

# Sierpinski Gasket: An example of a fractal.

```
Clear["Global`*"]
```

The Sierpinski gasket is a simple example of a **fractal**, i.e. a figure in which the same pattern occurs at different scales (down to the infinitesimally small). Here we construct a simple example of fractal, the Sierpinski gasket, in which the repeating figure is a triangle. We will see that we can easily generate and plot the Sierpinski gasket for several levels of iteration, using the recursive functions of *Mathematica*.

We first construct a single triangle. This is a list of three elements (one for each corner) and each of these elements is a list of two numbers (the x and y coordinates).

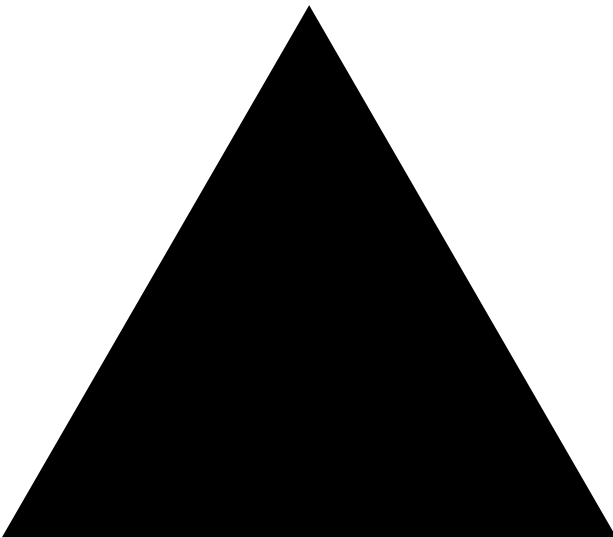
```
In[1]:= starttriangle = {{0.0, 0.0}, {0.5, N[Sqrt[3/4]]}, {1.0, 0.0}}
```

```
Out[1]= {{0., 0.}, {0.5, 0.866025}, {1., 0.}}
```

This is what it looks like:

```
In[2]:= Show[Graphics[Polygon[starttriangle]]]
```

```
Out[2]=
```



The command **Polygon[list]** constructs a graphics element comprising a filled in polygon connecting the points in **list**. This is then converted to a graphics object with the command **Graphics[ ... ]** and displayed with the command **Show[ ... ]**.

We now define a function **triple**, which takes a triangle and makes three triangles out of it in the same overall region.

```
In[3]:= triple[d_] := Module[{d1, d2, d3},  
  d1 = {d[[1]], 0.5 (d[[2]] + d[[1]]), 0.5 (d[[3]] + d[[1]])};  
  d2 = d1 + Table[(d1[[3]] - d1[[1]]), {3}];  
  d3 = d1 + Table[(d1[[2]] - d1[[1]]), {3}];  
  {d1, d2, d3}  
]
```

The argument **d** is a list of three elements (each of which is a list of two numbers, the x and y components) which give the coordinates of the original triangle. **d1** gives the coordinates of a triangle whose sides are half the size of the original triangle, in one corner of the original triangle. **d2** takes the coordinates of **d1** and displaces them halfway along one side of the original triangle, to get a triangle of half the size of the original one (same size as **d1**) in another corner of the original triangle. **d3** gives

the half-size triangle in the third corner of the original triangle.

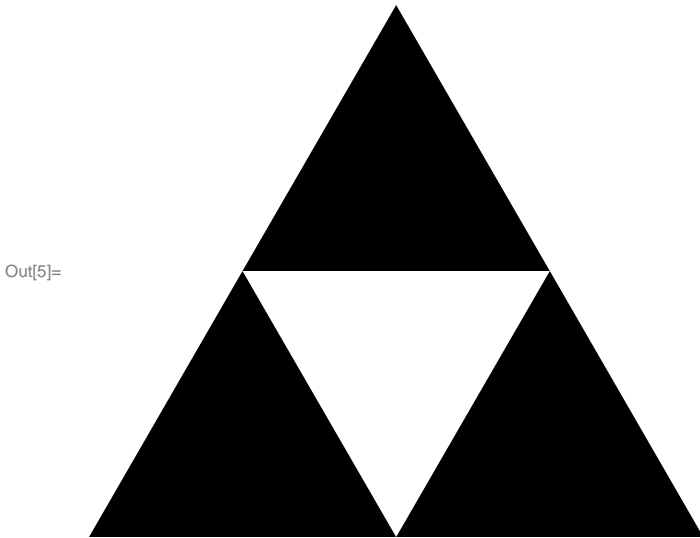
The function **triple** will be the building block out of which we will construct the fractal iteratively. For example, if we act with **triple** on **starttriangle**:

```
In[4]:= threetriangles = triple[starttriangle]
```

```
Out[4]= {{{{0., 0.}, {0.25, 0.433013}, {0.5, 0.}}, {{0.5, 0.}, {0.75, 0.433013}, {1., 0.}},
          {{0.25, 0.433013}, {0.5, 0.866025}, {0.75, 0.433013}}}
```

and plot it

```
In[5]:= Show[Graphics[Polygon /@ threetriangles]]
```



we get three triangles occupying the same overall region (with an empty triangle in the middle), as desired. This, then, is the basic operation: start with one triangle and produce three triangles occupying the same overall region, as shown in the figure immediately above.

The syntax **Polygon /@ threetriangles** is the same as **Map[Polygon, threetriangles]** and means apply the function **Polygon** to each element of the list **threetriangles**.

Now we are going to automate the construction of several iterations of this procedure. This involves acting on a *list* of triangles. We therefore make a list containing just one triangle by putting an extra layer of curly brackets around **starttriangle**.

```
In[7]:= startlist = {starttriangle}
```

```
Out[7]= {{{{0., 0.}, {0.5, 0.866025}, {1., 0.}}}}
```

We **Map** (i.e. apply) the function **triple** to each element of **startlist** (there's actually only one element so far, so **Map** is not yet necessary, but we do it anyway because it *will* be necessary in subsequent iterations and we want to use the same procedure for each iteration). Finally, we remove one layer of curly brackets with **Flatten[ ..., 1]** to again get a list of triangles on which we can act again with our function **triple**.

```
In[8]:= list2 = Flatten[Map[triple, startlist], 1]
```

```
Out[8]= {{{{0., 0.}, {0.25, 0.433013}, {0.5, 0.}}, {{0.5, 0.}, {0.75, 0.433013}, {1., 0.}},
          {{0.25, 0.433013}, {0.5, 0.866025}, {0.75, 0.433013}}}
```

This produces the list of three triangles that we plotted above. Note that we could alternatively remove the outer layer of curly brackets by taking the first element (there is only one element so this doesn't do anything else apart from removing the brackets):

```
In[9]:= list2 = Map[triple, startlist][[1]]
```

```
Out[9]= {{{{0., 0.}, {0.25, 0.433013}, {0.5, 0.}}, {{0.5, 0.}, {0.75, 0.433013}, {1., 0.}},
          {{0.25, 0.433013}, {0.5, 0.866025}, {0.75, 0.433013}}}}
```

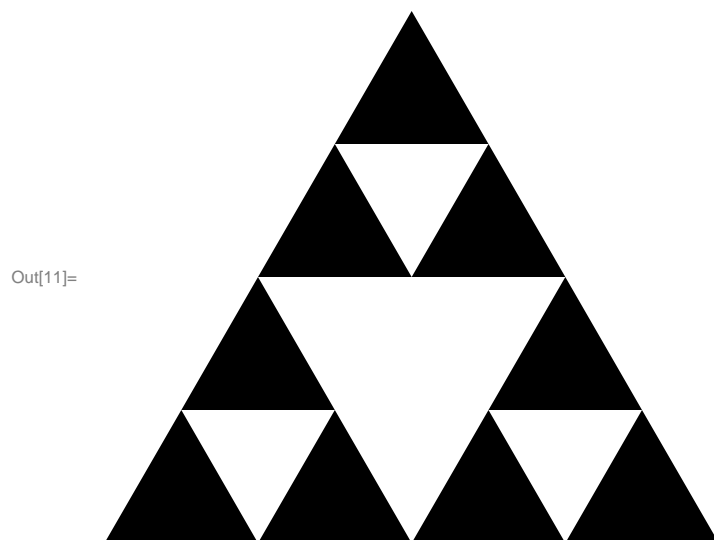
Next we iterate again:

```
In[10]:= list3 = Flatten[Map[triple, list2], 1]
```

```
Out[10]= {{{{0., 0.}, {0.125, 0.216506}, {0.25, 0.}}, {{0.25, 0.}, {0.375, 0.216506}, {0.5, 0.}},
          {{0.125, 0.216506}, {0.25, 0.433013}, {0.375, 0.216506}},
          {{0.5, 0.}, {0.625, 0.216506}, {0.75, 0.}}, {{0.75, 0.}, {0.875, 0.216506}, {1., 0.}},
          {{0.625, 0.216506}, {0.75, 0.433013}, {0.875, 0.216506}},
          {{0.25, 0.433013}, {0.375, 0.649519}, {0.5, 0.433013}},
          {{0.5, 0.433013}, {0.625, 0.649519}, {0.75, 0.433013}},
          {{0.375, 0.649519}, {0.5, 0.866025}, {0.625, 0.649519}}}}
```

which produces 9 triangles as shown:

```
In[11]:= Show[Graphics[Polygon/@list3]]
```



Now we define functions with which we will automate the construction of the Sierpinski gasket down to a specified scale. The function **iterate**, below, does one level of transformation, and this is repeated **n** times by the function **newlist**, starting from the single triangle in **startlist**.

```
In[12]:= iterate[list_] := Flatten[Map[triple, list], 1]
```

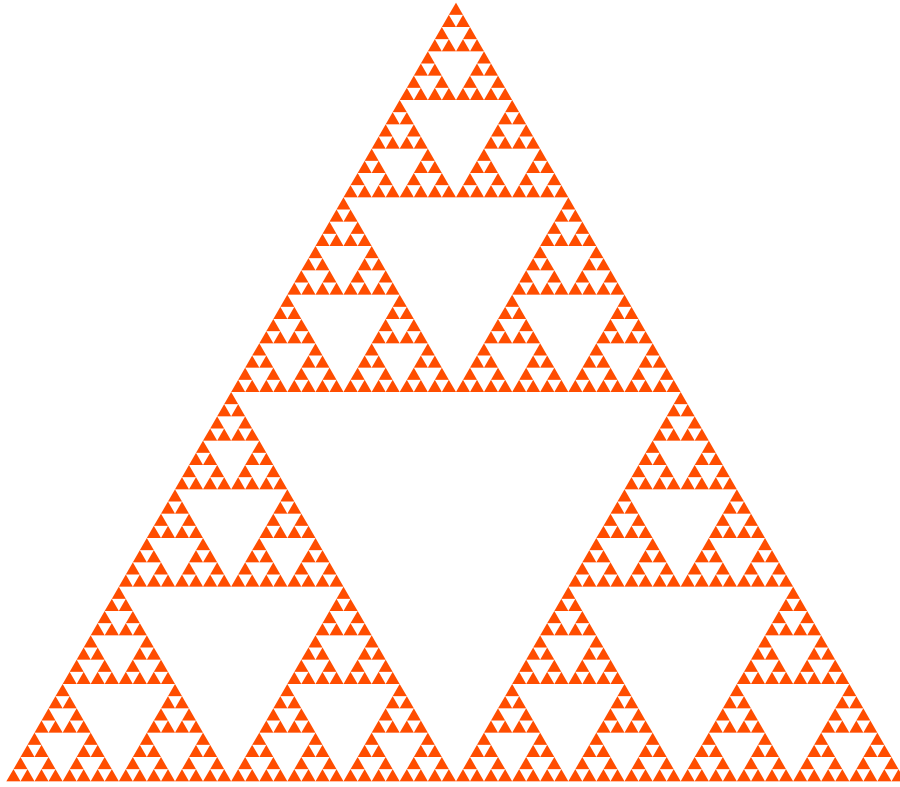
```
In[13]:= newlist[n_] := Nest[iterate, startlist, n];
```

In the *Mathematica* function **Nest[f, expr, n]** the function **f** is applied **n** times to **expr**.

The result of **newlist** is a list of triangles. This can be converted to a graphics object using the **Graphics** command, and then displayed with **Show**. As an example, we show the result of 6 levels of iteration which produces  $3^6 = 729$  triangles.

```
In[14]:= Show[Graphics[{Hue[0.05], Polygon /@ newlist[6]}]]
```

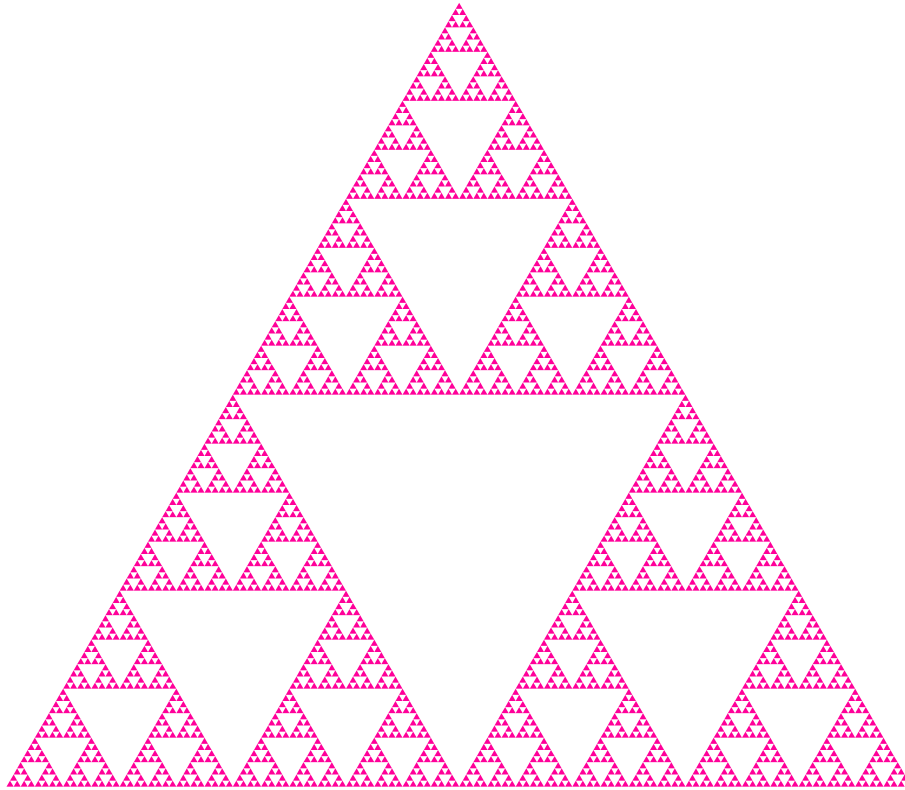
Out[14]=



Finally 7 levels of iteration:

```
In[15]:= Show[Graphics[{Hue[0.9], Polygon /@ newlist[7]}]]
```

```
Out[15]=
```



Note that these figures have the property that one can take a section of it, blow it up by an appropriate scale factor, and it looks like the original figure. This "**scale invariance**" is a characteristic feature of a fractal.

A basic way to characterize a fractal is by the **fractal dimension**  $d_s$ , also called the Hausdorff dimension. To define it for the Sierpinski gasket, let the length of the side of the smallest triangle be  $\epsilon$  and the overall length of a side of the triangular figure be  $L$ . Then, the fractal dimension of the shaded region is defined in terms of its area  $A$  by the relation

$$\frac{A}{A_\epsilon} = \left(\frac{L}{\epsilon}\right)^{d_s},$$

where  $A_\epsilon$  is the area of a single shaded triangle at the smallest scale (i.e. with side  $\epsilon$ ). Clearly if the whole region were shaded, we would have  $d_s = 2$ , the spatial dimension. To see this, consider, for example, the second figure in this notebook. If the central triangular region were also shaded, (so the *whole* figure would be shaded) then the large triangle of side  $L$  would be made up of 4 smaller triangles, each of size  $\epsilon = L/2$ , and so we would have  $A/A_\epsilon = 4$ , and  $L/\epsilon = 2$ , so  $4 = 2^{d_s}$ , which gives  $d_s = 2$ . Hence, **for regular solids, the fractal dimension is equal to the space dimension**. This is clearly desirable; our definition of fractal dimension should reproduce what we expect in simple cases of integer dimension.

However, for the gasket, only three of the four regions in the figure are shaded. In other words, doubling  $L/\epsilon$  increases the area by a factor of 3 (not 4), so

$$3 = 2^{d_s}$$

which gives

$$d_s = \frac{\ln 3}{\ln 2} = 1.58496 \dots$$

We see that the Sierpinski gasket has a fractal dimension in between that of the one-dimensional line and a two-dimensional triangle.

Iterating the construction of the gasket, we recover the same value for  $d_s$  since, at each iteration, the ratio of the length of a side of the figure to the side of the smallest triangle increases by a factor of 2, while the ratio of the area of the figure to the area of the smallest triangle increases by a factor of 3 (rather than 4 which would be the result if all the area in the larger triangle were included.)

If the fractal dimension of the object is less than the space dimension, the areal density of the object (i.e. the size of the shaded area per unit area of the plane) tends to zero as the number of iterations tends to infinity. This is because the areal density is given by

$$\rho = \frac{A}{\frac{1}{2} L^d} = A_\epsilon \left( \frac{L}{\epsilon} \right)^{d_s} \frac{1}{\frac{1}{2} L^d} = \left( \frac{L}{\epsilon} \right)^{d_s - d} \frac{A_\epsilon}{\frac{1}{2} \epsilon^d} = \left( \frac{L}{\epsilon} \right)^{d_s - d}$$

since  $A_\epsilon = \frac{1}{2} \epsilon^d$ . Hence the fraction of the area which is shaded equals  $(L/\epsilon)^{d_s - d}$ . Since  $L/\epsilon = 2^k$  where  $k$  is the number of iterations, and  $d_s$  is given above, **the fraction of area which is shaded is just  $(3/4)^k$ . This tends to zero as  $k \rightarrow \infty$ .** Qualitatively this can be seen in the figures since, the fraction of white increases as more iterations are performed.