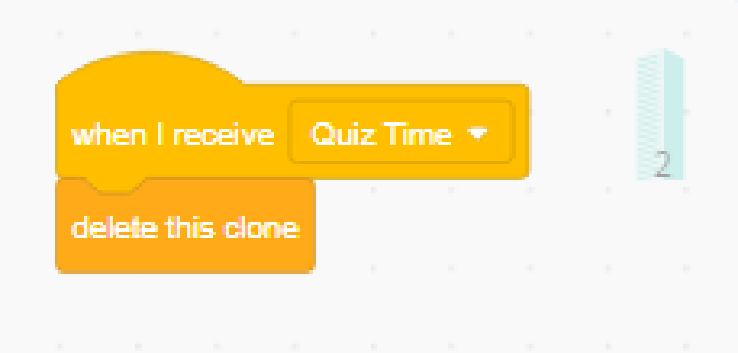


Once Noor has visited all the 4 buildings and kept a record in the list 'Noor Positions', broadcast a message QUIZ time.

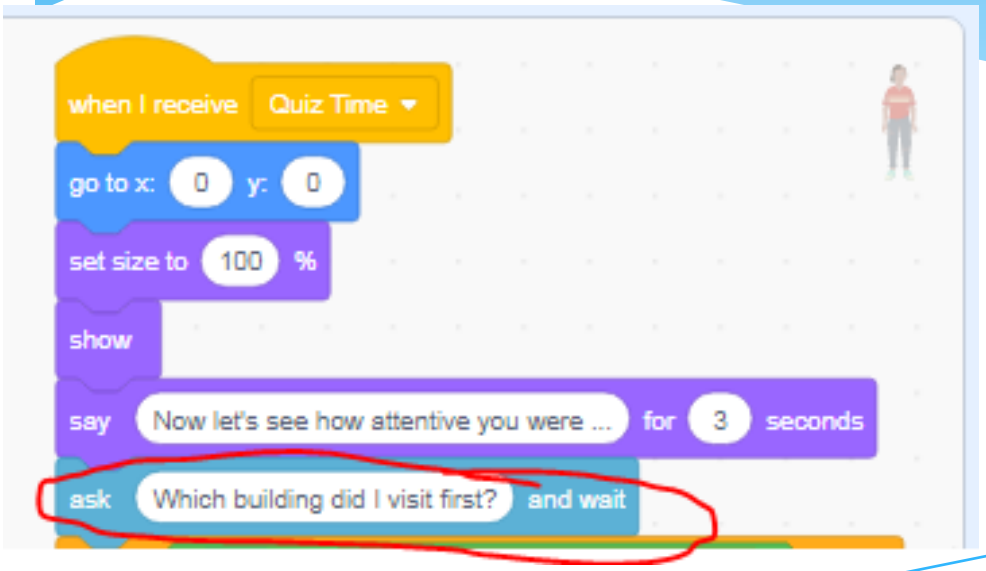
This ensures a sequence of events.

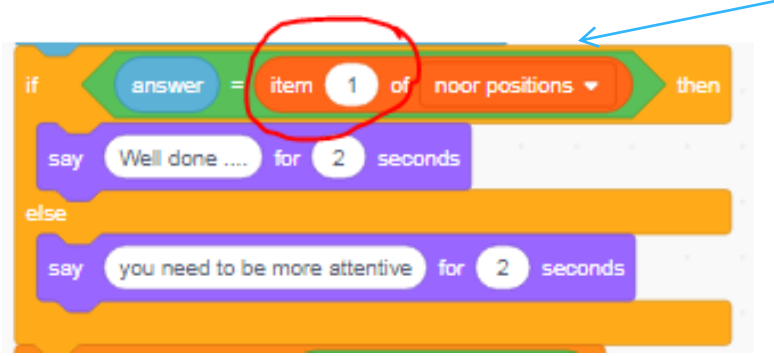
Delete the building clones

* Remove all the buildings when QUIZ starts.



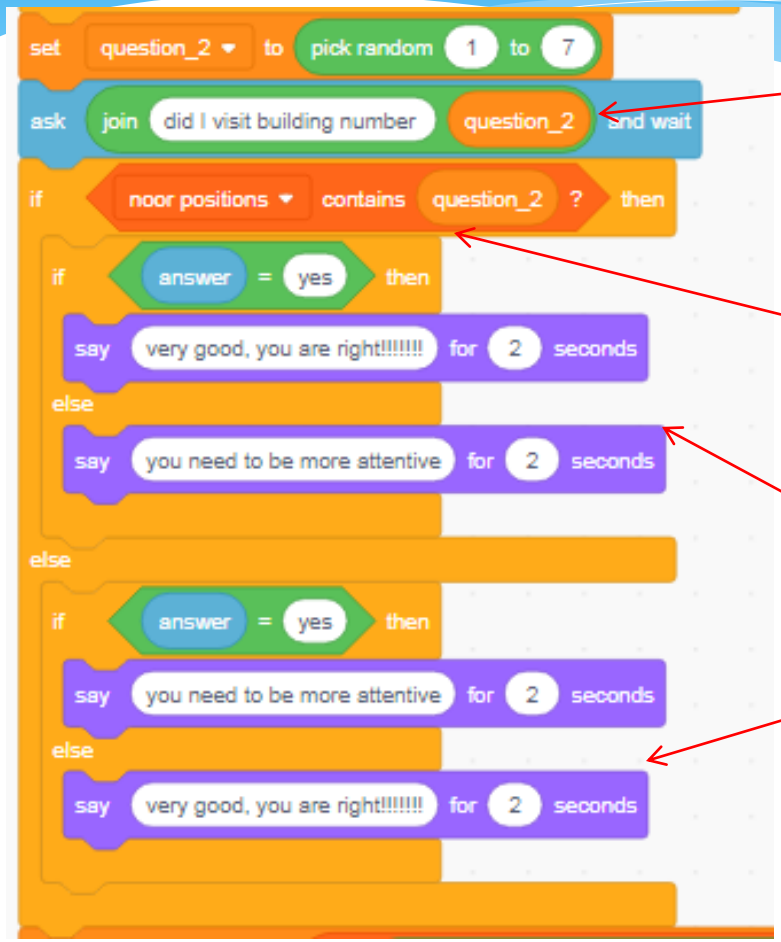
Noor's Question Number 1





Recall that item 1 of the list Noor Positions contained the first building that Noor visited.

Noor's Question Number 2



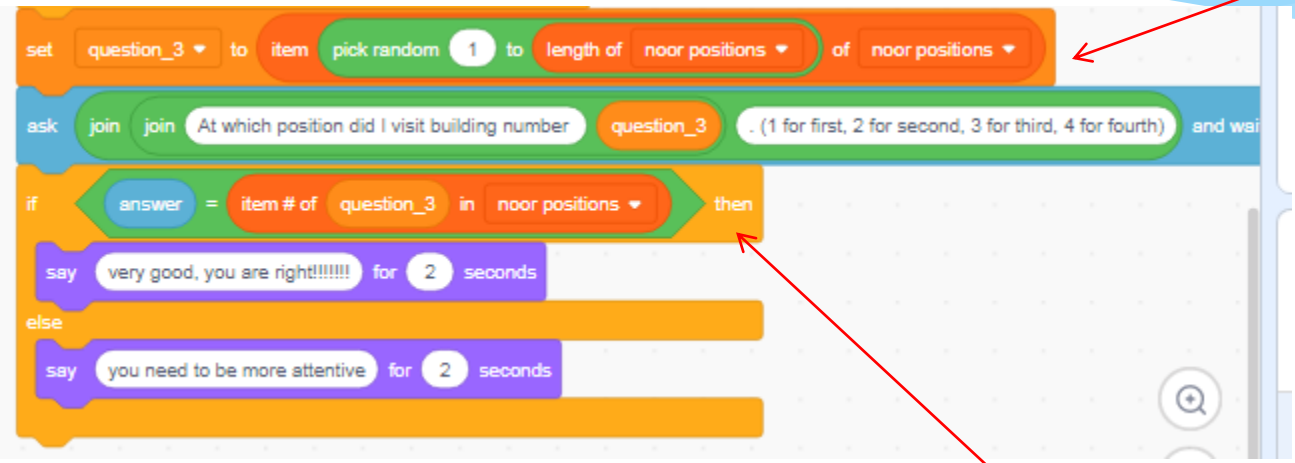
Set variable 'question_2' to a random number between 1 and 7.

If noor_positions contains the value of variable_2, it means that Noor visited that building.

Notice the reversal

Noor's Question Number 3

Set variable 'question_3' to a value randomly picked up from the list Noor positions.



Notice usage of item #. With this we can figure out the position at which Noor visited the specific building.

And you are all set

* We have built a very simple memory game.
* We have used this to illustrate the usage of lists.
* Go through the code carefully and see how list blocks are used. Also, try to appreciate how cumbersome this project would be 'without' using lists.
* Apply these ideas in your independent activity.

Extra Innings

Ideas to spice up the game!

* Can you change the game such that Noor visits 1 building and then asks a question, then visits 2 buildings and asks two question, then visits 3 buildings and asks 3 questions and so on.
  * To do this, you will have to use Broadcasts carefully and also populate the list Noor positions more times than the sequential flow here.
* Alternatively, can you make Noor go anywhere in a 2-dimensional grid of sprites (say like the pits in a whack-a-mole game) and create a memory game that way.