Fractals

There are various definitions of what fractals are and there are various methods to construct classes of fractals. One thing that all definitions have in common is that fractals remain equally complicated no matter how much you zoom in on them.

All constructions of fractals contain an element of repetition to create the fine details that characterize them. As a first example, perform the following recursive procedure:

(1) Start with a solid triangle, and remove the inner triangle whose corner points are the midpoints of the edges of the large triangle. You finish this step with three solid triangles that are connected on the corners.
(2) Apply the procedure of the first step to each of the remaining triangles.
(3) Repeat the last step infinitely many times.

The resulting object is called the *Sierpinski triangle* or *Sierpinski gasket* S. Notice that it consists of three scaled down copies of itself. You may ask yourself what the area of this object is. Can you figure it out?

Contracting Similarities. In order to understand this type of fractals it is important to know about contracting similarities. The easiest example of such a function $f\colon \mathbb{E}^2 \to \mathbb{E}^2$ is that of a *dilatation.* (Note that even though we can make fractals as subsets of $\mathbb{E}^n$ for an dimension n, we will restrict ourselves to the case $n = 2$ for this project.) The dilatation $\delta_{K,\lambda}$ with center K and dilatation factor λ sends K to K and sends a point $P \neq K$ to a point P' on the ray $\overrightarrow{KP}$ such that $|KP'| = \lambda|KP|$. A dilatation is called *contracting* or *contractive* if $\lambda < 1$. In that case we also say that the dilatation is contracting with *contraction factor* $\frac{1}{\lambda}$.

(1) Show that this implies that for all points P and Q in $\mathbb{E}^2$,
$$|\delta_{K,\lambda}(P)\delta_{K,\lambda}(Q)| = \lambda|PQ|.$$
(2) Describe the dilatations that send the Sierpinski triangle to each one of its three smaller copies of itself. Note that if these dilatations are δ_1, δ_2, δ_3, then we have that
$$S = \delta_1 S \cup \delta_2 S \cup \delta_3 S.$$

More general contracting similarities consist of the composition of a contracting dilatation with a rotation around the center of the dilatation. (So you first move a point P toward the center point K and then rotate around K.) All contracting similarities are of this form.

Any set that can be described in this way as a union of scaled down copies of itself is called *self-similar.* Give some examples of other self-similar sets (many of them don't look as interesting as the Sieprinski triangle; for example, can you show that the square is self-simlar?).

IFS-systems. An IFS-system (iterated function system) is a set of contracting similarities $\{F_1, F_2, \ldots, F_n\}$. An IFS-system gives rise to a unique set S such that

$$S = \bigcup_{i=1}^{n} F_i(S).$$

This set S is called the invariant set of the IFS-system. (If you like, you can derive this as part of your project. Then I will give you a couple of questions to lead you through the proof.)

You can find the unique set S by the following observation: if you start with an arbitrary closed and bounded subset X of $\mathbb{E}^2$ (closed just means that the set contains its own boundary), and you keep repeatedly taking the union of its images under all the maps, $X_1 = \bigcup_{i=1}^{n} F_i(X)$, $X_2 = \bigcup_{i=1}^{n} F_i(X_1)$, $X_3 = \bigcup_{i=1}^{n} F_i(X_2)$, etc., you can show that

$$\lim_{n\to\infty} X_n = S.$$

A way to obtain a series of objects that converges fairly quickly to S is by starting with the set of all the fixed points (centers) of the contracting operations F_i in the IFS-system. You know that they have to be part of S, and so every image we create will be a part of the invariant set S. There are computer applets that use this feature to draw quick approximations of the invariant set of an IFS-system. One of these applets is fractalina and can be used online at

http://math.bu.edu/DYSYS/applets/fractalina.html

(Fractalina actually just takes one point P and then keeps drawing $P_i = F_{m_i}(P_{i-1})$, where m_i is a randomly chosen index from the set $\{1, \ldots, n\}$; there are instructions for this applet at

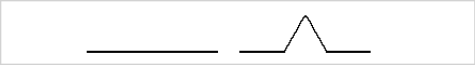
http://math.bu.edu/DYSYS/applets/fractalina_more.html

There are also a couple of pre-set fractals that you can run to get an idea of how it works. When you want to create a new IFS-system, you give the coordinates of the center, the angle, and the contraction factor (not the dilatation factor).

Chaos Game. In order to understand what actually happens when you do this, it is very useful to play the *chaos game*. This helps you to think through which part of the invariant set is created by each sequence of function applications. The applet is online at http://math.bu.edu/DYSYS/applets/chaos-game.html For your project describe the way to solve this game and what this tells us about the fractal.

The Dimension of an IFS-Fractal. Have you been able to figure out what the area of the Sierpinski triangle is? If you haven't yet: create a formula for each one of unions of triangles that you have created along the way and then take the limit for $n \to \infty$.

Just to show you how funny things can be, consider the Koch island. It is obtained by starting with the boundary of an equilateral triangle. Then place an equilateral triangle with side lengths 1/3 of the original side length at the center of each side and finally remove the middle third segment. So on each side you perform the following operation:



Then continue to do this to each side of the newly formed object and repeat ad infinitum.

(1) What is the length of the boundary of the island?
(2) What is the area of the island?

For variations on the construction of the Koch island and an applet to create other interesting fractal islands, go to:

http://www.stilldreamer.com/mathematics/koch_island/

There are various ways to formalize the concept of *dimension*, and one is that the dimension of an object should be related to how its "size" scales when you shrink or enlarge it. Another way to think about it is to say that the dimension is the level at which it makes sense to measure an object's size. If you ask for the volume of a square, then that is zero, because the dimension of a square is less than 3, and if you ask for the length of a square it is infinity because its dimension is higher than one. There is always precisely one number at which it makes sense to calculate an objects size.

This can be turned into a precise mathematical concept in more than one way. Read about this on:

http://en.wikipedia.org/wiki/Fractal_dimension
http://math.rice.edu/ lanius/fractals/dim.html

and then create some new fractals with fractalina and calculate their dimensions. Also, determine both the dimension of the Koch island and the dimension of its boundary. There are more sophisticated ways to calculate the dimension of an object that also apply to objects that are not self-similar, but they require more math background.

Become Creative with Fractalina. You have already seen that every IFS-system gives rise to a unique self-similar set. Now you may ask yourself which sets you could get in this way. The *Collage Theorem* gives the answer: you can create an image that is arbitrarily close to any given image if you allow yourself larger and larger sets of IFS-systems. The good news is of course that for any image we have only finite precision, so you can pretty much make any image! The question is then of course: how? Here, the Collage Theorem gives the answer again, and you could probably derive it yourself if you thought about it for a moment. What you need to do is cover your image by scaled down copies of itself (the copies are allowed to overlap) and then take the collection of maps that would send your image to each of these scaled down copies. With this idea in mind, create some interesting images in Fractalina. For more on the Collage Theorem, see

http://www.cut-the-knot.org/ctk/ifs.shtml

(You may also want to put it into Google and find more references.)

IFS-Systems with only two elements. With only two functions, you can still create a very interesting set of fractals. Experiment with this and report on any results you find.