Week 11

Christmas tree animation

This is our last session before the Christmas break, so we'll make a seasonal animation. A penguin will fetch in a Christmas tree, decorate it, then turn on the flashing lights.

We'll break this project down into stages. The penguin has to:

- Carry in the tree;
- Put each light onto the tree;
- Make the lights twinkle.

Backdrop and tree

You can find a backdrop in the library (the one in the picture above is '`room3`'). For the tree, you could use '`Tree2`' from the library, or find one on the web. Make the tree the right size to fit in the room.

Penguin

I chose '`Penguin2 Talk`' from the library. You don't have to use a penguin of course!

To make the penguin face in the direction it's going, go to its info ℹ and choose the 'flip left/right' rotation style: rotation style: ↻ ↔ ● — make sure the arrow ↔ is chosen.

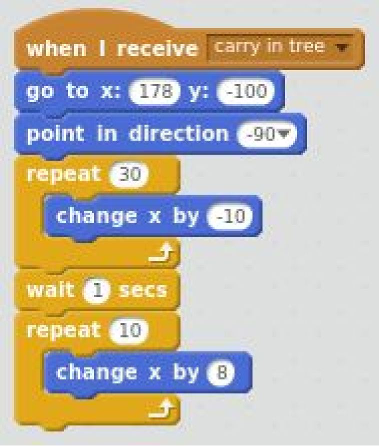
Carry in the tree

The animation starts with the penguin carrying the tree in from the right of the stage. The two sprites need to move together. The movement is fairly simple, so we'll be able to make sure their scripts line up time-wise. We'll trigger the animation with a broadcast.

Penguin

It starts off at the right-hand edge of the stage, faces left, then marches towards the left of the stage. Once it gets there, it leaves time to put the tree down, then steps back a bit.

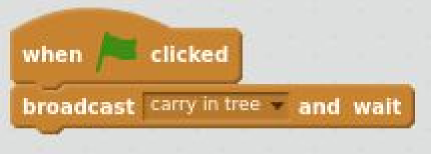
Add the script at the right **to the penguin**:



```
when I receive [carry in tree]
go to x: 178 y: -100
point in direction -90
repeat 30
    change x by -10
wait 1 secs
repeat 10
    change x by 8
```

We'll use a green-flag script in the stage to set everything going.

Add this script **to the stage**:



```
when [flag] clicked
broadcast [carry in tree] and wait
```

Test it!

Click green flag and check the penguin walks across the stage. Adjust the numbers in the script until it looks right for your penguin and room.

Tree

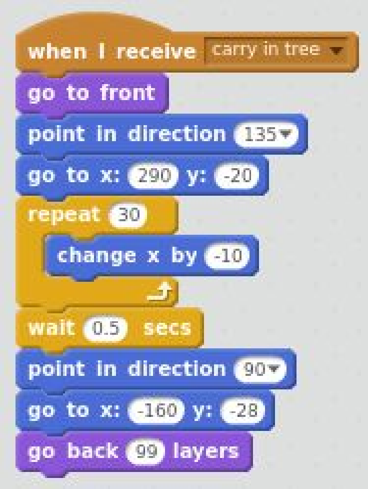
The tree will do something very similar, to make it look like the penguin is carrying it. It will be carried at an angle, and then put itself down the right way up.

We're being a bit lazy here, and just copying the first '`repeat`' block from the penguin to get the movement to match up.

The half-second wait should make everything happen at the right time.

The '`go back`' makes sure the penguin isn't hidden behind the tree once the tree has been put down.

You can adjust the carrying angle until you like how it looks.



```
when I receive [carry in tree]
go to front
point in direction 135
go to x: 290 y: -20
repeat 30
    change x by -10
wait 0.5 secs
point in direction 90
go to x: -160 y: -28
go back 99 layers
```

Test it!

Make sure that the penguin and the tree move OK when you click green flag.

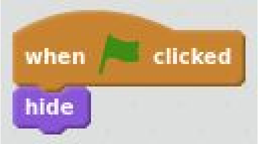
Tree lights

We need a sprite for the light. We'll use clones of this sprite to be the individual lights on the tree. I used '`Ball`', which has a few differently-coloured costumes.

Add a sprite to be the lights.

The 'original' sprite will never be seen, only its clones.

So ***add this script*** **to the light**:



```
when green flag clicked
hide
```

Put a light on the tree

We'll pretend the penguin takes each light out of its pocket. So the clone needs to appear where the penguin is, then glide to the right spot on the tree.

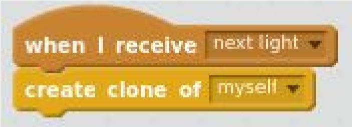
To tell the clone where 'the right spot' is, we'll use two variables.

Create two 'for all sprites' variables: **`light x`** and **`light y`**. They will be the coordinates of the spot where we want the light to end up on the tree. We'll worry about how to set them in a minute.

Make a new light appear

We need a new clone when a 'next light' message is received.

Add this script **to the light**:



```
when I receive [next light ▾]
create clone of [myself ▾]
```

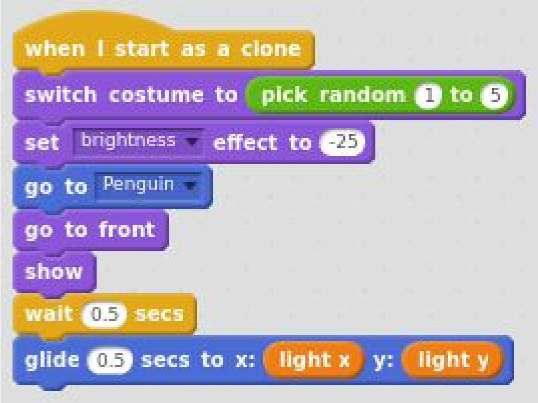
Make the new light go onto the tree

Each light clone which appears needs to go to the penguin, then glide to the tree.

We'll choose a random costume for each light. Each costume is a different colour.

Later, to make the light flash on and off, we'll use Scratch's ***brightness*** **effect**. When the light first goes on the tree, it should be dark.

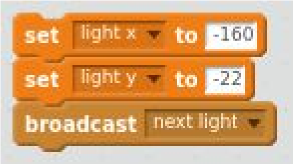
Add this script **to the light**:



```
when I start as a clone
switch costume to (pick random (1) to (5))
set [brightness ▾] effect to (-25)
go to [Penguin ▾]
go to front
show
wait (0.5) secs
glide (0.5) secs to x: (light x) y: (light y)
```

Test it!

To test this, ***add a little throw-away script*** **like this to the stage**, and double-click it.

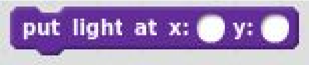


```
set [light x ▾] to (-160)
set [light y ▾] to (-22)
broadcast [next light ▾]
```

Putting up lots of lights

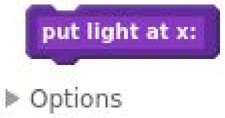
The stage will tell each new light to go onto the tree. To keep our code organised, we'll make a custom block to put up one light.

Custom block with inputs



Lots of Scratch blocks have ‘holes’ which you fill in with details of what you want to happen. We’ll a block to do the job of ‘put a light on the tree’, and this needs details — where should the light go?

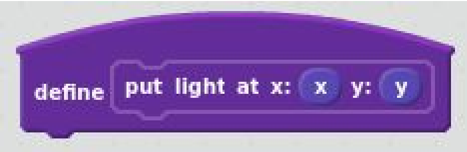
To create the custom block, **select the stage**, and then click Make a Block . Fill in ‘`put light at x:`’ but don’t click OK yet! We need to add two **inputs** for the ‘x’ and ‘y’ numbers saying where to put the light.



Click the little triangle to expand the ‘options’.

Then:

- Click Add number input: and call the input you just created ‘x’
- Click Add label text: text and put in the text ‘`y:`’
- Click Add number input: again, and call this input ‘y’

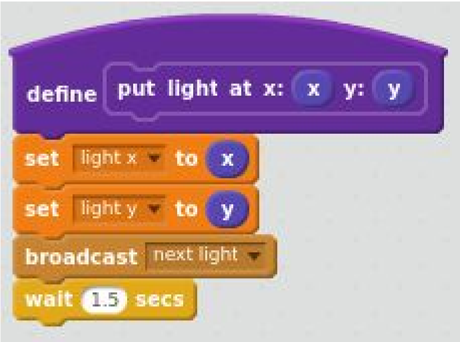


You should now have this ‘hat’ block to let you define what you want to happen when you use your new block. The inputs are like special variables which only exist inside your custom block’s definition.

We need to:

- Set up the ‘light x’ and ‘light y’ variables using the inputs to the block;
- Send a message to tell the light to clone itself and put that clone on the tree;
- Wait until the clone is on the tree.

To put this together as a script, ***add blocks* under the ‘define’ hat block to get**:

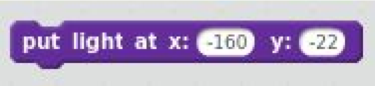


To get the purple ‘x’ and ‘y’ into the two ‘set’ blocks, drag them from the hat block.

(There are better ways of making sure the light’s clone has finished its work than ‘*wait 1.5s*’, but we’ll keep it simple for now!)

Test it!

Make a one-block throw-away script for the stage, and double-click it, to check this is working.

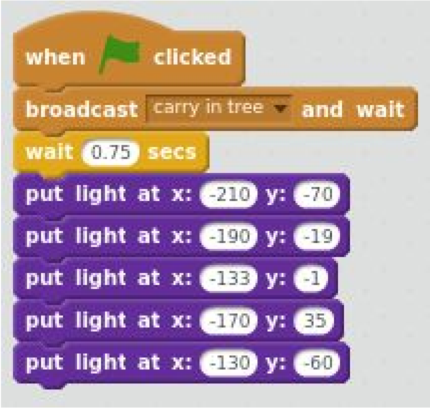


Put all the lights on the tree

Now we can use our block to put lights in different places on the tree.

***Add blocks* to the stage's green-flag script** to put lights in places you choose. You will probably need different numbers to the ones I've got here. One way to find the numbers is to point the mouse pointer where you want the light to go, and read the 'x' and 'y' numbers under the bottom-right of the stage.

You can put more lights in by using the 'put light at' block more times.



```
when ⚑ clicked
broadcast carry in tree ▾ and wait
wait 0.75 secs
put light at x: -210 y: -70
put light at x: -190 y: -19
put light at x: -133 y: -1
put light at x: -170 y: 35
put light at x: -130 y: -60
```

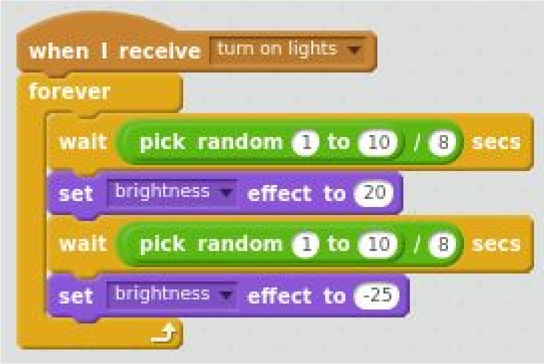
Turn on the lights!

After all that work, the lights should twinkle. We'll do this by switching the 'brightness' effect back and forth between bright and dark, waiting random times between each switch.

We don't want whole numbers of seconds, so we'll use a bit of arithmetic to get sensible waiting times.

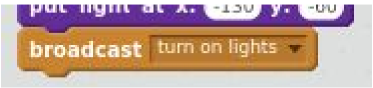
Remember '/' means 'divide'.

***Add this script* to the 'light' sprite**:



```
when I receive turn on lights ▾
forever
    wait (pick random 1 to 10) / 8 secs
    set brightness ▾ effect to 20
    wait (pick random 1 to 10) / 8 secs
    set brightness ▾ effect to -25
```

And to make this be the last stage of the animation, ***add a block* to the stage's green-flag script**:



```
put light at x: -130 y: -60
broadcast turn on lights ▾
```

Test it!

Green flag should make the whole animation work, ending with flashing lights on the tree.

Challenges

If you have time, either now or over the break, you could try:

- Make the lights fade brighter and dimmer instead of flashing.
- Put a star on top of the tree and make it spin round when the lights are turned on.
- Make the penguin put presents under the tree.
- Add other decorations.
- Make Santa and Rudolph fly overhead once the tree is lit up.

Key points

Break a problem down into stages, testing as you go along.

Use the ‘brightness’ effect to make each light flash.

Choose costumes randomly, by number.

Use variables to communicate numbers between sprites.

Make a custom block with inputs.

About this document

Scratch is developed by the Lifelong Kindergarten Group at the MIT Media Lab. See http://scratch.mit.edu.