

Using Noise to Generate Artistic Textures and Patterns

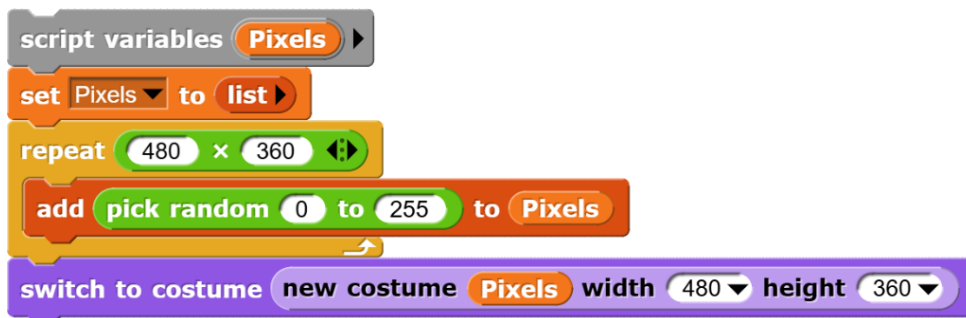
Claude Shannon and Warren Weaver developed what became known as information theory just after the second world war. Their key insight was that information could be measured mathematically, distinguishing the signal portion of a message that carried information from noise that degraded the signal. Their work primarily focused on telephone communications in which random fluctuations in voltage made the speaker more difficult to understand. Their book, *A Mathematical Theory of Communication* (1949), became one of the most influential and cited scientific works of all time. Scientific American described it as the “Magna Carta of the Information Age.”

In Shannon and Weaver's information theory, noise is anything that interferes with or changes a message as it travels from sender to receiver. This could consist of static on a telephone line or a smudge on a photograph. Shannon and Weaver demonstrated that noise is predictable and measurable.

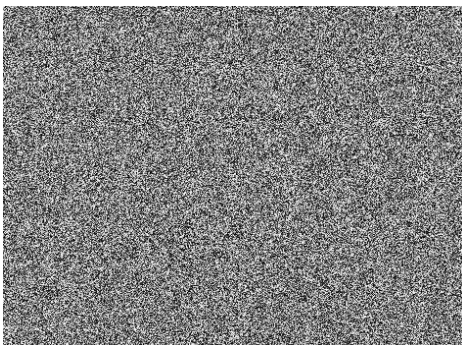
Many artists in the twentieth century explored randomness, chance, and systematic processes in ways that anticipate some of these concepts. Impressionists deliberately introduced what traditional painters might have called noise - visible brushstrokes, color mixing on the canvas rather than the palette, and atmospheric effects. They realized that this apparent imperfection could actually carry more information about light, movement, and sensory experience than highly polished academic paintings

Bernat Romagosa explores some of these concepts related to the relationship between noise and artistic patterns in a forthcoming book on Snap! algorithms. In the mid-1980s Ken Perlin worked for a company that developed computer-generated imagery (CGI) for the Disney film, *Tron*. This was the first movie that made extensive use of CGI through the entire film. Perlin used noise to generate textures for the movie and received an Oscar for Technical Achievement for this accomplishment.

Romagosa's noise generator uses fractals (discussed in greater depth in the book) to create random patterns. His initial noise generator uses a traditional loop to create a costume the size of the stage that has each pixel filled with a random number from 0 to 255.



The result is a gray speckled block in which the darkness or lightness of each dot depends on the number (0 to 255) assigned to that pixel.



Once the basic principle for generating noise is understood, Romagosa introduces Snap! HyperBlocks. HyperBlocks can be used to perform repetitive actions more quickly and efficiently through use of matrix operations. The **Reshape** block can be used to quickly generate a matrix of any size and dimension.



	A	B	C	D
3				
1	0	0	0	0
2	0	0	0	0
3	0	0	0	0

The **Reshape** block can be combined with the **Pick Random** block to fill a matrix with random numbers.

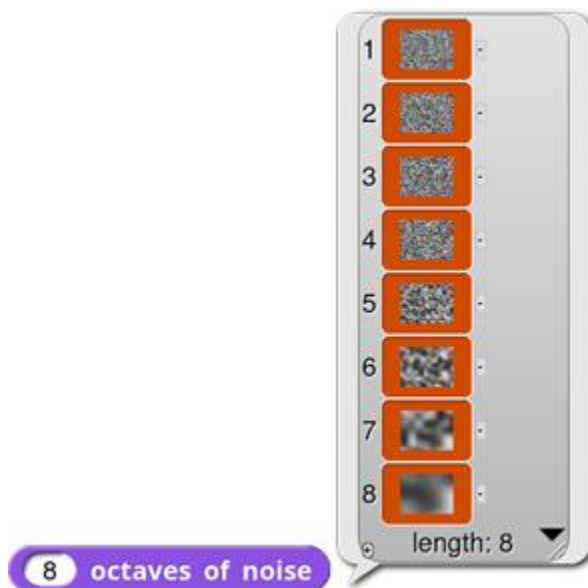


	A	B	C	D
3				
1	239	23	218	208
2	19	216	8	169
3	165	164	238	212

This method can then be used to create a costume the size of the stage filled with random noise.



Romagosa uses an extension of this technique to stamp successive layers of noise that he describes as “octaves of noise” – possibly foreshadowing later use of noise to generate music.

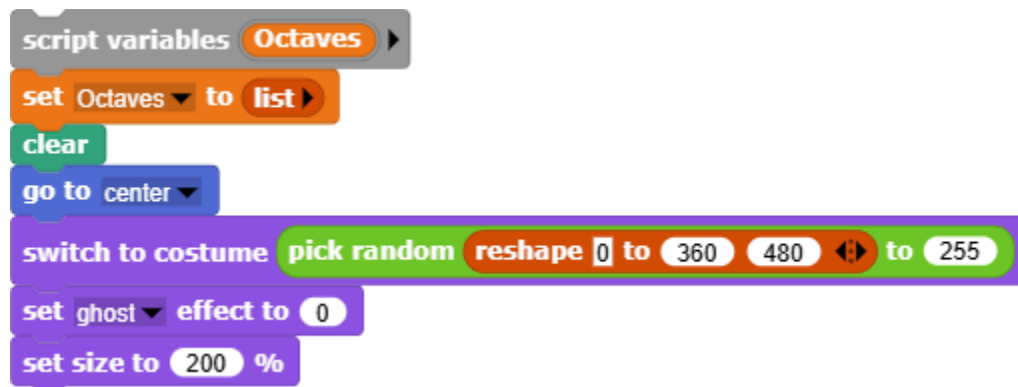


1	
2	
3	
4	
5	
6	
7	
8	

length: 8

8 octaves of noise

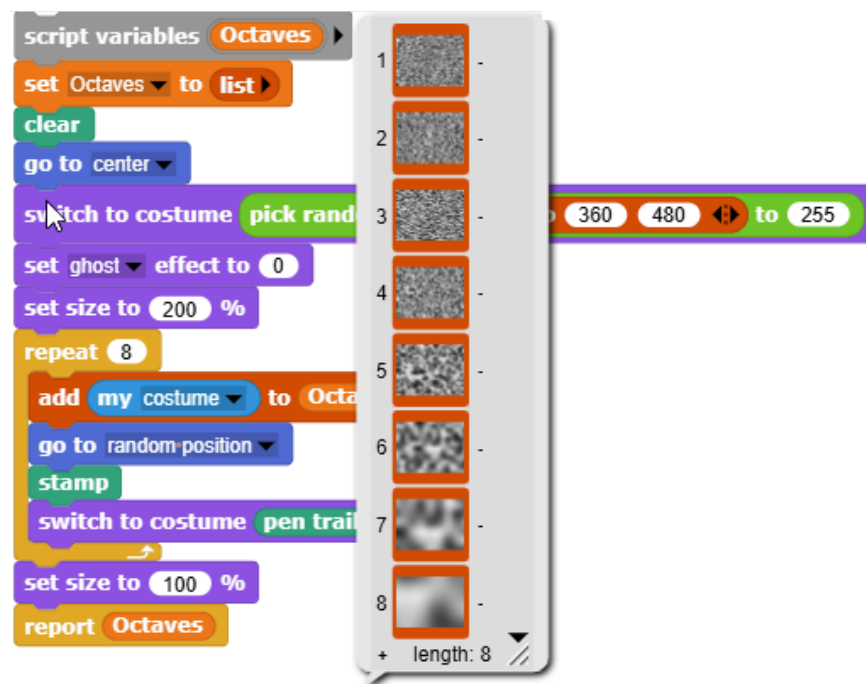
The first portion of the function that Romagosa constructs consists of setup and initialization.



The following code is the heart of the function. It creates the successive layers of noise which Romagosa describes as “Octaves of Noise.”



If the setup code and loop are combined, the function reports the following list of images.



Once the code is tested, it can be used to create a procedure in which the input is used as a variable to determine the number of iterations of the loop.

8 Octaves of Noise

This function is used to create a Fractal Noise generator. It is employed in an initial setup procedure.

```

script variables octaves ▶
set octaves ▼ to 8 Octaves of Noise
set size to 100 %
set ghost ▼ effect to 50

```

The setup code is once again followed by a loop that is the heart of the function.

```

for each Octave in octaves
  switch to costume Octave
  stamp
  set ghost ▼ effect to 0
  switch to costume pen trails
  clear
  report my costume ▼

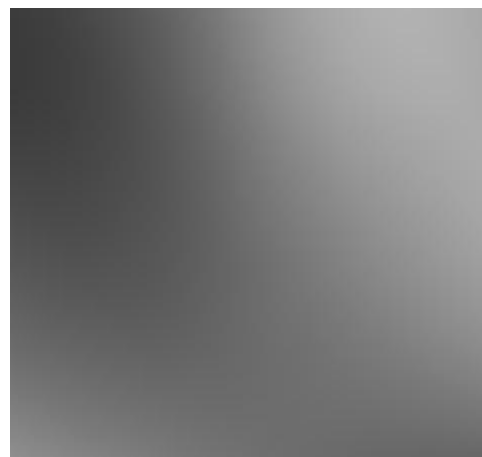
```

The final function wrapped in a warp block generates the type of image shown on the right.

```

+Fractal+Noise+of+ Octave Count +Octaves+
warp
  script variables octaves ▶
  set octaves ▼ to Octave Count Octaves of Noise
  set size to 100 %
  set ghost ▼ effect to 50
  for each Octave in octaves
    switch to costume Octave
    stamp
    set ghost ▼ effect to 0
    switch to costume pen trails
    clear
  report my costume ▼

```



The previous **Fractal Noise** function works. Romagosa then introduces another method that is better aligned with professional computer science guidelines. The second method achieves a similar result but uses clones. This method follows recommended functional programming practice but entails a greater cognitive cost to understand. For that reason, we are providing the function as a resource on the class web site.

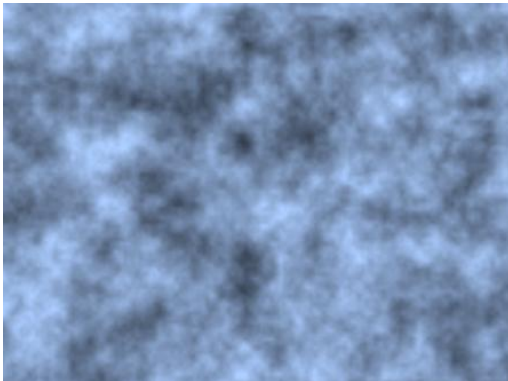


If you have an interest in the way in which this was achieved, you can examine the code or read Romagosa's explanation in Chapter 3 of his book on Snap! algorithms.

The following method will work either with the original Fractal Noise function described above or the new function involving clones.

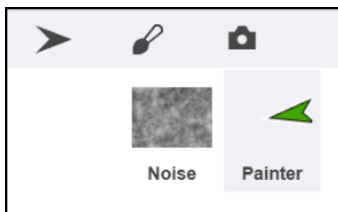


This code maps desired ratios of red, green, and blue onto the corresponding RGB channels of the fractal noise costume. The result is this type of texture.



Painting Designs on a Canvas Created with Noise

In the unit on emulation of impressionist paintings, we used one sprite as a paintbrush to draw designs on an image on a second sprite. Romagosa uses a similar strategy to draw designs on a canvas created with fractal noise. The sprite that employs noise to generate a canvas is called *Noise* and the second sprite that paints on the canvas is called *Painter*.



A fractal noise canvas is created by the *Noise* sprite in the following manner.

```
go to x: 0 y: 0
point in direction 90
switch to costume Fractal Noise of 8 Octaves (clone)
set ghost effect to 75
```

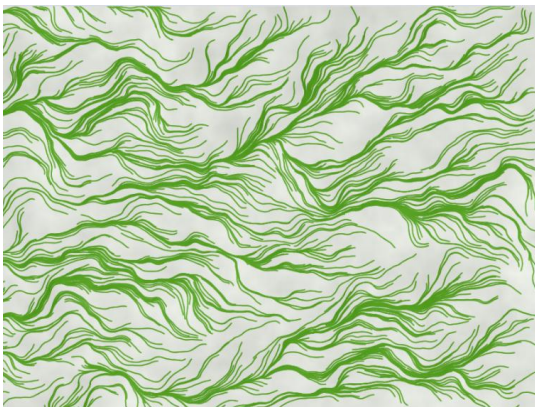
The following setup procedure is used for the *Painter* sprite.

```
set pen color to 
set pen size to 1
pen up
clear
```

Once the setup code is run, the following loop generates a pattern on the noise canvas.

```
repeat 1000
  warp
  go to random-position
  pen down
  repeat 100
    point in direction brightness at myself × 5
    move 1 steps
  pen up
```

The result is an organic pattern that looks like this:



The direction of the patterns drawn are governed by the brightness levels of the noise-generated canvas. By varying factors such as color, brightness, and transparency, any number of textures and patterns ranging from fur to lightning.

Reference

Shannon, Claude E., and Warren Weaver. *The Mathematical Theory of Communication*. University of Illinois Press, 1949.