Phoenix

You will need:

Access to the Scratch website, from a computer or laptop.
(It might also work from an ipad or tablet)

This tutorial is best for people who have used Scratch a little bit before.

Setup:

If you haven't used Scratch before, click **join scratch** at the top and create an account. **You might need your parent's email to do this, so ask for help.** If you can't make an account, you can follow along, but you won't be able to save your work.

Once you've made an account follow the Phoenix Scratch studio here. This is where you can find the template versions of all the workshops we create. Feel free to remix them and post the new things you create.

Wait, what is Scratch?

Scratch is a free, online, block-based code editor. You might have used it in school. If you've been to one of our family workshops, you've probably seen us using it there.

There are lots of different tutorials available for using Scratch. Here we are going to focus on showing you ways of using Scratch that link to the work we do at Phoenix. This will include things like animation and drawing.

Global Garden

This project introduces the use of global variables. These can be used to build projects that people can interact with together.

To get started open up this project,

https://scratch.mit.edu/projects/385907861/

Step 1

Hopefully you have now had a look at the complete project. Flowers will have grown all around the stage.

If you click the watering can you can add an extra flower.

The project will grow a flower for each time the watering can has been clicked, by anyone, anywhere.

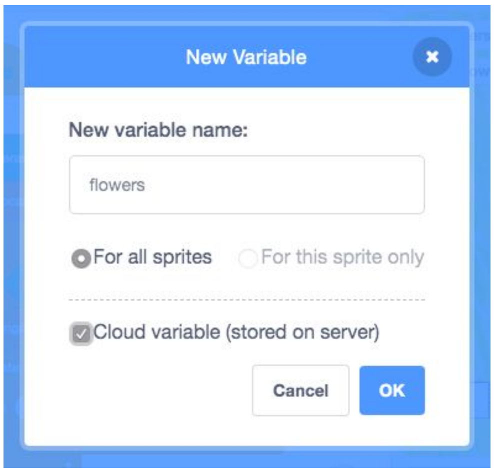
After you've had a play with this, open up this project to remix

https://scratch.mit.edu/projects/385933730/

Here we have exactly the same components as the finished project, but without as many things happening.

Step 2

The first thing to look at, is the variables section. Do you see the cloud symbol next to the flowers variable? This means that it is a cloud variable. You can see how we've set this up below.


New Variable

New variable name:

flowers

[✓ For all sprites] For this sprite only

[✓] Cloud variable (stored on server)

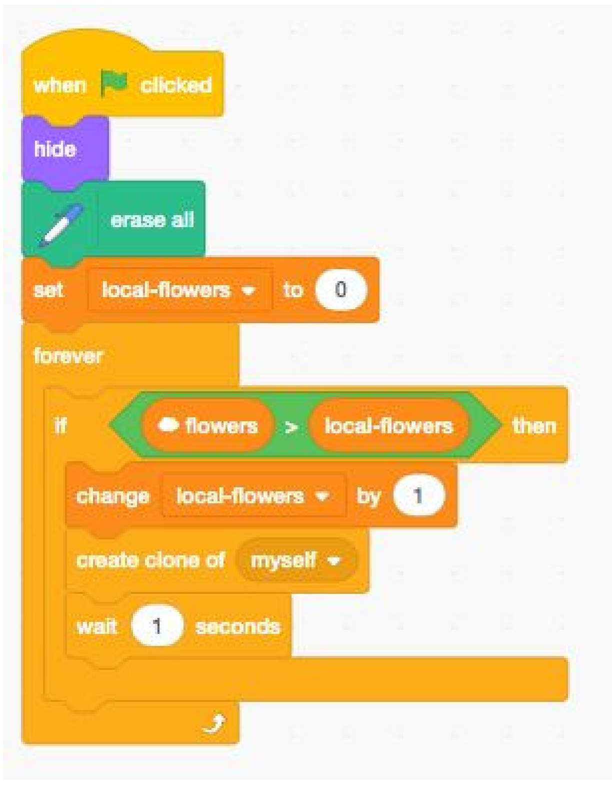
Having a variable set in this way, means that it will be stored on the Scratch servers. Any time someone opens up the project, instead of creating copies of the variable on their own computer, the value is retrieved from the server.

In this project we are using the cloud variable to store the amount of flowers we want to place in the scene. You can use cloud variables to do all kinds of things. Keep in mind that these variables can only store numbers.

So now we have a way of keeping track of how many flowers we need. How will we create them?

Step 3

Open up the code for the flower sprite. See if you can work out what the code below is doing.



```
when flag clicked
hide
erase all
set local-flowers to 0
forever
    if flowers > local-flowers then
        change local-flowers by 1
        create clone of myself
        wait 1 seconds
```

As you can see, there are two variables here. We have the cloud variable, flowers as well as a local variable, handily named, local-flowers.

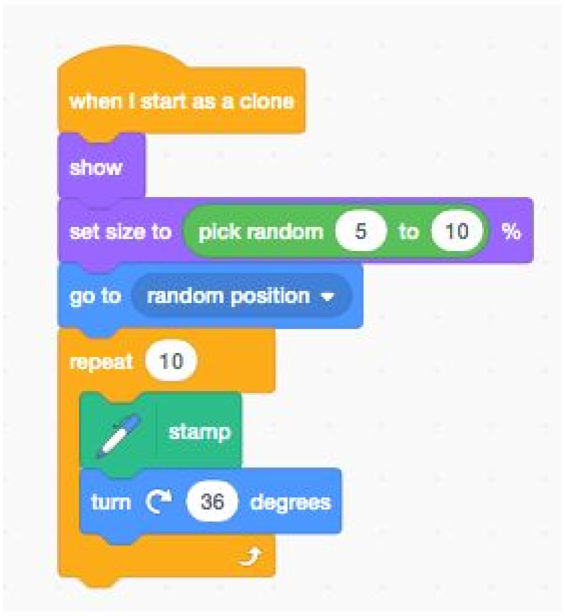
We make sure the local-flowers count is 0, and then forever, if flowers is greater than local-flowers, we add 1 to our local-flowers and make a clone.

In this way we will keep making clones until we reach the correct amount of flowers.

Step 4

But what are all the clones up to?

Our first code block for the clone is set up like this.



```
when I start as a clone
show
set size to (pick random (5) to (10)) %
go to (random position ▾)
repeat (10)
    stamp
    turn ↻ (36) degrees
```

Flowers are usually slightly different sizes. To achieve this we add randomness to the size.

Next we find a random position to plant the flower.

If you open up the costume tab you can see that our flower is just made of a single petal.

We use the repeat 10 loop to draw each petal in turn.

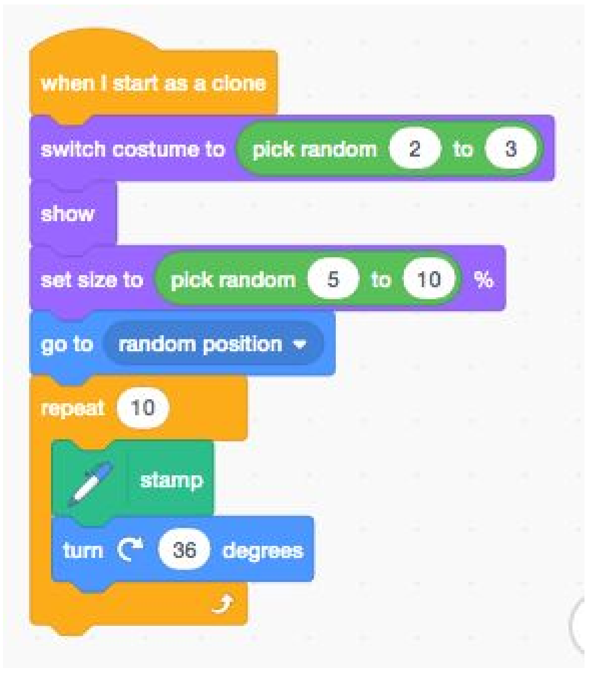
See what happens if you adjust some of the numbers here.

Can you think of some of the benefits of drawing our flowers like this?

Step 5

Well one reason for this, is that they are easier to draw! We only have to draw one petal ourselves, instead of them all.

If you connect the block as it is below, you'll start to see how fun this is.



```
when I start as a clone
switch costume to (pick random (2) to (3))
show
set size to (pick random (5) to (10)) %
go to (random position ▾)
repeat (10)
    stamp
    turn ↻ (36) degrees
```

We can switch between different costumes, to create all kinds of flowers. As we are saving time by only drawing a small part of the flower we can design more costumes!

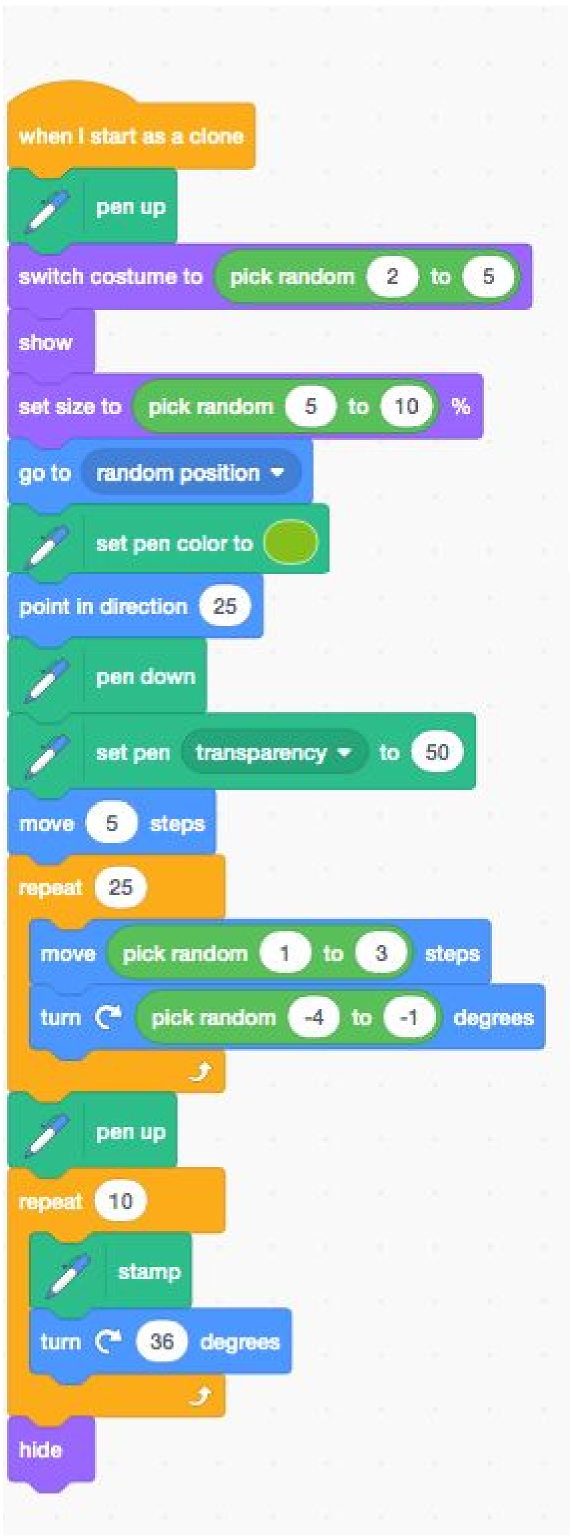
Make sure you remember to change the numbers in the switch costume block to match the costumes you have created.

Step 6

Let's connect our final flower sprite block now. As you can see there is quite a lot of extra code here.

All this code is doing is drawing a stem for the flowers. So before the petals are drawn, we point in direction 25 and then step forwards with the pen tool. The repeat 25 block allows us to draw a wobbly curve.

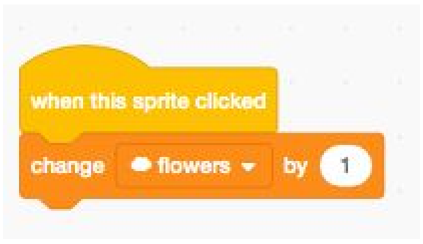
Experiment changing the values of these to see what kind of results you get.


```
when I start as a clone
pen up
switch costume to (pick random (2) to (5))
show
set size to (pick random (5) to (10)) %
go to (random position ▾)
set pen color to (🟩)
point in direction (25)
pen down
set pen (transparency ▾) to (50)
move (5) steps
repeat (25)
    move (pick random (1) to (3)) steps
    turn ↻ (pick random (-4) to (-1)) degrees
end
pen up
repeat (10)
    stamp
    turn ↻ (36) degrees
end
hide
```

Step 7

Now we'll turn our attention to the watering can.

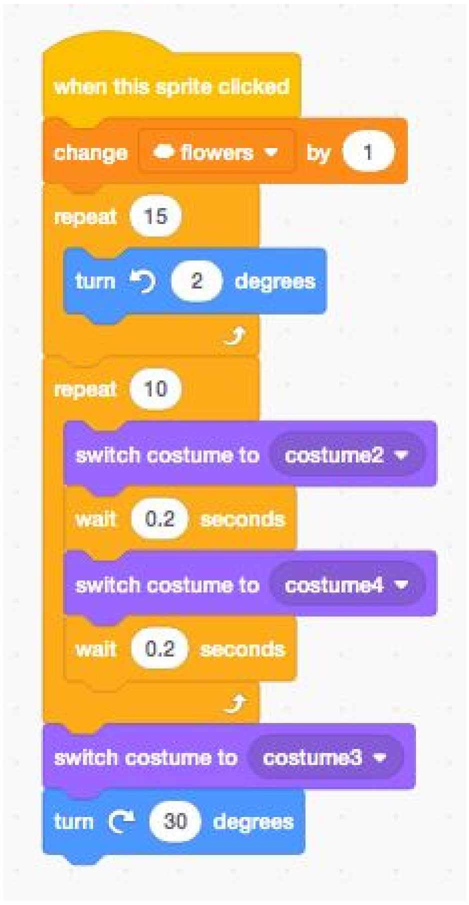
If you've tried clicking it already, you will have noticed that this block is connected.



```
when this sprite clicked
change [flowers ▾] by (1)
```

We are using this to add a value to the 'cloud' flowers.

Disconnect those blocks and attach the set below.



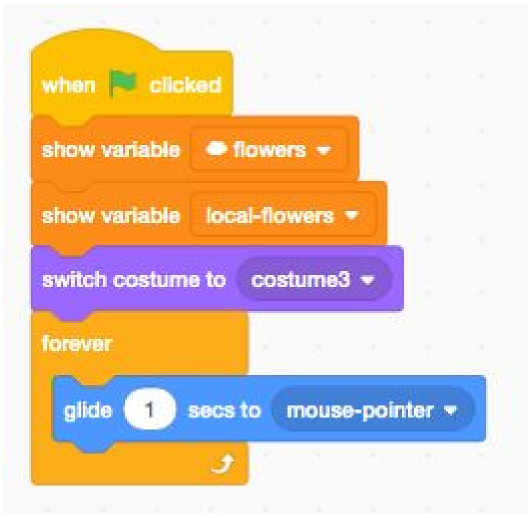
```
when this sprite clicked
change [flowers ▾] by (1)
repeat (15)
    turn ↺ (2) degrees
repeat (10)
    switch costume to [costume2 ▾]
    wait (0.2) seconds
    switch costume to [costume4 ▾]
    wait (0.2) seconds
switch costume to [costume3 ▾]
turn ↻ (30) degrees
```

This piece of code now makes use of the extra watering can costumes, to make it appear animated.

Open up the costumes and see if you can design your own watering can animation.

Step 8

The only thing left to connect now, is the way the watering can was following the mouse-pointer around. To do that, attach the blocks below.



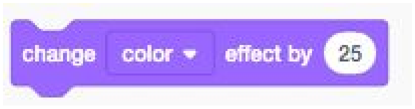
```
when [flag] clicked
show variable [flowers ▾]
show variable [local-flowers ▾]
switch costume to (costume3 ▾)
forever
    glide (1) secs to (mouse-pointer ▾)
end
```

We have used glide 1 secs to mouse-pointer. Putting this within the forever block, means that it will always be trying to glide towards the mouse-pointer. If you change the value here, you can make it move more slowly or more quickly.

Extra steps

And that's it. You're fully connected. You're now ready to really experiment with your garden design!

- How many different flower varieties can you create?
- Can you use the graphic effects to create different flowers as well?

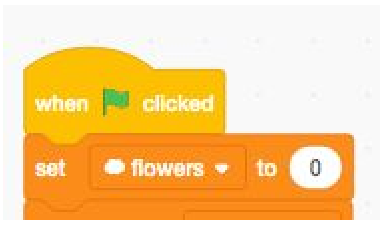


```
change [color ▾] effect by (25)
```

- Try creating a different backdrop for the garden.
- What happens if the flowers are huge!
- Can you make the flowers only be planted where the mouse-pointer is?

Remember to share your project to the Phoenix Scratch studio!

If you need to reset your 'cloud' flower variable, try adding something like this to get it back to 0.



```
when [flag] clicked
set [flowers ▾] to (0)
```

Hopefully you found this exercise interesting! We can’t wait to see what you create.

Phoenix.org.uk
@PhoenixLeic