How to draw a Fern plant (and implement local variables):

Concepts used in this program:

- Algorithms
- Conditionals (IF)
- Conditions: YES/NO questions
- Lists
- Pen art
- Procedures with inputs
- Recursion
- Relational operators (=, <, >)
- Simple looping (repeat, forever)
- Stopping scripts
- Variables - local/global scope
- Variables – numbers
- Variables as sliders

The Fern Program:

Scratch (as of Version 3) allows you to have private variables for a sprite, but not for a new custom block. In this blog, we will see how to overcome this shortcoming.

I wanted to port my Logo script that draws a nice-looking Fern plant (actually just a branch) as shown below. As you can see, it's a recursive design.

Level = 2 Level = 3 Level = 4

Here is the recursive Logo script that draws this design:

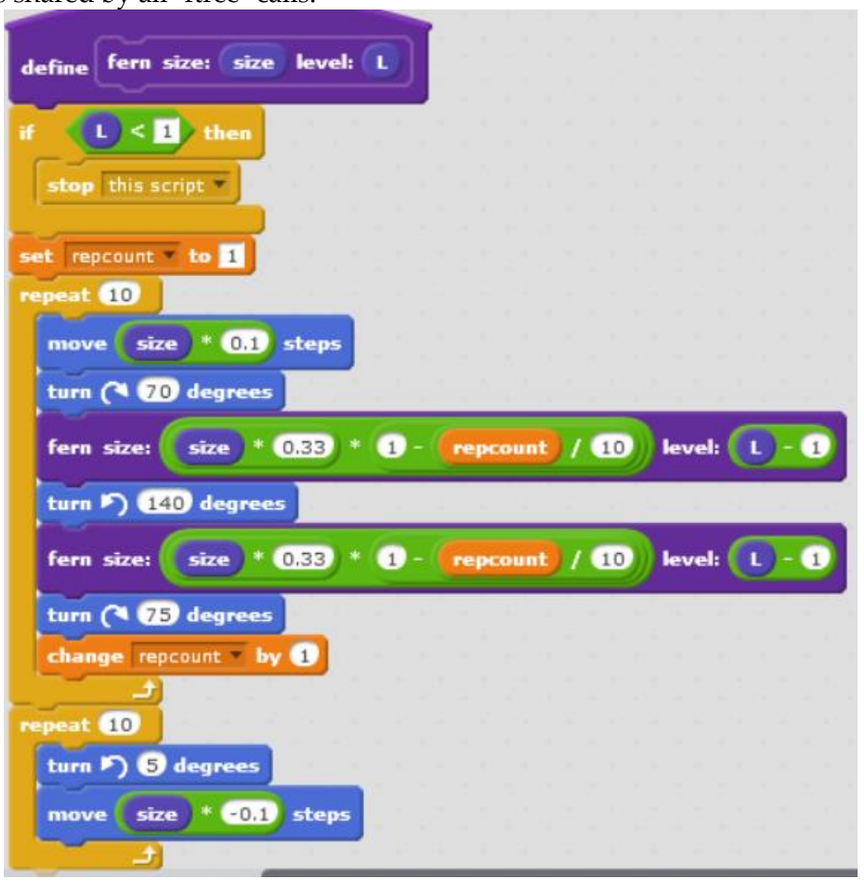
```
Erase "ftree
TO ftree :s :l :t
IF :l<1 [STOP]
SETPENSIZE ROUND :t
REPEAT 10 [
  FD :s/10
  RT 70 ftree 0.33*:s*(1-REPCOUNT/10) :l-1 :t*.7 LT 70
  LT 70 ftree 0.33*:s*(1-REPCOUNT/10) :l-1 :t*.7 RT 70
  RT 5
]
```

```
REPEAT 10 [LT 5 BK :s/10]
SETPENSIZE ROUND :t
END
ftree 700 2 8 ; draws the level=2 design above
```

Note: The parameter "t" is for gradually reducing pen thickness as you go deeper into the recursion to draw a more realistic-looking Fern.

When I started porting this script to Scratch I hit a major roadblock! The "repcount" thing in the script above is a local variable, i.e. it is local to "ftree". Every recursive call to "ftree" uses its own "repcount" variable which is not visible to any other running instance of "ftree".

So, if your Scratch script looks something like this, it won't work because the variable "repcount" is shared by all "ftree" calls.



How do we fix this problem?

The Solution:

Well, I thought back to my college course in Operating Systems in which I had learned that local variables are memory locations on a stack. So, I said to myself: Why not implement a stack in

Scratch? And that's exactly what I did to solve the above problem and draw my beloved Fern plant.

I used the List mechanism to implement my stack. Here is how it works. For the sake of brevity, I decided to use "i" in place of "repcount".

I created a list variable called "ilist". The last location in "ilist" is used by the custom block as its local "i" variable. The custom block adds this new location at the beginning and deletes it before returning. (That is how a stack works.) See the algorithm for the custom block below:

```
Add a new item to ilist
<Do whatever>
Delete the last item of ilist
```

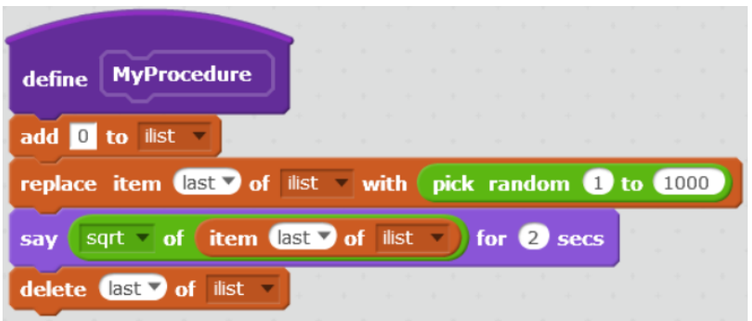
So, now the custom block has its own local variable – a location in "ilist".

Let's take a trivial example to understand how this might work:

Let's say I have this new custom block:

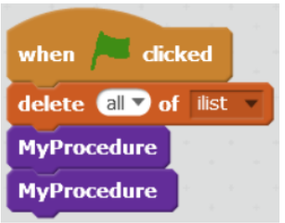
```
MyProcedure:
Local variable I = pick random 100 to 200
Print square root of I
```

Here is how this can be implemented using our stack.



The first line allocates a local variable by adding an item to ilist. The last line de-allocates the local variable by deleting the last item in ilist.

Calling script:



So, that's how you can implement local variables in Scratch! You will need a separate list for each local variable.

Here is my final working Fern program: https://scratch.mit.edu/projects/296836752/

Author: Abhay B. Joshi (abjoshi@yahoo.com)
Last updated: 2 July 2019