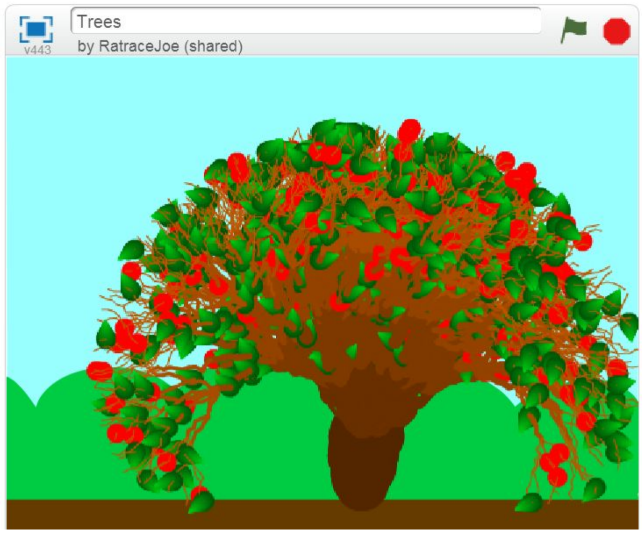
Longlevens Code Club

Scratch - Tree Generator

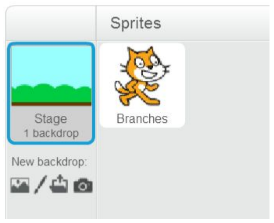
This project will show how to animate a tree growing. The pen tool will be used to do all the drawing, sprite cloning will be used to generate the many branches/leaves/fruit required.

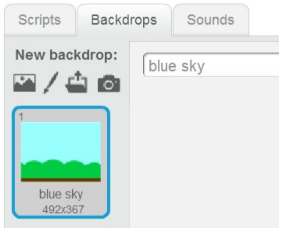
The way the game will work is the player clicks on the stage, and the tree should grow from that click point.
The following screenshot shows what we are aiming for (eventually)



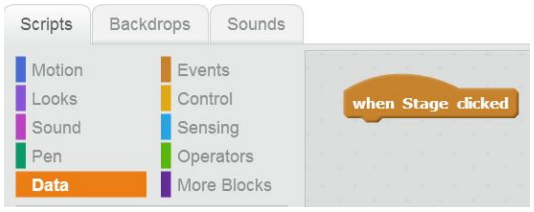
Capture the click, setup the Sprite

The mouse click will be captured on the stage, so click on the stage. Pick a backdrop that suites, I have selected the **blue sky** backdrop. You may need to delete the default white backdrop.

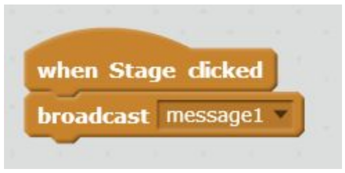




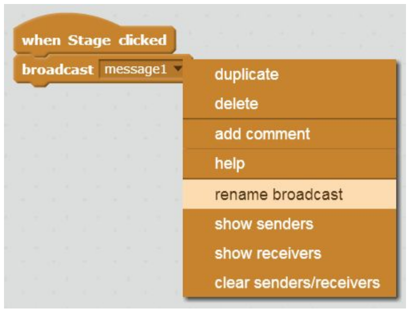
The first thing we must do is capture the mouse click.
Under **Events** drag a **When Stage Clicked** block into the code area.

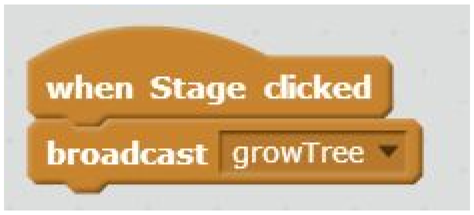


Under **Events** drag a **broadcast <message1>** block into the code area.

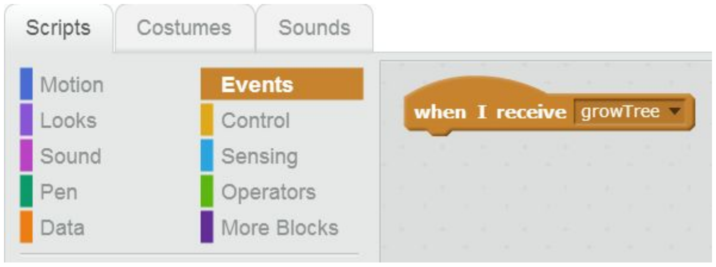


To make the broadcast more meaningful, right click on the broadcast block, click on **rename broadcast** and give it a name **growTree**.





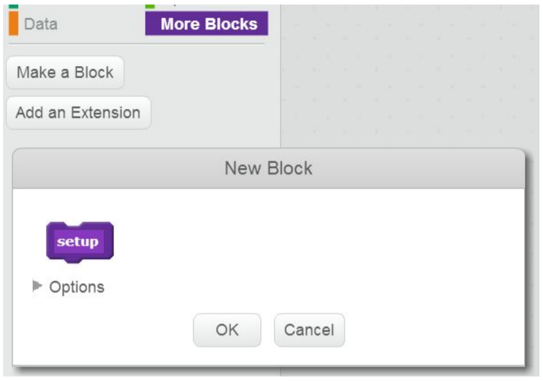
Now we have to head over to our sprite to enter the code. Click on **Sprite1**. You should now have a blank code area.
From **Events** drag a **When I receive <growTree>** block into the code area.

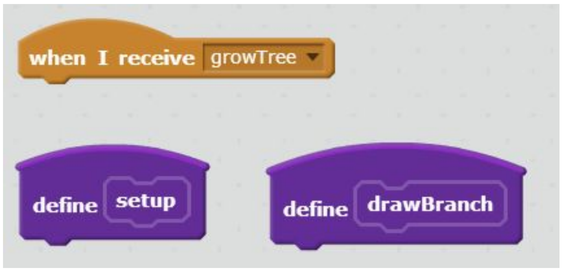


In order to keep the functions manageable, we will create two new blocks. One for setup which will be used to give our variables values, the other will be called to draw the tree itself.

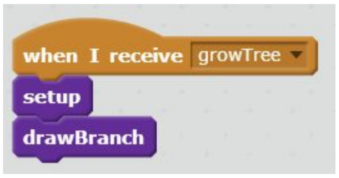
To create each block, under **More Blocks** click on **Make a Block**. Type in the name and click ok. Neither of these will require any options.

- setup
- drawBranch



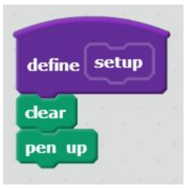


Call these two functions in the broadcast receiver, first drag **setup** from **More Blocks** in, then **drawBranch**. The broadcast receiver should now look like this.



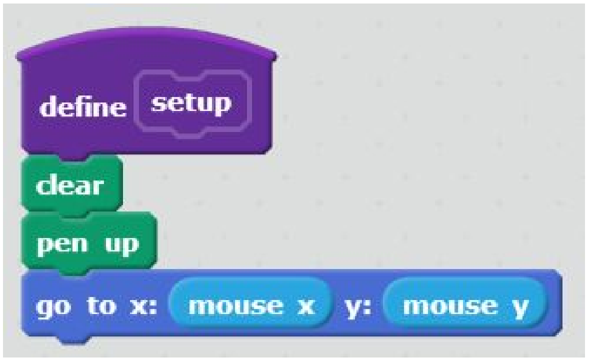
```
when I receive growTree
setup
drawBranch
```

Before we do anything else we should ensure the pen is cleared, and up. From **Pen** drag a **clear** block into our **setup** function. Then drag a **pen up** block in. This ensures that any residual tree has been cleared.



```
define setup
clear
pen up
```

Now we should move the sprite to the click location, remember this was triggered by the player clicking on the stage. From **Motion** drag a **go to x: y:** block into the setup function.
From **Sensing** drag the value of **mouse x** into the **x** parameter, and **mouse y** into the **y** parameter. The setup function should look like this.



```
define setup
clear
pen up
go to x: mouse x y: mouse y
```

Try now clicking in the game area, the sprite should move to the location of the click.

Setup Variables

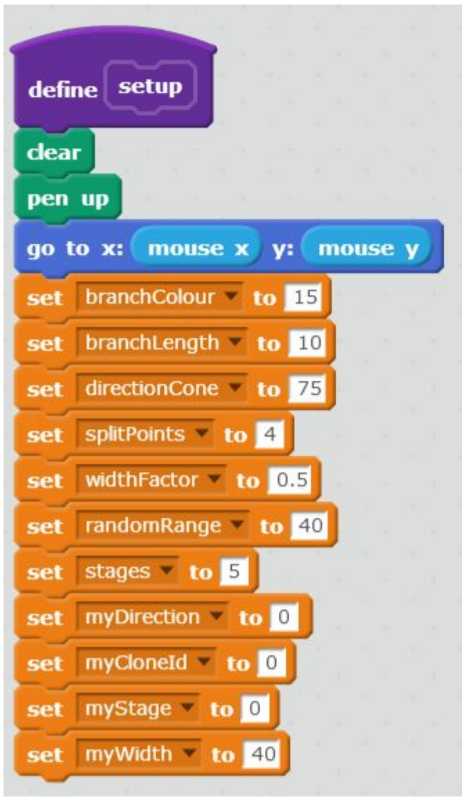
Before we start drawing, we need to create some variables. These variables will be used to define the behaviour of the sprite. Since we are using cloning, some of these variables will have to be private to the sprite. When a variable is private to a sprite, each clone gets their own copy. This will be essential for drawing multiple branches simultaneously.
Under **Data** click on **Make a Variable**, ensure **For this sprite only** is clicked create the following.

- myDirection - each clone will head out in a different direction
- myCloneId - each clone will use its clone id to select a direction from a cone
- myStage - a stage is a distance from the trunk, the tree will fan out a limited number of times
- myWidth - this is the width to draw the given branch, the branches will get thinner the further out they are, this will create a more realistic effect.

Now we need to create some more variables, these will be **For All Sprites**. They are constants which are used to drive the behaviour of the tree.

- branchColour - This is the base colour for the branch
- branchLength - This is the length of each stage
- directionCone - This is the angle cone that branches will fan out. The larger this is, the more spread out the branches are.
- splitPoints - This is the number of branches to split into at each stage.
- randomRange - This will be used to introduce random direction to the branches. This will give a much more realistic effect.
- widthFactor - This is the reduction factor for the width of the branches for each stage.
- stages - This is the number of stages to fan out.

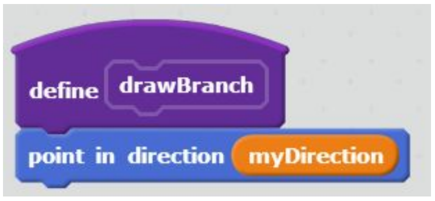
These variables will be setup in the **setup** function. So, from **Data** drag a **set <varName> to []** block into the **setup** function. Do it for the variables shown, with the values shown. The setup function should then look like this.



```
define setup
clear
pen up
go to x: (mouse x) y: (mouse y)
set branchColour to 15
set branchLength to 10
set directionCone to 75
set splitPoints to 4
set widthFactor to 0.5
set randomRange to 40
set stages to 5
set myDirection to 0
set myCloneId to 0
set myStage to 0
set myWidth to 40
```

Draw Branches

Now the **setup** function is ready, we now need to build the **drawBranch** function.
From **Motion** drag a **point in direction <>** block into the function. From **Data** drag **myDirection** into the direction setter.



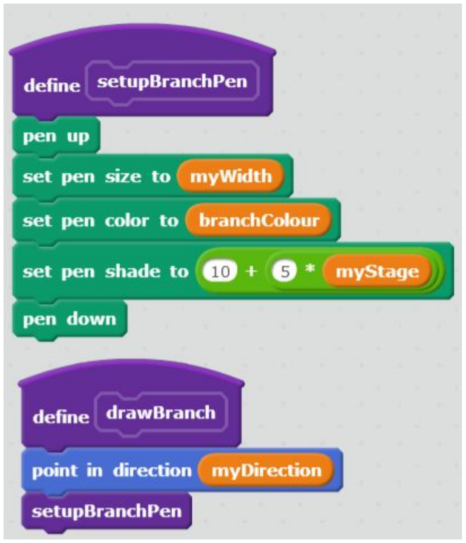
```
define drawBranch
point in direction myDirection
```

In order to setup each clone's branch pen, we shall create a new block.
Under **More Blocks** click on **Make a Block** and give it the name **setupBranchPen**.

Using tools from **Pen** and variables from **Data** build the following.

Drag a call to this function into **drawBranch**.

The two functions should look like the image on the right.



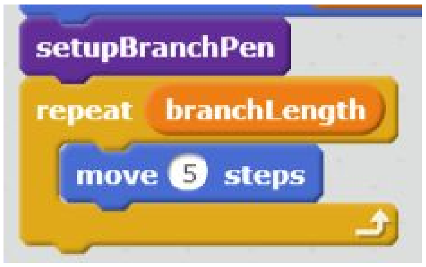
```
define setupBranchPen
pen up
set pen size to myWidth
set pen color to branchColour
set pen shade to (10 + (5 * myStage))
pen down

define drawBranch
point in direction myDirection
setupBranchPen
```

Now we are ready to draw the branch. From **Control** drag a **repeat <>** block into **drawBranch**.

From **Data** drag **branchLength** into the repeat loop.

From **Motion** drag a **move <5> steps** into the loop.

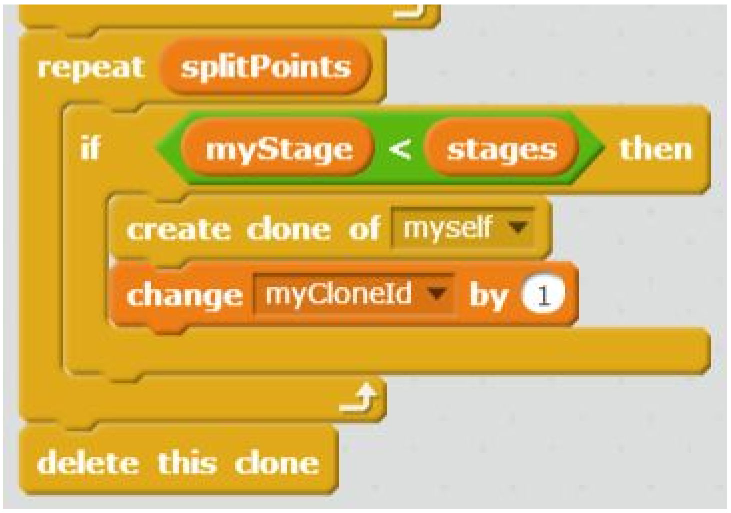


```
setupBranchPen
repeat branchLength
    move 5 steps
```

Now we need to split out and clone our sprite to draw many branches.

From **Control** drag another **repeat <>** block into our function. Then from **Data** drag **splitPoints** into the repeat block.

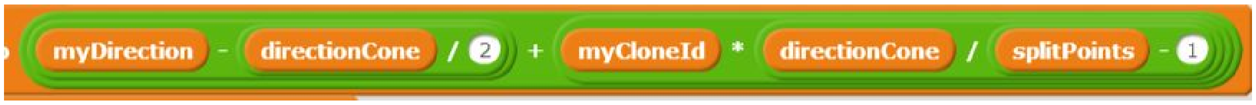
From **Control** drag an **if <>** block into this loop. Using **Operators** and variables from **Data** build the function as shown on the right.
Finally from **Control** drag a **delete this clone** to the end of the function.



```
repeat splitPoints
    if myStage < stages then
        create clone of myself
        change myCloneId by 1
delete this clone
```

Now we have to handle the cloning process. So from **Control** drag a **when I start as a clone** block into the code area.
From **Data** drag a **change <myStage> by 1**.
From **Data** drag a **set <myDirection> to <>** in
Pay careful attention to the brackets in the following formula, they show you what order the operators should be dragged in. Using **Operators** and variable values from **Data** try and create the following.

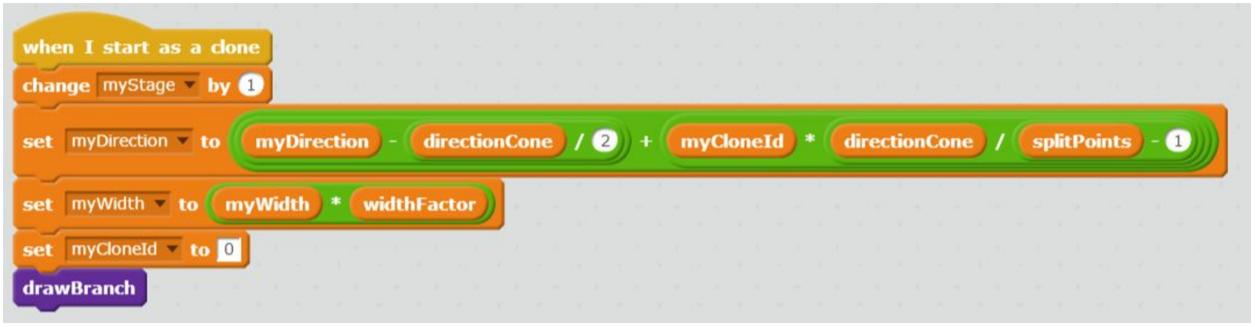
(myDirection - (directionCone / 2)) + (myCloneId * (directionCone / (splitPoints - 1)))



```
((myDirection - ((directionCone / 2))) + ((myCloneId * ((directionCone / ((splitPoints - 1)))))))
```

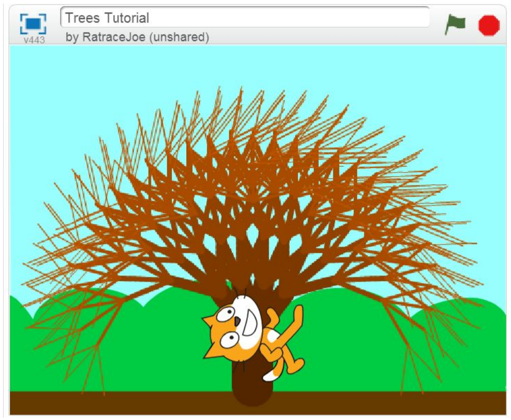
From **Data** drag a **set <myWidth> to <>** in.
Using **Operators** multiplication block and the variables from **Data** create the formula **myWidth * widthFactor** to set the width of this branch. This has the effect of reducing the width from its previous value.

From **Data** drag in **set <myCloneId> to <0>**
Lastly, from **More Blocks** drag a call to **drawBranch** onto the end of the function. The clone function should look like this.



```
when I start as a clone
change myStage by 1
set myDirection to ((myDirection - ((directionCone / 2))) + ((myCloneId * ((directionCone / ((splitPoints - 1)))))))
set myWidth to (myWidth * widthFactor)
set myCloneId to 0
drawBranch
```

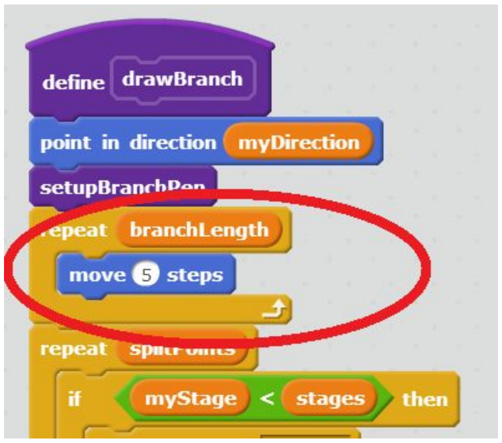
Try running this code and you should get the following, notice that we get a lot of cat sprites cloning during the drawing!



Introduce Randomness

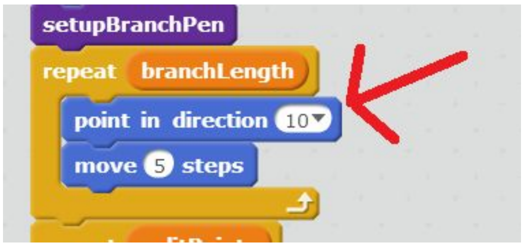
Now we have our tree growing, it looks a bit fake, the lines are all perfectly straight, which normal trees are not. So we will now introduce some randomness to give the tree a more organic look.

To do this we can simply add a step to our **drawBranch** function. Find the section of the function that draws the branch.



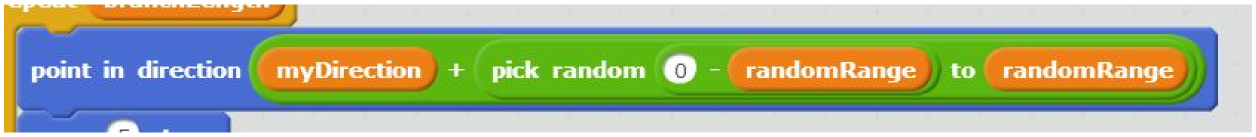
```
define drawBranch
point in direction (myDirection)
setupBranchPen
repeat (branchLength)
    move (5) steps
end
repeat (splitPoints)
    if <(myStage) < (stages)> then
```

Before we execute the **move 5 steps**, let us change the direction by a small randomised amount. During an earlier step, we should have created a variable called **randomRange**. This is the variable we will be making use of now.
From **Motion**, drag a **point in direction []** block into the repeat statement.



```
setupBranchPen
repeat (branchLength)
    point in direction (10 ▾)
    move (5) steps
end
```

Using a combination of **Operators** and variables from **Data** create the following formula.



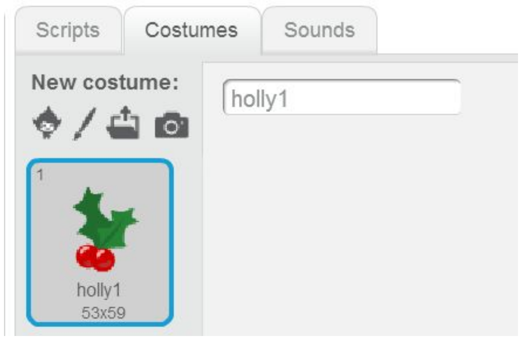
```
point in direction ((myDirection) + (pick random ((0) - (randomRange)) to (randomRange)))
```

Now try clicking in the stage again, the branches should look a bit more natural. Although a little sparse!

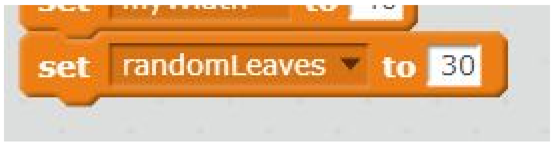
Add Leaves

Now that we have a tree, it would be nice to add some foliage (leaves, fruit etc). The way we we will do this is to use the **stamp** block under **Pen**. The **stamp** block leaves a copy of the sprite on the game area, what is great about using **stamp** it allows us to leave visual copies of a sprite but without incurring the overheads of having more clones.

First we will need to change the costume of our sprite, the default is the cat. Under **Costumes**, click on **Choose Costume from Library**. I have used the **holly** image. You could draw your own, but wait until you have finished the code before you do this (see extension tasks)

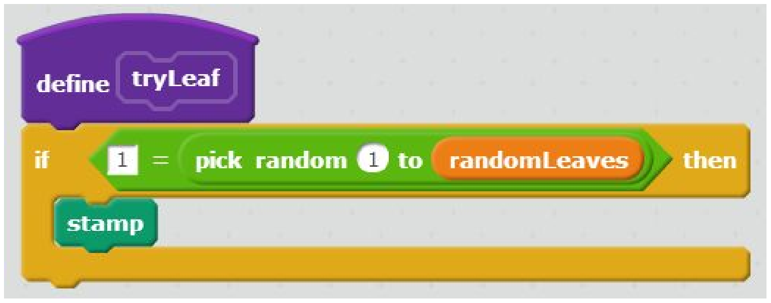


We now need to create a variable to tune the random generation of fruit/leaves. Under **Data** create a new variable called **randomLeaves** and set its value to **30** in the **setup** function.



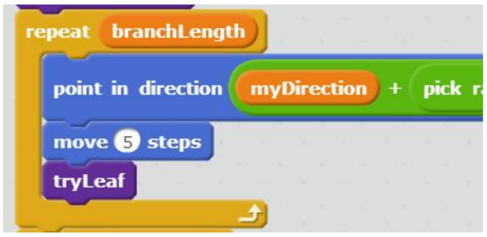
Now under **More Blocks** create a new **Block** called **tryLeaf**. This block will use a randomly generated number to see if it should draw a leaf.

From **Control** drag an **if <>** statement into the new function. Into the **if <>** statement create the following formula. Then from **Pen** tool drag a **stamp** block into the **if <>** statement.



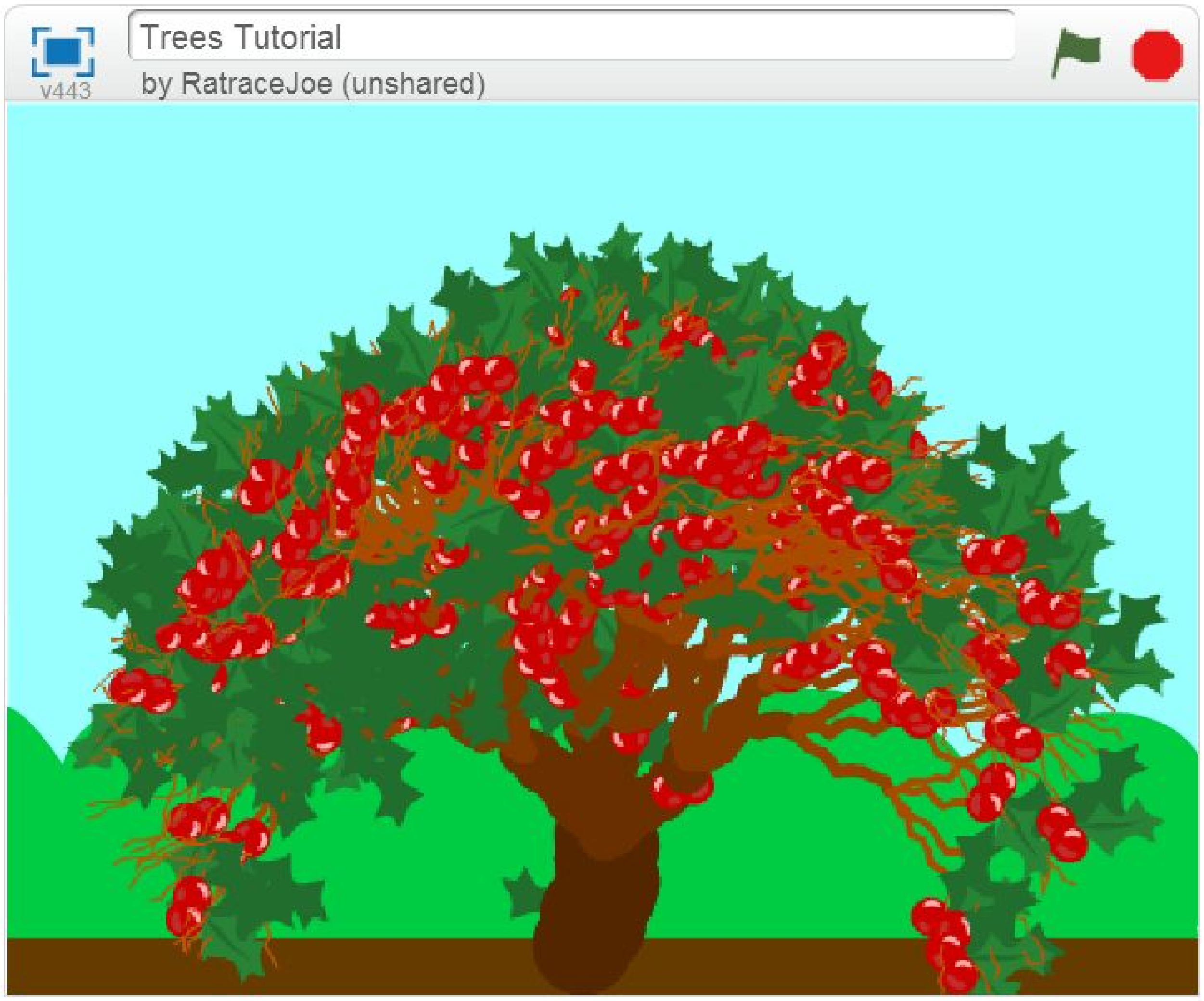
```
define tryLeaf
if <(1) = (pick random (1) to (randomLeaves))> then
    stamp
```

To call our function drag a **tryLeaf** block into our **drawBranch** function. Put it just after the direction randomiser we wrote in the last step.



```
repeat (branchLength)
    point in direction ((myDirection) + (pick r
    move (5) steps
    tryLeaf
```

Now if you click on the game area, you should get something like the following.



Extension Tasks for Finishing Early

- Try drawing your own leaf costume
- Try randomly switching costumes to the fruit so you get a mixture.
- Try playing with the different variables like **branchLength** and **directionCone** to see what effect they have on the appearance of the tree.