

Introduction

In this project you will make a Scratch game in which you need to find a sprite that is hidden among a huge crowd of other characters.

What you will make

You have to find the right sprite amongst all these characters before your time runs out.



What you will learn

- How to create custom blocks that have inputs
- How to use lists to store grid coordinates
- How to use loops to cycle over items in a list

What you will need

Hardware

- A computer capable of running Scratch 3

Software

- Scratch 3 (either [online](#) or [offline](#))

Step 1: Add costumes

✔ Activity Checklist



Open a new Scratch project.

Online: open a new online Scratch project at rpf.io/scratch-new.

Offline: open a new project in the offline editor.

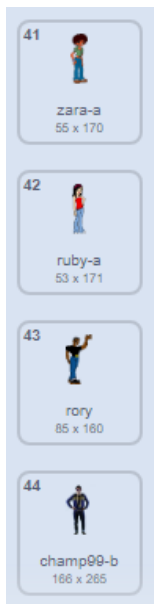
If you need to download and install the Scratch offline editor, you can find it at rpf.io/scratchoff.



Add some more costumes to the cat sprite. You need to add at least forty different costumes for your sprite.

It's best to select costumes from the **People** section, but if you want to, you can choose costumes from other sections as well.

Once you have your costumes, you can delete the default cat costumes if you want to.



Step 2: Create a grid

You are going to create a grid of stamped costumes:



To do this you need to know the **x** and **y** coordinates of where each stamp should be placed.

✓ Activity Checklist

- First, create a new block called **generate positions**. The block needs to have two 'number input' parameters. Call the two parameters **rows** and **columns**. The values of these parameters will decide how many rows and columns your grid has.

```
define generate positions rows columns
```

- Create two lists, and call one of them **x_positions** and the other **y_positions**. These lists are for storing the **x** and **y** coordinates for the stamps.
- Inside your **generate positions** block, add blocks to delete all the items from both lists, so that each time the game starts, the lists are empty.

```
define generate positions rows columns
```

```
delete all of y_positions
```

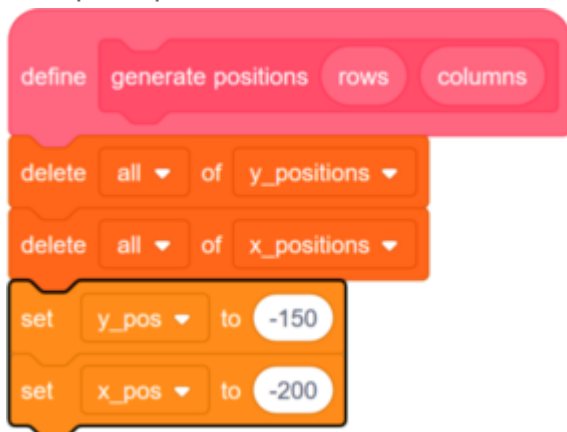
```
delete all of x_positions
```

- Next, create two variables, and call one of them **x_pos** and the other **y_pos**.

The **x_positions** list should contain ten numbers in total, and these should start at **-200** and go up to **200**.

For now, the `y_positions` list can just contain the number `-150` ten times, so that the grid only has one row.

- Start by adding code to the `generate positions` block to set the `y_pos` variable to `-150` and the `x_pos` variable to `-200`. This is the location of the first stamped sprite.



```
define generate positions rows columns
  delete all of y_positions
  delete all of x_positions
  set y_pos to -150
  set x_pos to -200
```

- Next, add a `repeat` loop to put coordinates into the lists. The `repeat` loop should run once for every column you want the grid to have. The `generate positions` block takes `columns` as an input, so you can use `columns` for the `repeat` loop.



```
define generate positions rows columns
  delete all of y_positions
  delete all of x_positions
  set y_pos to -150
  set x_pos to -200
  repeat columns
    [ ]
```

Within the `repeat` loop, add the values of `x_pos` and `y_pos` into the lists. Then you need to increase the value of `x_pos` by a little. How much should the value of `x_pos` increase by?

This is how to figure it out:

- `x_pos` starts out with the value `-200`
- The final time the loop `repeat` runs, `x_pos` should reach the value `200`
- That's a total increase of `400`
- The first `x_pos` value is for the first column on the grid, and how many columns there are is determined by the `columns` input

So after the first `x_pos` value is added, each time around the loop, the value of `x_pos` should increase by $400 / (\text{columns} - 1)$

- Add in the code that will add all the `x_pos` and `y_pos` values into the `x_positions` and `y_positions` lists.

Here is the completed script for the `generate positions` block:

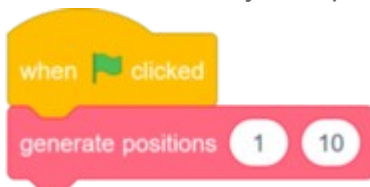


Step 3: Test the script

✔ Activity Checklist

- To test the script, you need to **call** the custom block and provide it with the number of `columns` you want in your grid.

Add this code to your sprite:



- Now click on the green flag to run your code. You should see your two lists fill with values.

x_positions		y_positions	
1	-200	1	-150
2	-155.55...	2	-150
3	-111.111...	3	-150
4	-66.666...	4	-150
5	-22.222...	5	-150
6	22.2222...	6	-150
7	66.6666...	7	-150
+ length 10 =		+ length 10 =	

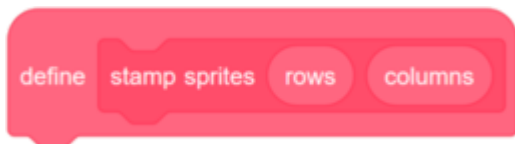
If your results don't look like this, then go back to the previous step, have a look at the hints, and try to fix your script.

Step 4: Stamp a row

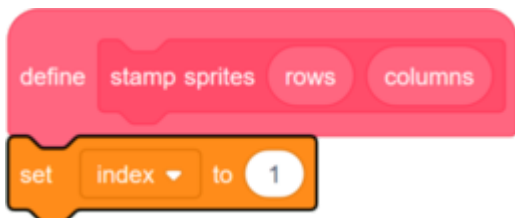
So far you have ten values in each of the two lists. Now stamp some costumes at the Stage coordinates stored in the lists.

✓ Activity Checklist

- Add the **Pen** extension to your project.
- Create a new block and call it `stamp sprites`. This block needs two number inputs named `row` and `columns` just like the other custom block.

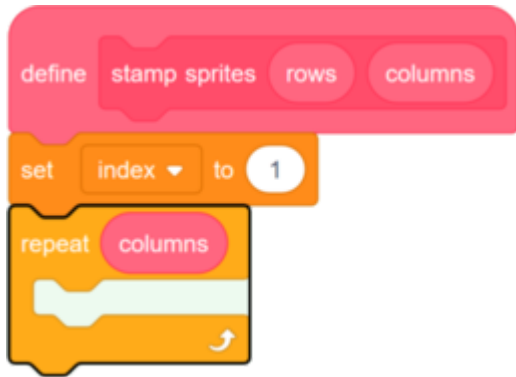


- Create a new variable called `index` with which to track the position in the lists that your program is reading. To begin with, set `index` to `1` to fetch the first item of each list.



- The `stamp sprites` block should stamp a sprite for each pair of coordinates in the list. To do this, the block needs a `repeat` loop that runs once for each

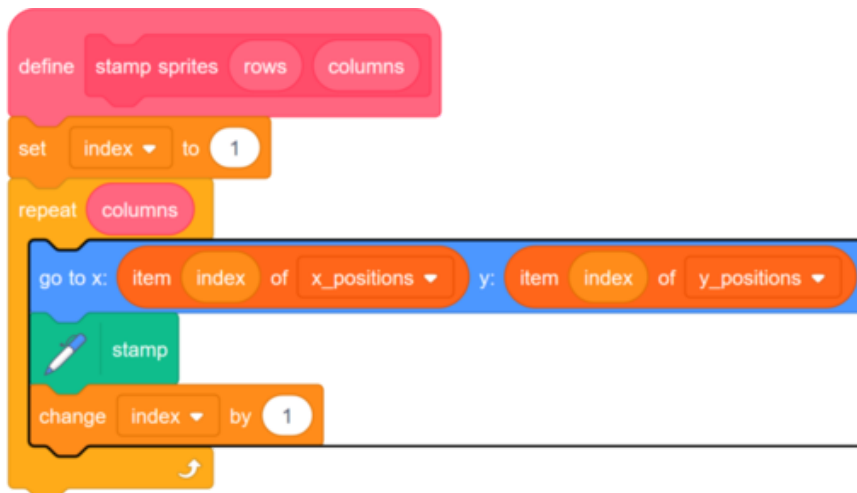
column.



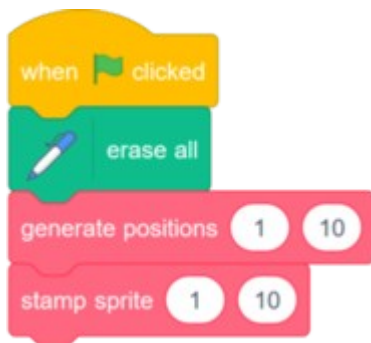
Within the `repeat` loop:

- Move the sprite to the `index` position in the `x_positions` and `y_positions` lists
- `Stamp` the sprite
- Change the `index` by `1`

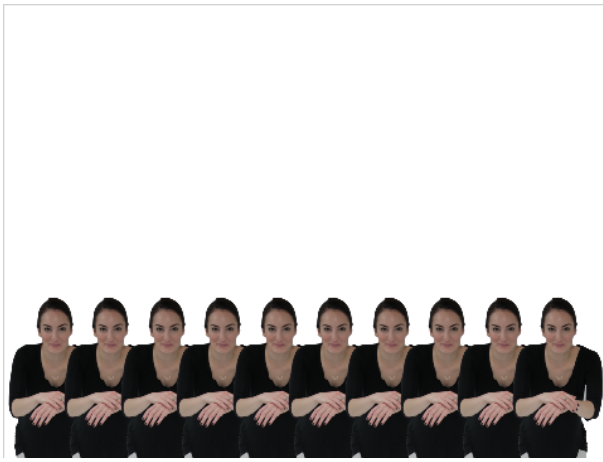
Here is the completed script for the `stamp sprites` block:



Add a `erase all` block below the `when flag clicked` block to clear the Stage each time the game starts. Then add the `stamp sprites` block at the bottom of the `when flag clicked` script so you can test your new code.



Click the green flag. You should see something like this, depending on the costumes your sprite has:



Step 5: Change the costumes

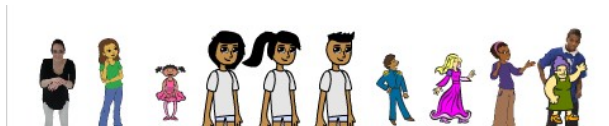
At the moment, your program stamps the same sprite costume over and over, and the size of the costume is too large.

✓ Activity Checklist

- Add code to the `stamp sprites` block to make the sprite a suitable size before the `repeat` loop starts. Add a block inside the loop to switch the `next costume` after the `stamp` block.

```
define stamp sprites rows columns
  set size to 40 %
  set index to 1
  repeat columns
    go to x: item index of x_positions y: item index of y_positions
    stamp
    next costume
    change index by 1
```

When you run the script now, you should see something like this:



Your program cycles through all the costumes in order. So that each costume does not show up in the same place every time the program runs, you should stamp the sprite in random places on the grid.

To do this, you need to follow this **algorithm**:

1. Repeat until the list is empty
2. Set `index` to a `random` number between `1` and the length of a list
3. Move the sprite as you did before
4. Delete the item at the `index` position from the `y_positions` list
5. Delete the item at the `index` position from the `x_positions` list

Add code to stamp the sprite in random places on the grid.

Remove the `set index to 1` from before the `repeat` loop.

Then within the loop, `set index to` a `random` number between `1` and the `length of x_positions`.

Then `delete` the item at the `index` from both the `x_positions` and `y_positions` lists.

This is what your code should look like:

```
define stamp sprites rows columns
  set size to 40 %
  set index = to 1
  repeat columns
    set index to pick random 1 to length of x_positions
    go to x: item index of x_positions y: item index of y_positions
    delete index of x_positions
    delete index of y_positions
    stamp
    next costume
    change index = by 1
```

Step 6: Add rows

Now that you have the code to create a single row of stamped costumes, you should add code to create more rows.

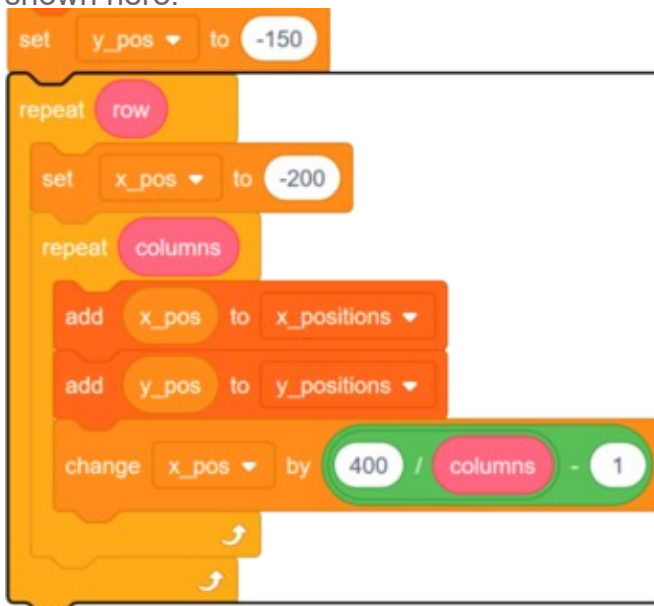
✓ Activity Checklist

- Go to your `generate positions` block.



```
define generate positions rows columns
  delete all of y_positions
  delete all of x_positions
  set y_pos to -150
  set x_pos to -200
  repeat columns
    add x_pos to x_positions
    add y_pos to y_positions
    change x_pos by 400 / columns - 1
```

- Add another `repeat` loop that runs the number of times you give to the `generate positions` block as the `rows` input. Place the `repeat` loop into your script as shown here:



```
set y_pos to -150
repeat row
  set x_pos to -200
  repeat columns
    add x_pos to x_positions
    add y_pos to y_positions
    change x_pos by 400 / columns - 1
```

Next you need to increase the value of `y_pos` each time the `repeat (rows)` loop runs.

You do this in a similar manner to how you increase the value of `x_pos` in the `repeat (columns)` loop.

- At the end of the code inside the `repeat (rows)` loop, `y_pos` should increase up to `150`, which is `300` away from its starting value of `-150`. This needs to happen for each row of stamps.

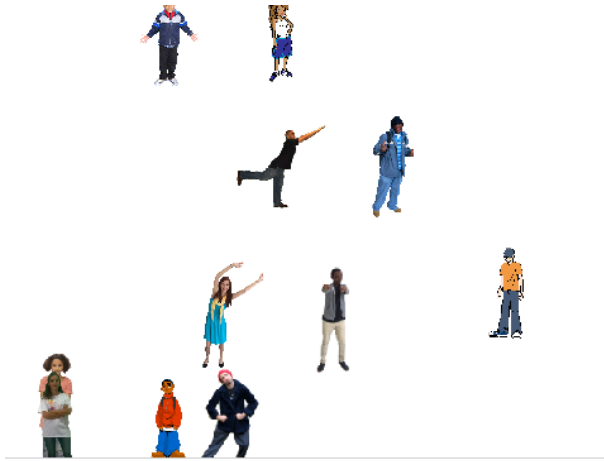
```
define generate positions rows columns
  delete all of y_positions
  delete all of x_positions
  set y_pos to -150
  repeat rows
    set x_pos to -200
    repeat columns
      add x_pos to x_positions
      add y_pos to y_positions
      change x_pos by 400 / columns - 1
    change y_pos by 300 / rows - 1
```

- Make sure you give the number of `rows` as an input to your blocks.

```
when clicked
  erase all
  generate positions 4 10
  stamp sprite 4 10
```



Run your code now.



You won't get a neat grid of stamps.

This is because, right now, the `stamp sprite` block only runs for the total number of columns.



Change your `stamp sprites` script so that it `repeats` enough times to stamp the complete grid of sprites.

Here's the completed `stamp sprites` script:

```

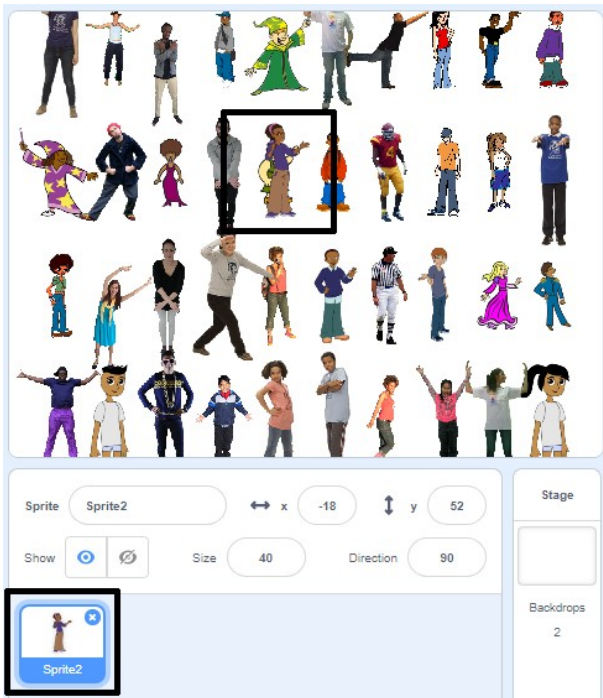
when clicked
  erase all
  generate positions 4 10
  stamp sprite 4 10

```



Step 7: Hide your sprite

Now it's time to hide your sprite among the crowd of stamps. At the moment the sprite overlaps one of the stamps.



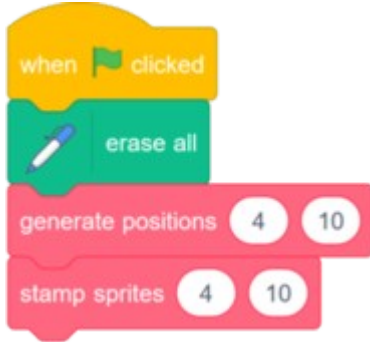
Activity Checklist

- So this doesn't happen, make your stamp loop run one time less: $(rows * columns) - 1$

```
define stamp sprites rows columns
set size to 40 %
repeat rows * columns - 1
  set index to pick random 1 to length of x_positions
  go to x: item index of x_positions y: item index of y_positions
  delete index of x_positions
  delete index of y_positions
  stamp
  next costume
```

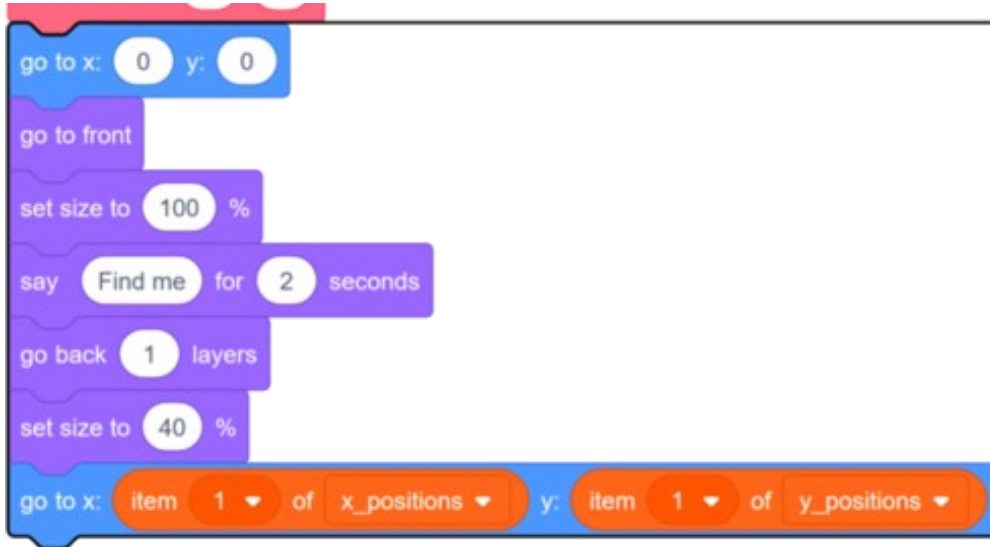
If you run the script now, you can see that your sprite still overlaps with a stamp and there is a hole in your grid. And in the `x_positions` and `y_positions` lists, there is one coordinate position left.

- To finish this part your game, go to the **when flag clicked** section of the scripts.



At the start of the game, the sprite should appear at a large size and say "Find me". Then the sprite should hide itself among the stamps in the empty space you have left for it.

Here is the what you need to add to your **when flag clicked** script:



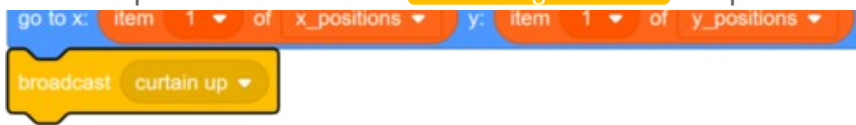
Step 8: Finish the game

✔ Activity Checklist

- To finish the game, search for and download an image of a stage curtain. Import this image as a sprite.
- Position the new curtain sprite at **x:0 y:0**, and then change its size so that it fills the screen. Make sure it is visible.



- Then, in the scripts for your character sprite, add a **broadcast** with the message 'curtain up' to the end of the **when flag clicked** script.



```
go to x: item 1 of x_positions y: item 1 of y_positions
broadcast curtain up
```

- When the curtain sprite receives the **broadcast**, the sprite needs to move upwards for 10 seconds so that it looks like the curtain is raised to reveal the stamps. Then the curtain should drop again, so the curtain sprite needs to move downwards.

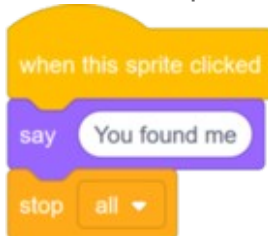
This is the completed script:



```
when I receive curtain up
  go to front
  wait 1 seconds
  glide 1 secs to x: 0 y: 300
  hide
  set timer to 10
  repeat 10
    wait 1 seconds
    change timer by -1
  show
  glide 1 secs to x: 0 y: 0
```

The very last part is to let the player know if they've won.

- In the scripts for the the character sprite, add code so that, when the sprite is clicked, the sprite says **You've found me**, and all the scripts in the game stop.



```
when this sprite clicked
  say You found me
  stop all
```

Challenge: improve your game

Here are some ideas for how to make your game more interesting:

- Can you alter your scripts so that you can use even more costumes?
- How about choosing a background and then making the sprites nearer the top of the screen appear smaller, so they seem further away?
- Can you make the game run for several rounds and provide you with a score based on how long it takes you to complete five rounds?