Raspberry Pi Projects

Lineup

Introduction

In this project you will be making a game using Scratch 2.0. The game will involve finding a stamped sprite on the stage, hidden amongst a huge group of other stamps.

What you will make

Hit the green flag and, once the curtain has been raised, try and find the clone before your time runs out.

What you will learn

This project covers elements from the following strands of the Raspberry Pi Digital Making Curriculum (http://rpf.io/curriculum):

- Apply basic programming constructs to solve a problem (https://www.raspberrypi.org/curriculum/programming/builder)

What you will need

Hardware

- An internet-connected computer

Software

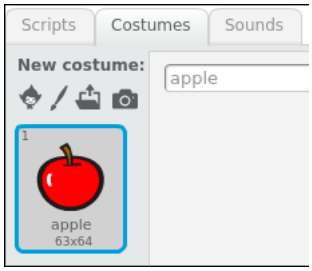
- Scratch 2.0 (online or offline)

Import your costumes

- Use the default cat sprite, and start by adding some more costumes.
- For this game you will need a lot of costumes! You'll want to import at least 40 from the Scratch library.
- It's best to select costumes from the **People** section, but the choice is yours.

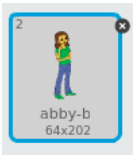
Adding new costumes in Scratch

- With your sprite selected, click on the Costumes tab

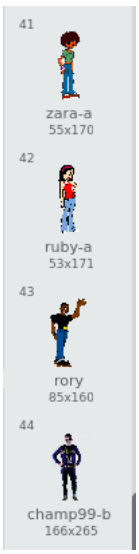


- You can choose one for the four options from the tab. From left to right they are:
  - Choose costume from library
  - Paint new costume
  - Upload costume from file
  - New costume from camera

- To import multiple costumes, you can hold down **Shift** while you are selecting them. Click on **OK** when you are finished.
- If you wish to delete the imported costume, select it and click on the small cross in the top right hand corner.



- Once you have your costumes, you can delete the default cat costumes if you like.



Making a grid

- You are going to produce a grid of stamped costumes, just like this one:

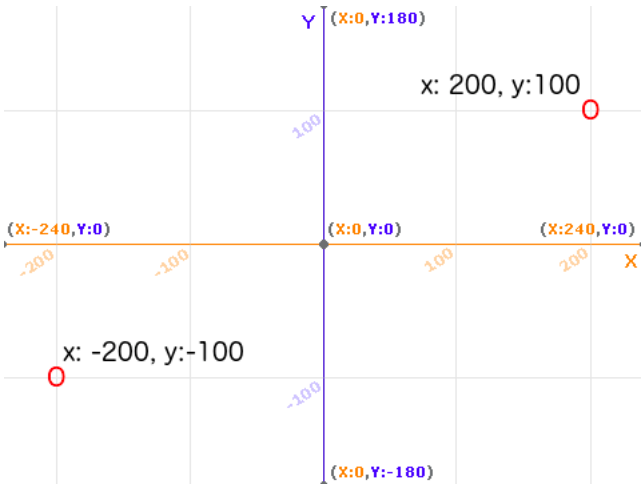
- To do this you will need to know the `x` and `y` coordinates of where each stamp is going to be placed. Have a look at the section below if you need a reminder about Scratch coordinates.

Scratch coordinates

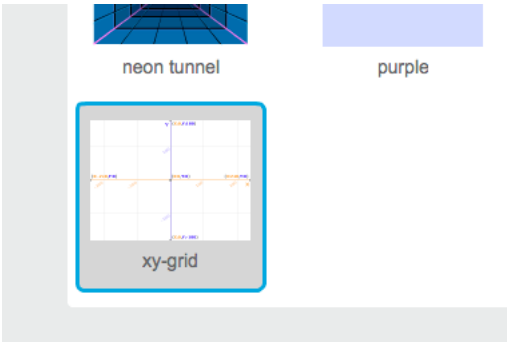
Scratch coordinates

- In Scratch, the coordinates `x:0, y:0` mark the central position on the Stage.

  A position like `x:-200, y:-100` is towards the bottom left on the Stage, and a position like `x:200, y:100` is near the top right.



- You can see this for yourself by adding the **xy-grid** backdrop to your project.



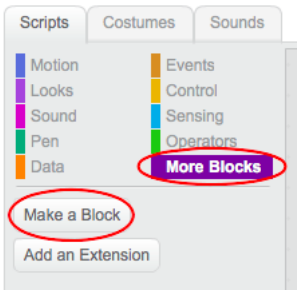
- To find out coordinates of a specific position, move your mouse pointer to it and check the readings below the bottom right corner of the Stage.

- To begin with you'll need to create a new block called `generate positions` for your sprite. The block will need to have two 'number input' parameters. Call the two parameters `rows` and `columns`. The values of these parameters will define how many rows and columns your grid will have.

Making a block

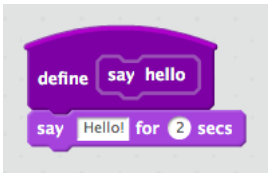
Making a block

- Click the **Scripts** tab, then on **More Blocks**, and then click **Make a Block**.



- Give your new block a name and then click **OK**.

- You will see a new `define` block. Attach code to this block.



- You can then use your new block just like any normal block.

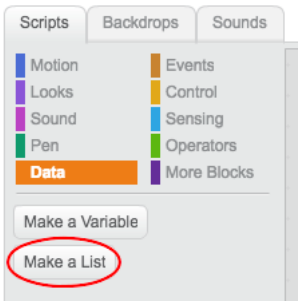
- The code attached to your new `define` block is run whenever the block is used.

- Next you will need to create two lists. One will be called `x_positions`, and the other will be called `y_positions`.

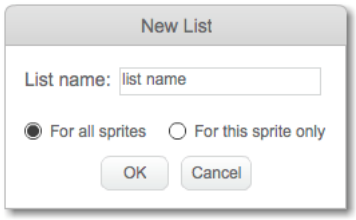
Make a list

Make a list

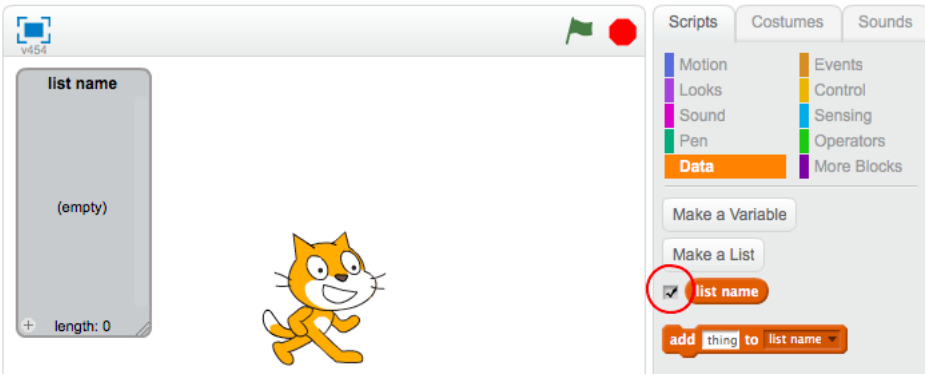
- Click on **Data** in the **Scripts** tab, then click on **Make a List**.



- Type in the name of your list. You can choose whether you would like your list to be available to all sprites, or to only a specific sprite. Press **OK**.



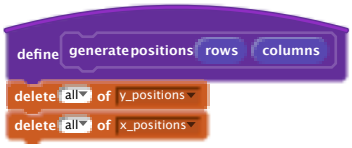
- Once you have created the list, it will be displayed on the stage, or you can untick the list in the **Scripts** tab to hide it.



- Click the + at the bottom of the list to add items, and click the cross next to an item to delete it.

- New blocks will appear and allow you to use your new list in your project.

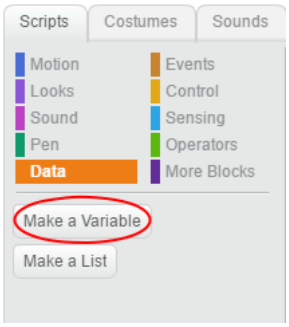
- Underneath your `generate positions` block, add blocks to delete all the items from both lists.



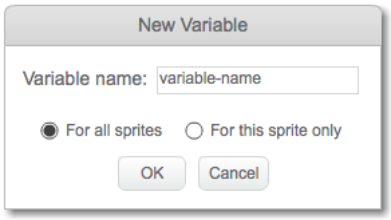
- Next you will need to create two new variables. Call these `x_pos` and `y_pos`.

Add a variable in Scratch

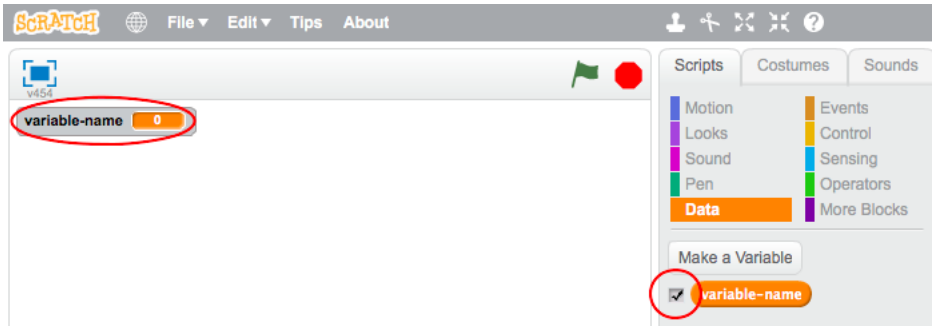
- Click on "Data" in the Scripts tab, then click on "Make a Variable"



- Type in the name of your variable. You can choose whether you would like your variable to be available to all sprites, or to only this sprite. Press OK.



- Once you have created the variable, it will be displayed on the stage, or you can untick the variable in the Scripts tab to hide it.



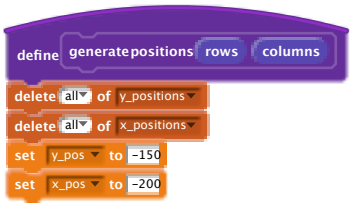
- New blocks will appear and allow you to change the value of the variable.

- Your two lists are going to hold all the `x` and `y` coordinates for the characters on the stage. If you look at the two lists side by side, as shown in the table below, you will see that the first index will correspond to the bottom left-hand corner of the stage. The next one will be a little bit to the right , and so on.

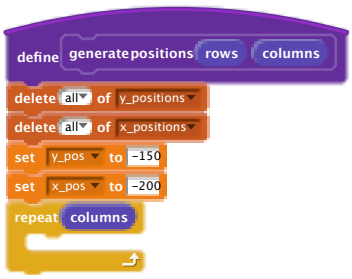
index	y_positions	x_positions
1	-150	-200
2	-150	-155
3	-150	-111
4	-150	-66
5	-150	-22
6	-150	22
7	-150	66
8	-150	111
9	-150	155
10	-150	200

- The table shows the coordinates of the first row of images. They could be added to the lists manually, but that would take a little too much time. It's easier to do it with a loop.

- Start by setting the `y_pos` and `x_pos` variables to `-150` and `-200`. This will be the location of the first image.

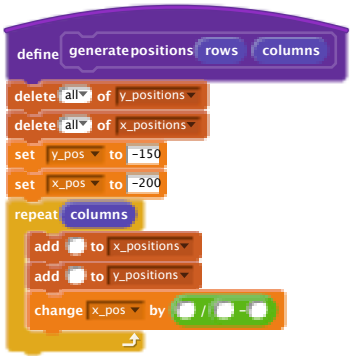


- Now you need a loop to add the first ten coordinates to the lists. So how many times should the loop repeat? In this case it is ten - one time for each column in the grid. As you've set up the block to take `columns` as a parameter, you can use this value.



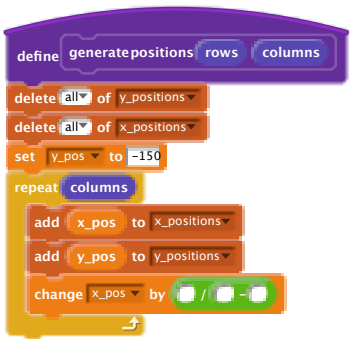
- Within the loop, you'll need to add the values of `x_pos` and `y_pos` into the lists. Then you'll need to increase the value of `x_pos` by a little. The first value you added is `-200` and the next is approximately `-155`, up until you reach `200`. So how can you calculate these increments? Well, as you want a total of ten values in the list, and the first is already set to `200`, you need an additional nine values. This is the value of `columns` minus one, so `columns - 1`. You want to go up to `200`, so you'll need to add an additional `400` to `x_pos`. This means you want to add `400 / (columns - 1)` each time the loop comes around.

- Here's some code to get you started, and you can use the hints below if you need more help.

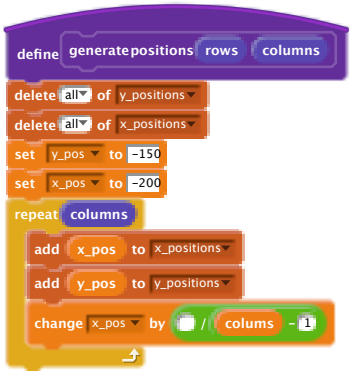


I need a hint

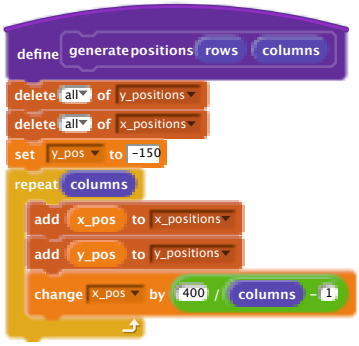
- Start by adding the variable values into the lists.



- Now use the `columns` parameter to calculate the number of repeats.



- Lastly you can add in `400` to the calculation, so it calculates the increase in `x_pos` each time.



Testing the script

- To test the script you'll need to **call** the block you have made. You also need to provide it with the number of `columns` you want in your grid.

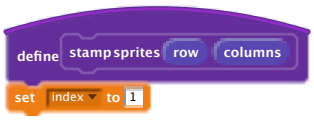
- Now click on the green flag to run your code. You should see your two lists fill with values.

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

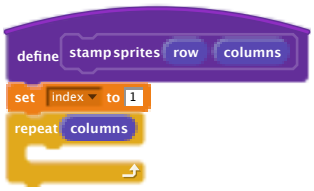
Stamping a row

- So far you have ten values in the two lists. Let's stamp some costumes at the stage positions given in the list.
- Create a new block and call it `stamp sprites`. It needs two parameters as well, both of which should be number inputs and named `row` and `columns` just like the last ones.

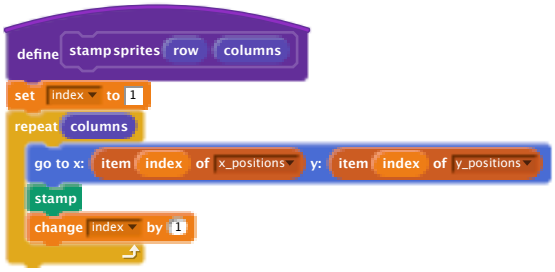
- Create a new variable called `index`. You can use this to track which position in the list you are reading. Set it to `1`.



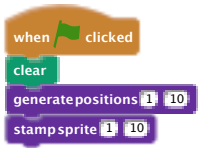
- Now you're going to stamp a sprite for each set of coordinates in the list. This will require a loop that will repeat once for each column.



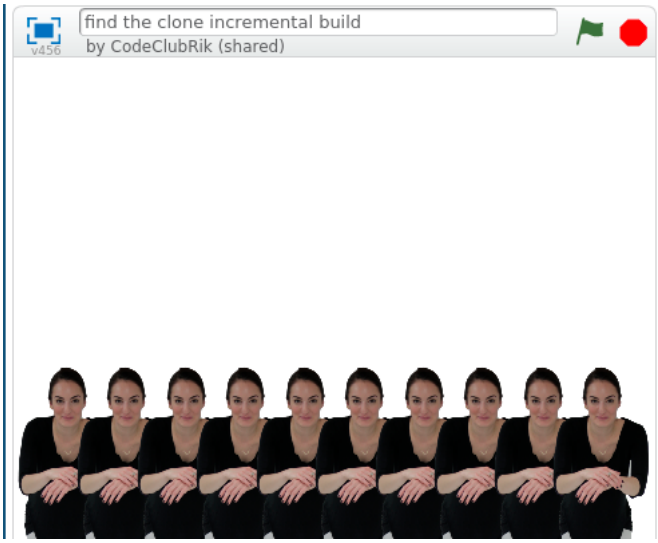
- Within the loop, move your sprite to the first position in the list, stamp it, then increase the `index` by 1.



- Next you need to call this block as well. You should also add a `clear` block to you starting script so that it clears the stage each time.

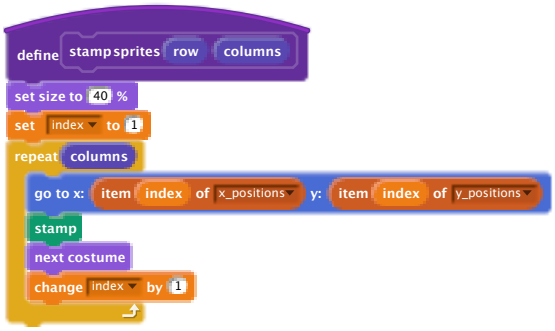


- When you click the green flag, you should see something like this:

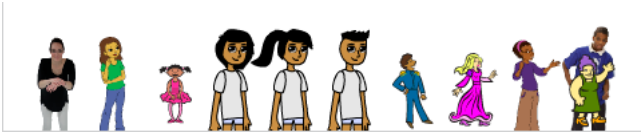


Changing the costumes

- Let's change the stamp each time and make it a more appropriate size.



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

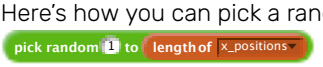


- At the moment, your program cycles through all the costumes in order. This isn't a problem, so long as you place the sprite in a random location each time.

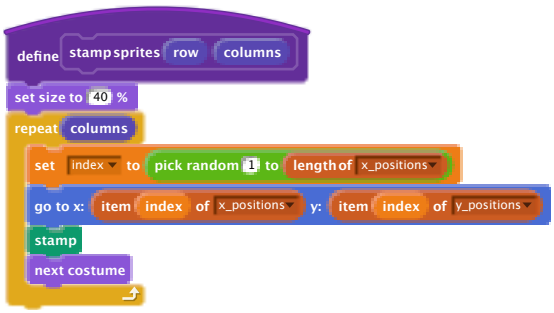
- To do this, you'll need to follow the following **algorithm**:
  - Set `index` to a random number between `1` and the length of a list
  - Move the sprite as you did before
  - Delete the `index` position from the `y_positions` list
  - Delete the `index` position from the `x_positions` list
- Have a go at doing this part yourself, and take a look at the hints if you need some help.

I need a hint

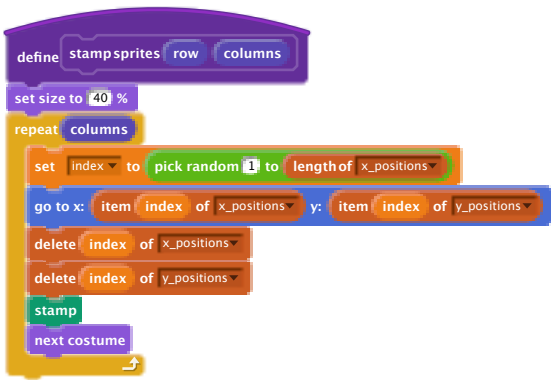
- Here's how you can pick a random number from within the list:



- Here's how to pick a random item from the list:



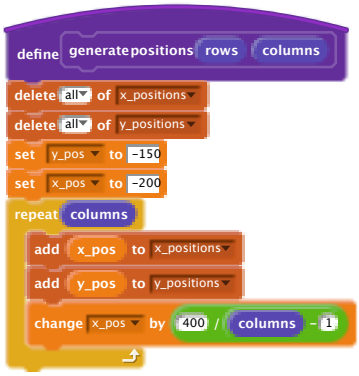
- Here is your completed script showing how to delete the items from the list:



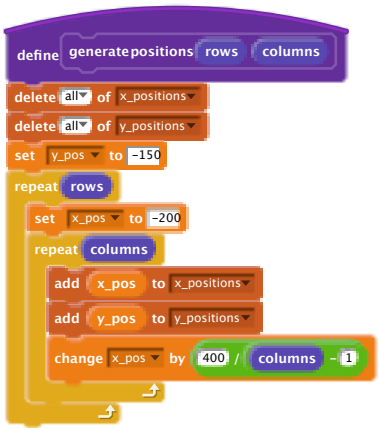
Adding rows

- Now that you've written the code to generate a single row, you need to add in more rows.

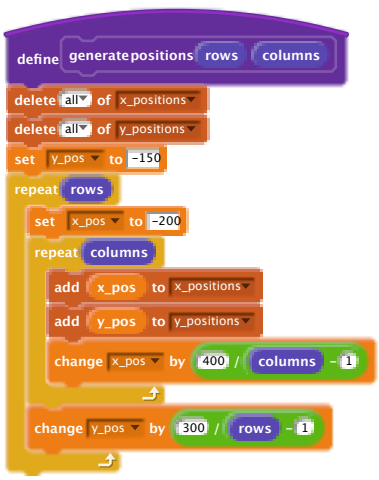
- Go back to your `generate positions` block.



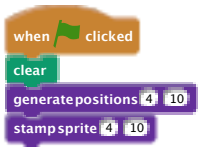
- Now you need another loop that will repeat the same number of times as the number of rows you need. Place it into your script as shown below.



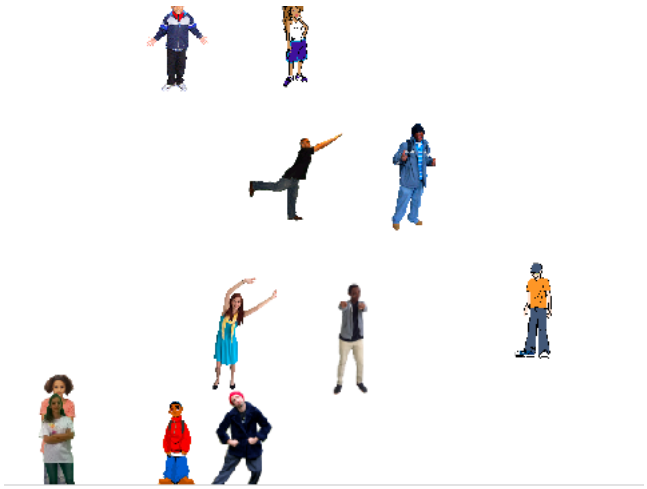
- Next you need to increase the value of `y_pos`. This will increase up to a maximum of `150`, which is `300` away from its starting value of `-150`. This needs to happen for each row your create.



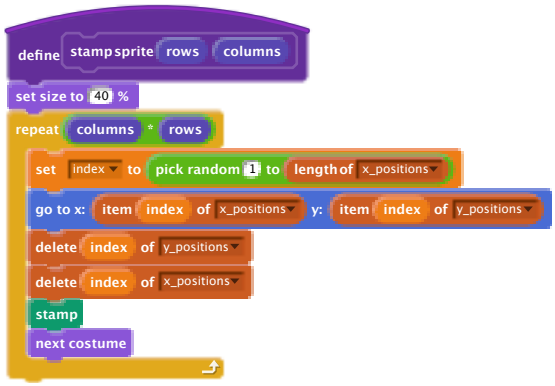
- Then you need to make sure you're passing the number of `rows` as a parameter to your blocks.



- If you run your code now, you won't get a neat grid of stamps.



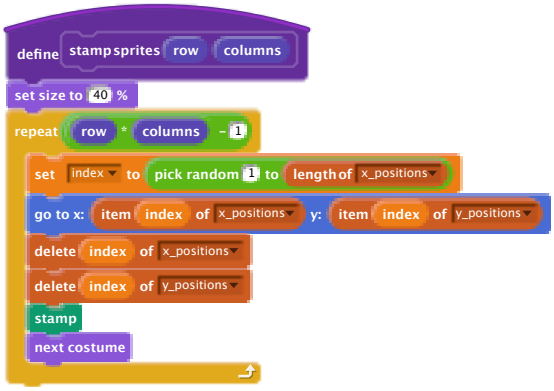
- This is because your `stamp sprite` block is only repeating for the total number of columns. It needs to repeat for the product of the number of columns and the number of rows (`columns * rows`).



Placing your sprite

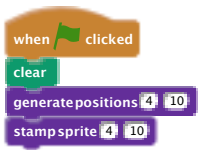
- Now it's time to position your sprite amongst the stamps. You'll notice at the moment that your sprite will overlap one of the stamps.

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



- If you run the script now, then your sprite still overlaps with a stamp, but there should be a hole in your grid of stamps. If you look have your `x_positions` and `y_positions` list, then you'll also see that there is one coordinate position left.

- To finish off this part the game, you'll need to continue the **green flag** section of the scripts.



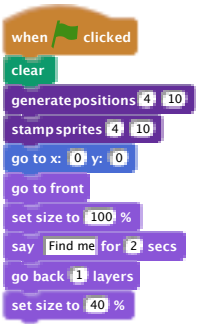
- Here's what it needs to do:
  - Send your sprite to `x:0 y:0`
  - Bring the sprite to the front and set its size to 100%
  - Say `Find me` for two seconds
  - Move back one layer
  - Set the sprite's size to 40%
  - Move to the last remaining position in the lists

- Here's an animation showing what should happen

- See if you can do this independently, and use the hints below if you need more help.

I need a hint

- The first part is fairly simple:



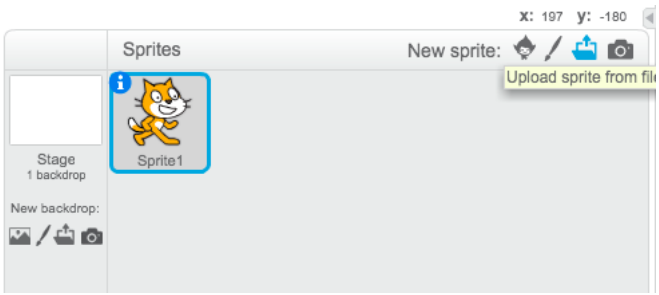
- To move your sprite to the correct location, you can use this code:

Finishing the game

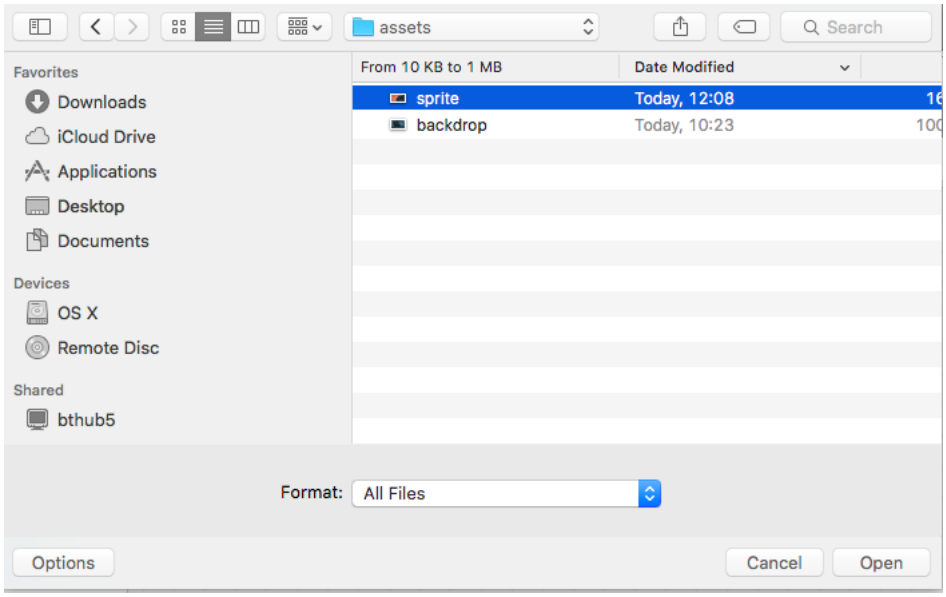
- To finish off, you'll need to find and download an image of a curtain or a screen (https://www.google.co.uk/search?q=stage+curtain&source=lnms&tbm=isch&sa=X&ved=0ahUKEwjKg901k8_VAhXSL1AKHe1HDMIQ_AUICigB&biw=1362&bih=58
- This image needs to be imported as a second sprite.

Adding a sprite from a file

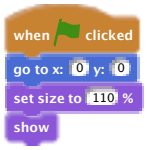
- Click **Choose sprite from file** to open up a file browser.



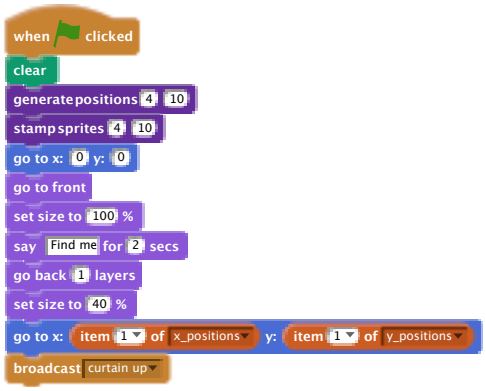
- Choose a file and click on **Open** when you are done.



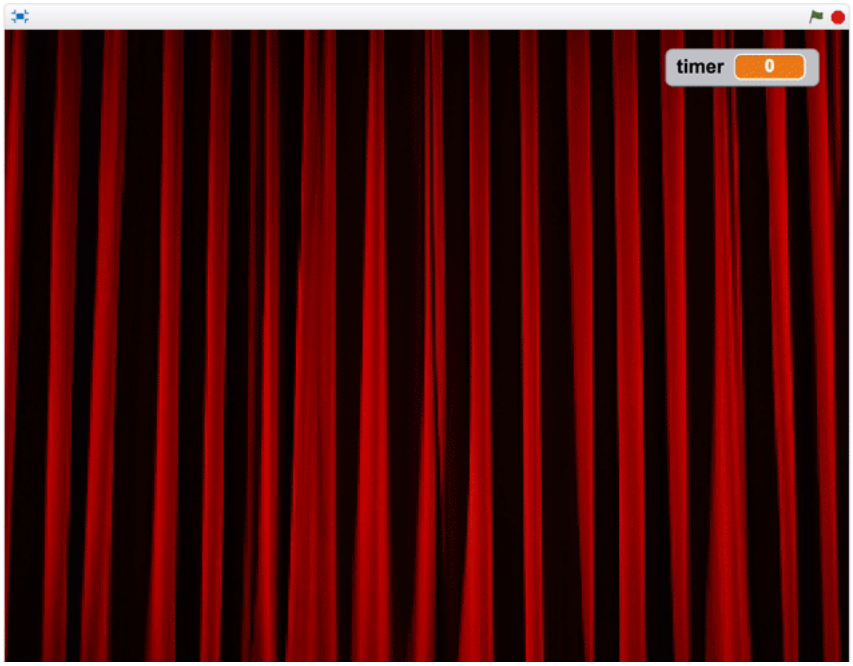
- To begin with, position the curtain sprite at `x:0 y:0` and then change its size so that it fills the screen. You also want to make sure it is visible.



- Then, back on your character sprite, add a broadcast of `curtain up` to the end of the starting script.



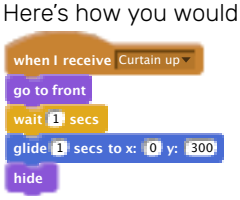
- On the curtain sprite, you need a script that will do the following:
    - Bring the curtain to the front
    - Wait a little bit while the sprites all get drawn
    - Glide the curtain upwards so it's near the top of the screen
    - Hide the curtain
    - Start a loop that counts for 10 seconds
    - When the time is over, show the curtain
    - Glide the curtain back to its original position
- Here's what it should look like:



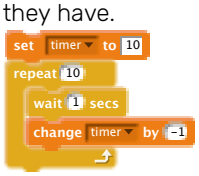
- Have a go at doing this yourself, and use the hints if you need help.

I need a hint

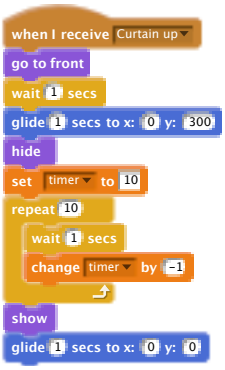
- Here's how you would start your script:



- Here's a simple script to create a delay of ten seconds. If the variable can be seen, it will let the player know how much time they have.



- Here's the full script:



- The very last part is to let the player know if they've won. On the character sprite, if the sprite is clicked, it should say `You've found me`, all the scripts in the game should be stopped.

Taking it further

Here are some ideas for further additions to your game that could make it 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 repeat, and provide you with a score based on your time to complete five attempts?

Published by the Raspberry Pi Foundation – www.raspberrypi.org