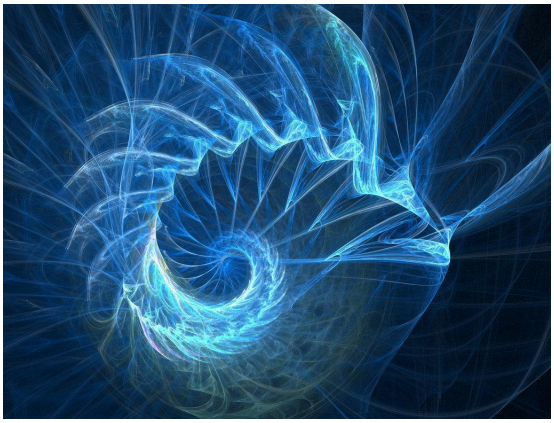
Creating Fractals in StarLogo Nova

Fractal (n): an object or quantity that displays self-similarity, in a somewhat technical sense, on all scales. (http://mathworld.wolfram.com/Fractal.html)

Description

This activity teaches fractals and recursive thinking through Sierpinski Triangles, presenting two ways to build them in StarLogo Nova. The first way is to apply recursion by splitting agents, and the second is to have one agent recursively draw every triangle.
Each challenge starts with a project that has some starting code programmed for you. The challenge describes the starting state of the program and explains the recursive idea behind the fractal. Your job is to turn the recursive idea into code on the Turtle page. The solution is provided on the back of each challenge, along with a link to a working version of the program.



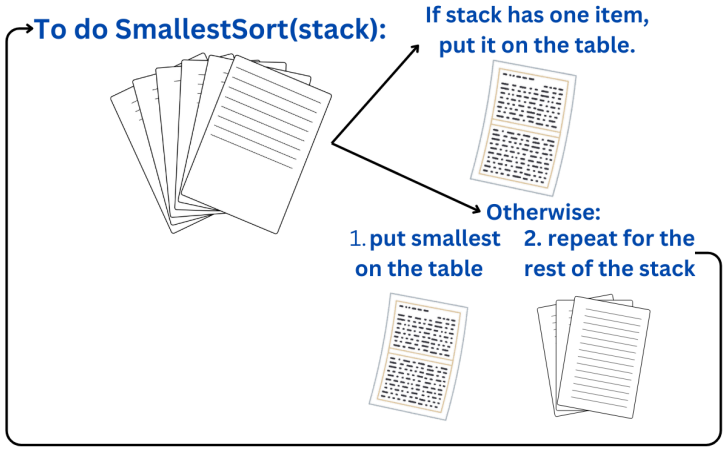
(image credit: http://wildekyote.deviantart.com/art/Fractal-365-Day-49-179679535)

Recursion

In computer science, recursion is a method of solving a problem in which the solution depends on solutions to smaller instances of the same problem.[1] Recursion solves such recursive problems by using functions (StarLogo calls them *procedures*) that call themselves from within their own code. The approach can be applied to many types of problems, and recursion is one of the central ideas of computer science.[2]
https://en.wikipedia.org/wiki/Recursion_(computer_science)

Example: Imagine that you needed to sort a stack of papers by last name. One obvious way would be to look through the stack for the "lowest" (first alphabetically) paper, put that facedown on the table, and then repeat with the rest of the stack. Of course, once "the rest of the stack" is just one paper, you can put it straight down without needing to look at it, and you will know that your stack is now sorted alphabetically.

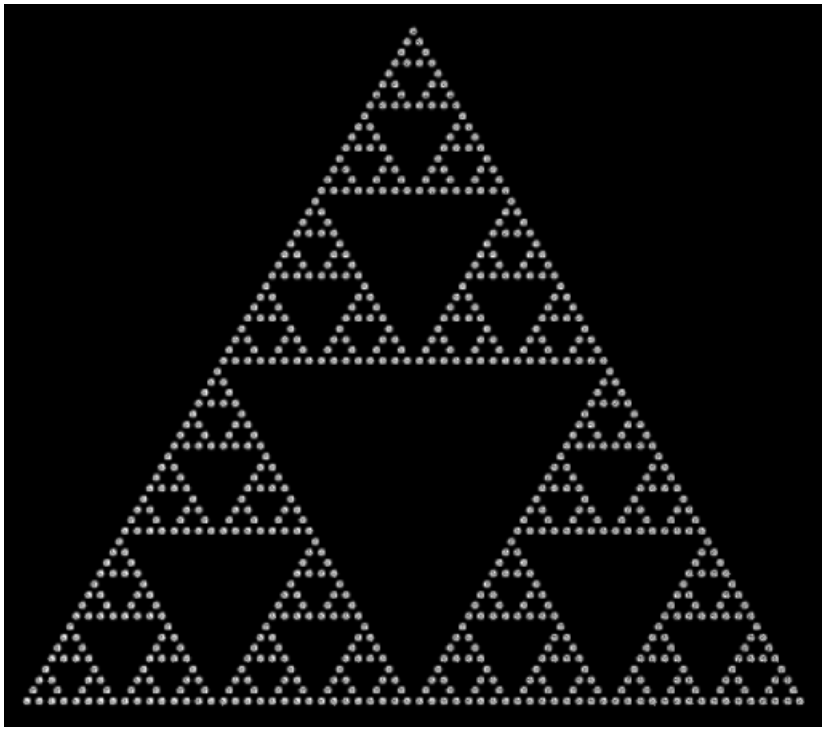
You could call this sorting procedure "SmallestSort(stack)", and the pseudocode for how it works is this:




```
To do SmallestSort(stack):
      If stack has one item,
put it on the table.
      Otherwise,
            Find the smallest item in the
      stack,and put it on the table.
            Then do SmallestSort(rest of
the stack)
```

Sierpinski Triangle

Example Image



The Sierpinski triangle is a well-known fractal created by subdividing a triangle. It is a great example of recursion. Here are the instructions for drawing a Sierpinski triangle:

To DrawSierpinski (of a given height):
    Draw a triangle of the given size
    In each of the three corners of the triangle,
        DrawSierpinski (of half the height)

You'll notice that unlike sorting, this algorithm doesn't have an "end state" or "base case" - a true fractal goes on forever! But since StarLogo Nova is finite, we will stop drawing at a certain "depth", or number of recursions.

This activity has two methods for making a Sierpinski triangle. The first is to recursively split an agent (who represents the "parent" triangle) into three smaller ones (who represent the "corner" triangles). This method is easier to understand. The second method is to use a recursive procedure (that is, a procedure that calls itself) to stamp colors on the terrain to draw the triangles. This method is much more efficient, but also more difficult to understand. Feel free to try either method, or do them both.

Part 1 - Recursive Splitting

Starting State

Open the Sierpinski Triangle with Recursive Splitting - Starter project from the Fractals Activity gallery at https://sailctm.slnova.org/mgriza/?open=381465

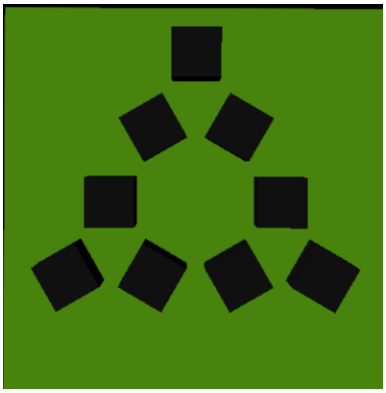
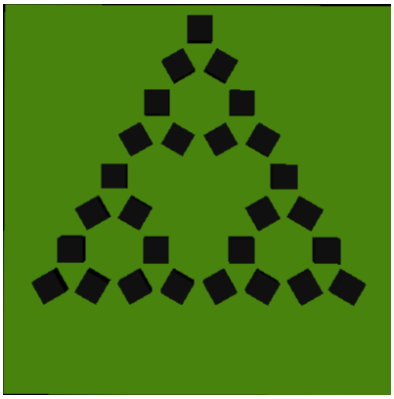
The project starts with a working setup button that creates a single Turtle and counts the number of Turtles on the screen. Each time the "recurse" button is pushed, the program re-counts the number of agents. It first uses yield in order to wait for the agents to split and then count the result. Additionally, there is already some code on the Turtle page that you might want to use and add to.

Recursive Idea

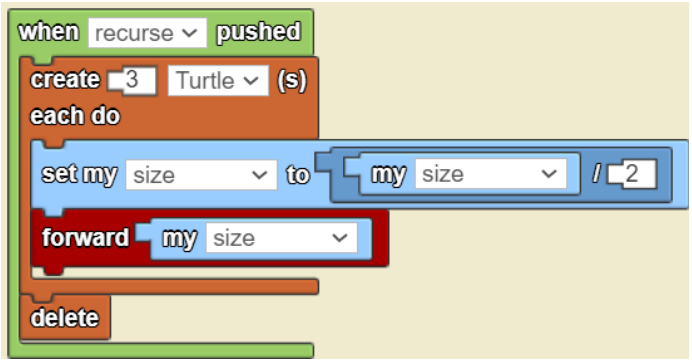
The recurse button is meant to increase the level of "depth"/ recursion the Sierpinski Triangle has. At its core, The Sierpinski Triangle is a triangle made up of three smaller Sierpinski Triangles. So, try splitting your turtle into three smaller turtles and spacing them out a little. The recursive step is that each of *them* will split into three smaller turtles, and so on.

Hints/Tips

To determine the right amount of distance that new agents should walk from the location they were created from, let's look at the Sierpinski triangles with a smaller amount of recursive levels. Each time recurse is pushed, for the new agents created, the size and distance from the center is **½ of the size and distance** from the center of the parent they replace.

Solution



Additional Challenge: make a 3D version, using a tetrahedron as the shape for the turtles, such that it looks like triangular holes are simply appearing in the original turtle.

Part 2 - Recursive Procedures

Starting State

Open the Sierpinski Triangles with Recursive Procedures - Starter project from the Fractals Activity gallery at https://sailctm.slnova.org/mgriza/?open=381465

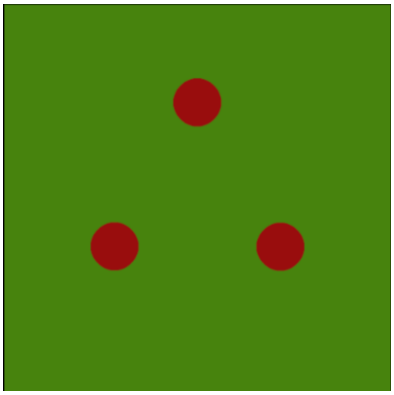
The project starts with a working setup button that resets the terrain color and recursion level data box, as well a draw button that calls the procedure on the Turtle page, which draws the first level of recursion by stamping. There is already some code on the Turtle page that you might want to use and add to.

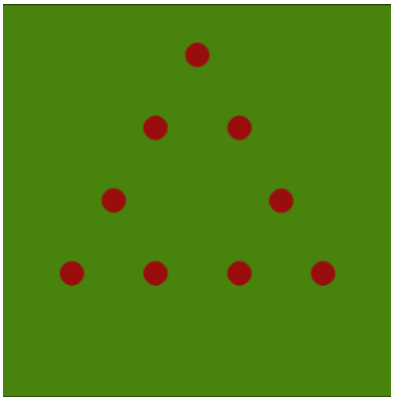
Recursive Idea

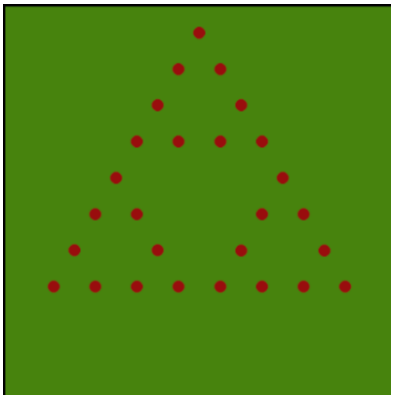
The recurse button is meant to increase the level of "depth"/ recursion the Sierpinski Triangle has. In this project, we will have one agent perform all the drawing steps. At each location where the agent is, have the agent either stamp (if it reaches the lowest level of recursion) or call the drawing procedures once again, with a decreased level. Note that the size of the stamp is based on the agent's size, so for small triangles the size will have to be adjusted accordingly.

Hints/Tips

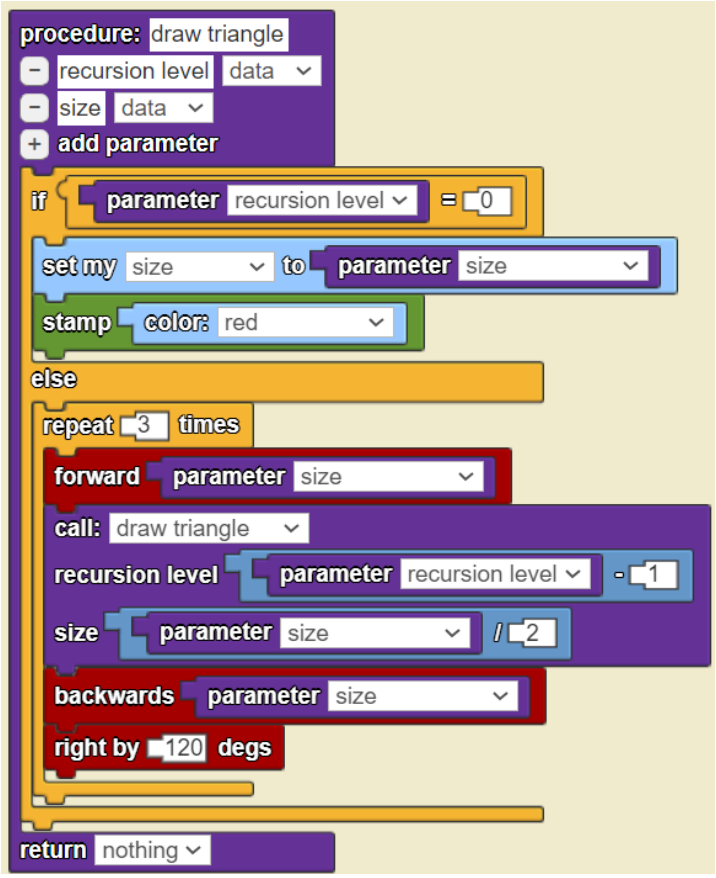
To determine the right distance that new agents should walk from the location they were created from, let's look at the Sierpinski triangles with a smaller amount of recursive levels. Each time the draw is pushed, for the new locations the agent will stamp, the distance it has to walk from the center is **½ of the distance** previously walked.







Solution



Explanation: The function "draw triangle" takes in two parameters: the level of recursion (which you can think of as the number of levels left), and the size. At each point in the function, the agent sits at a specific location and acts in terms of the recursion level value. If the value is 0, it draws (by stamping), with the updated size it should have. When talking about recursion, we refer to the drawing step as the base case, which signals the stopping point of that thread. If however, the recursion value is bigger than 0, the agent draws another triangle at that location, by calling the function again, with updated parameters. The recursion level is decreased by 1, because we get closer to the base case, and the size gets halved because we will require smaller stamps.