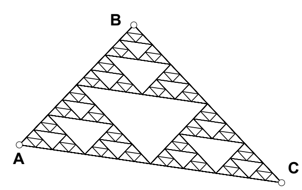
Dynamical systems, fractals, and chaos are relatively new fields of mathematics. Many of the most important concepts were not discovered until high-speed computers made it easier to visualize iterative mappings.

A *dynamical system* is one that changes over time, with its state at any moment computed from the state of the system at the previous moment. As the dynamical system evolves, sometimes its limiting state may approaches a specific fixed state (a *point attractor*), or may alternate among a set of fixed states (a *periodic attractor*). A limiting state that is neither a point attractor nor a periodic attractor is called a *strange attractor*.

SIERPIŃSKI'S TRIANGLE AS AN ATTRACTOR

1. Open the sketch **Chaos1.gsp** in the folder **Supplemental Activities | Chaos**. Select the parameter *depth* and increase it using the **+** key. Be careful not to go too far. A depth of 7 should take the image to the limit of the screen resolution.



This figure may look familiar to you. It is Sierpiński's triangle, a fractal. It is self-similar and infinitely complex. As you will see, it can also act as a strange attractor.

Q1 Let p be the perimeter of the of the midpoint triangle added in the first iteration. What is the sum of the perimeters of the three triangles added in the second iteration? What is the sum of the perimeters of the triangles added in the third iteration? As the number of iterations grows, is there a limit to the sum of the perimeters added at each iteration?

Repelling

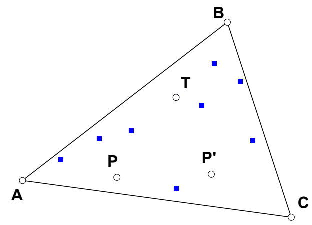
2. Open the Repel page of **Chaos1.gsp**. Point P' is the dilation of point P by a factor of 2 about the nearest vertex.

The *orbit* of a object is the set of all iterated images of that object.

Q2 If this dilation is defined as an iteration rule, and point P is on the interior of the triangle, do you think the orbit will eventually leave the triangle? If so, how many iterations can remain in the triangle before the next iteration leaves?

3. Select point P and the parameter *depth*. Hold the Shift key while choosing **Transform | Iterate By Depth**. Map point P to P'. Click the Iterate button in the dialog box to create the iteration.

4. Construct the terminal point of the iterated point image by selecting the image and choosing **Transform | Terminal Point**. Label this point *T*.



Q3 Drag point *P* around to test your conjecture from Q2. Adjust the depth and see how many iterations you can keep inside the triangle.

Q4 Once an iterated point has left the triangle, is it possible for some subsequent iteration to return to the triangle?

In fact, Sierpiński's triangle is a repeller for this iteration rule. What that means is that any point that is actually on the fractal will have an orbit that never leaves the fractal. Any other point, however near, will eventually be cast out.

Attracting

Since the iteration rule avoids the fractal, perhaps running this iteration in reverse would attract the orbit to the fractal. A given orbit point is created by doubling the distance from the nearest vertex. We can get back to the previous point by halving the distance. However, there is no way of knowing which vertex was the center of dilation, so you'll pick a vertex at random and halve the distance to that vertex.

5. Open the Attract1 page of **Chaos1.gsp**. The image of point *P* has three possible locations, depending on the position of point *R*. Drag point *R* to see each of the possible locations of *P'*.

You will iterate *P* to *P'*. For each iteration, you will randomly choose the vertex toward which to dilate.

6. Select points *P* and *R* and the parameter *depth*. While holding the Shift key, choose **Transform | Iterate To Depth**. Map *P* to *P'* and *R* to itself. In order to make point *R* choose a new random vertex for each iteration, choose **To New Random Locations** from the Structure menu of the dialog box.

To make the iterated image points small, select them and choose **Display | Line Width | Dashed.**

7. Make the iterated image points small.

Q5 The initial value of *depth* is 100. Describe the pattern formed by the iterated points.

Q6 Select the depth parameter and press the + key to increase the depth to about 3000. Describe the pattern now.

Q7 Drag point P to change the initial conditions of the iteration. What changes can you observe in the pattern?

Q8 At any particular step of the iteration, can you predict where the next point will be? Does this iteration have a point attractor, a periodic attractor, or a strange attractor? Explain your answer.

To change the properties, select the iterated image and choose **Edit | Properties**. On the Iteration panel, click the button To Same Location Relative to Original.

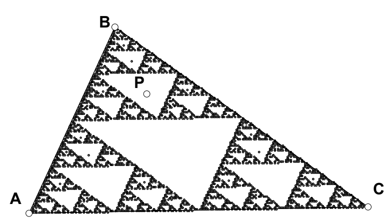
Q9 Change the properties of the iterated image so that the iterated images of point *R* move to the same location relative to the original, rather than to random locations. How does this change the image? Does it now have a point attractor, a periodic attractor, or a strange attractor? What happens if you drag point *R*?

With a small change, you can produce the same pattern with far fewer iterations.

8. Open the Attract2 page of **Chaos1.gsp**. On this page all three possible image points have been constructed.

9. Select point *P* and the parameter *depth*. While holding the Shift key, choose **Transform | Iterate To Depth**. Map *P* to one of the image points. Choose **Add New Map** from the Structure menu, and map *P* to one of the other image points. Choose **Add New Map** again, and map *P* to the third point. Finally, click Iterate.

10. Make the iterated points small. Hide the initial images of *P*, the three black points. Increase the depth—but do not enter too high a value, because the number of iterated objects increases exponentially.



Q10 Are all of the iterated points on Sierpiński's triangle? Are any of them? Try starting with point *P* near the middle of the triangle, clearly not on the fractal.

Summarize

You used two mappings here. The first mapping repels points from Sierpiński's triangle, and the second and third attract points. In either case, given a starting point on or within a triangle, these two behaviors hold true:

A point that begins on the fractal will stay on the fractal.

A point that is not on the fractal will never reach the fractal (though the attraction mapping will cause it to approach the fractal).

Q11 Prove or demonstrate both statements. It is sufficient to show that for both mappings, if a point is on the fractal, so is the next point(s).

MIRA, JULIA, AND MANDELBROT SETS

Mira Sets

1. Open the sketch **Chaos2.gsp** in the folder **Supplemental Activities | Chaos**.

The Mira page contains an iteration based on a function f(x), using these equations:

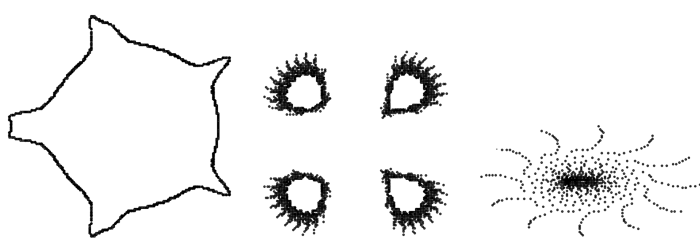
$$f(x) = ax + \frac{2x^2 - 2ax^2}{1 + x^2}, \quad x' = y + f(x), \quad y' = f(x') - bx$$

Point P' is the first iterated image of P, plotted at (x', y'). Point T is the terminal point of the iteration.

If you select either *a* or *b*, pressing the **+** or **–** key will change the parameter's value by 0.01.

2. Experiment with small changes in parameters a and b. It is interesting to see that very small changes in the parameters can produce radical changes in the orbit. Drag point P to see what effect the initial seed has.

Q1 The initial settings for this sketch are a = 0.20, b = 1.00, and P (8.61, 2.97). With these settings, the orbit appears to be stable. As you experiment, record the settings for interesting figures you find. Which settings attract the orbit to a single point? Which make it periodically go between two or more points? Which send it out of range?



Julia Sets

The Mira mapping is actually an offshoot from some research into the behavior of elementary atomic particles. The next page shows a Julia set, based on this mapping:

$$x' = x^2 - y^2 + a, \qquad y' = 2xy + b$$

3. Open the Julia Forward page of **Chaos2.gsp**. The construction is the same as that on the Mira page, but uses the equations above to define x' and y'.

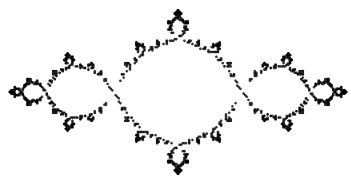
Q2 The patterns are not generally as striking, but there are similar results. When is the orbit attracted to a single point? When does it jump between two or more points? When does it fly off the screen? Again, record settings of parameters a and b.

It turns out that this mapping repels a point from a Julia fractal. As before, the fractal is an attractor for the inverse of the mapping, as defined by the equations below. Notice that there are two values for x', and a y' corresponding to each step of the iteration. When performing the iteration backward, the path branches and the number of iterated points grows exponentially, swarming around the Julia fractal.

$$x' = \pm\sqrt{\frac{1}{2}\left[x - a + \sqrt{(x-a)^2 + (y-b)^2}\right]} \qquad y' = \frac{y-b}{2x'}$$

4. Open the Julia Back page of **Chaos2.gsp**.

Experiment by dragging point P and changing the parameters a and b. For certain settings, the fractal is in a single connected piece. For others, it is broken into discrete points.



Q3 What is the effect of dragging the starting point P across the screen? Does it change the overall shape? Does it alter the detail?

Q4 The abbreviation $J(a, b)$ is used to refer to the fractal that comes from some specific settings for a and b. For each of the following fractals, describe it, identify its symmetries, and tell whether or not it is connected: $J(-1, 0)$, $J(0.3, 0.8)$, $J(0, 0)$, $J(0, 1)$.

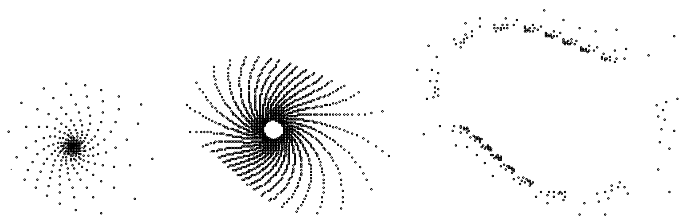
Q5 What general statement can be made about Julia fractals where $b = 0$?

The Mandelbrot Set

Benoit Mandelbrot divided the Julia fractals into two sets: those that are connected, and those that are disconnected. The parameters of the fractal can be imagined as point coordinates. Consider the set of points (a, b) such that $J(a, b)$ is a connected fractal. That set of points is itself a fractal, the Mandelbrot set. There is a helpful shortcut for determining which points belong to the Mandelbrot set. Run the Julia mapping forward, using (a, b) as a starting point. If the orbit diverges, the fractal $J(a, b)$ is disconnected.

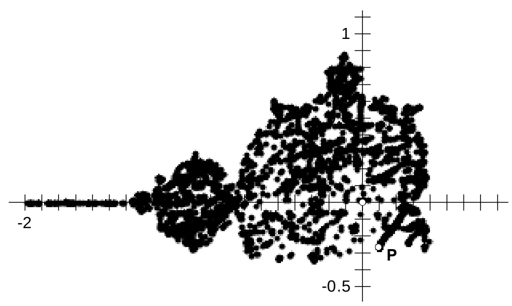
5. Open the Julia Forward page of **Chaos2.gsp** again. Select parameter a. Choose **Edit Parameter** from the Edit menu. Delete the existing value in the Calculator, and click in the sketch on measurement x_P. Use the same procedure to set parameter b to y_P.

As you drag point *P* across the screen, the orbit changes rapidly. When it clearly converges on one or more points, point *P* is in the Mandelbrot set. Drag *P* until the orbit appears to explode. That point is near the boundary of the set.



6. Select point *T* and the origin point. Choose **Measure | Coordinate Distance**. Hide *T* and the iterated images. Reduce the *depth* to about 50.
7. Choose the custom tool **Less than 2**. Click on point *P* and then on the measurement that you created in the previous step.
8. Switch back to the **Arrow** tool and drag point *P*.

Point *P* now leaves a black trail whenever *T* is within 2 of the origin. Once *T* has exceeded that radius, it can never return. If the orbit is still within 2 after 50 iterations, the seed (point *P*) probably belongs to the Mandelbrot set. Drag point *P* across the screen to color a rough approximation of the set.



EXPLORE MORE

Even such a rough graphical rendering of the Mandelbrot set was not possible until the development of high-speed computers, but it was possible to draw conclusions about the orbits of certain individual points. Calculate the images of point (0, 0) after the first and second iterations. Do the same for the point (-1, 0). Prove that these points are in the set. Try some other points on the *x*-axis.

CHAOS

Objective: Students create various fractal designs involving strange attractors. In the first part they create the Sierpiński triangle, and in the second they create and explore Mira, Julia, and Mandelbrot fractals.

Prerequisites: Students will find this activity easier if they have already encountered some of the concepts and terminology of mathematical iteration (for instance, pre-image, image, seed, orbit, and mapping).

Sketchpad Proficiency: Intermediate. Students create iterations, but with the necessary objects prepared in advance.

Class Time: 30–40 minutes for each part. Each part stands on its own. It's probably best to give students a day or two to digest the first part of the activity (the Sierpiński triangle) before moving to the second part of the activity (Mira, Julia, and Mandelbrot sets).

Required Sketches: Chaos1.gsp, Chaos2.gsp

Example Sketch: Chaos1 Work.gsp, Chaos2 Work.gsp

SIERPIŃSKI'S TRIANGLE AS AN ATTRACTOR

Q1 If the perimeter of the first midpoint triangle is p, the three triangles of the second iteration have a combined perimeter of $3p/2$, and the triangles of the third have a combined perimeter of $9p/4$. The terms constitute a geometric sequence with a ratio of 3/2. The sum of the perimeters in the nth iteration is $p\ (3/2)^{n-1}$. These terms increase without limit. (Although the activity does not discuss the concept of fractal dimension, this ratio corresponds to a fractal dimension of $\log 3 / \log 2 \approx 1.58$.)

Repelling

Q2 Conjectures will vary. Students use the actual sketch to investigate these questions in Q3.

Q3 The orbit will stay in or on the triangle only if it begins from a point on the fractal. Drag point P slowly to find a "sweet spot." With a bit of work it is possible to make as many as 15 iterations stay in before the orbit leaves. If you could drag P by moving the mouse less than a pixel at a time, you could get closer to the fractal and keep more iterations inside the triangle.

Q4 Once an iterated point has left the triangle, no subsequent iteration can return to the triangle. Dilating a point outside the triangle about any vertex results in an image point that is farther from the triangle than the pre-image point was.

Attracting

Q5 No pattern is yet visible with only 100 iterations.

Q6 With 3000 iterations, the shape of the Sierpiński triangle is filmy but clearly identifiable. The larger open triangles are quite prominent.

Q7 Dragging P has very little effect on the image. If P is in an area far from the fractal (for instance, in the middle of the largest midpoint triangle), you can identify the first few iterations, because they are near the middle of their respective (successively smaller) triangles. Dragging P has no visible effect beyond these first two or three images.

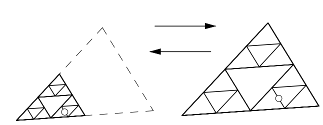
Q8 At any particular step, you cannot predict where the next point will be, because the position of the next point is determined by a random process. As a result, the iteration has a strange attractor.

Q9 By making the position of R consistent from one iteration to the next, the process always picks out the same vertex. The chosen vertex is a point attractor for the iteration. Dragging point R changes which of the three vertices of the triangle serves as the point attractor.

Q10 If you drag point P to a spot that's clearly not on the fractal, it's clear that the first images of P are also not on the fractal. In fact, *none* of the image points lies on the fractal. The fractal is visible only because the image points are increasingly close to it, although never actually on it.

Summarize

Q11 Sierpiński's triangle is self-similar. Dilating it with respect to one of the vertices, by factor 1/2, maps the entire fractal onto itself. Hence, any point on the fractal is mapped to the fractal. Dilating by a factor of 2 maps the nearest of the three sections onto the whole fractal.



MIRA, JULIA, AND MANDELBROT SETS

You may want to introduce students to the shape and properties of the Mandelbrot set before beginning this activity. Several programs, books, and video presentations are available for this.

Mira Sets

Q1 Answers will vary, with each student trying different parameter values and different positions of point P. You may want to have students print their more interesting images. The orbit diverges when a or b is greater than one. Experiment with the settings below for some interesting images. With some settings, the position of point P has little effect on the overall image. With others, the effect is significant.

a	b
-0.05	1.00
-0.01	0.99
-0.75	0.92
0.99	0.99
0.94	0.91

Q2 Try the settings below and experiment with others. The position of point P has a great deal of influence on whether the orbit converges, but it does not appear to change the attractor point(s).

a	b
0.32	0.39
0.25	-0.44
-0.97	0.23

Q3 When you drag point P, the overall shape does not change, but the detail does.

Q4 All Julia fractals have 180° rotational symmetry on the origin.

$J(-1, 0)$	reflection symmetry on both axes, connected
$J(0.3, 0.8)$	disconnected
$J(0, 0)$	a circle centered on origin, unlimited number of rotation and reflection symmetries, connected
$J(0, 1)$	connected

Q5 When $b = 0$, the fractal has reflection symmetry on both axes.

The Mandelbrot Set

The rendering of the Mandelbrot set created in this activity is rather crude, but so was the first version created by Benoit Mandelbrot himself. Students can trace the figure more easily if they first see a finished picture. However, if you give them time to find it themselves, they may share a certain sense of discovery.

EXPLORE MORE

pre-image	1st image	2nd image
(0, 0)	(0, 0)	(0, 0)
(-1, 0)	(0, 0)	(-1, 0)

The point (0, 0) continues to be mapped onto itself no matter how many iterations. The point (-1, 0) is mapped to (0, 0), then back, then continues to jump between these two points.

RELATED ACTIVITY AND SKETCH

The related activity The Mandelbrot Set enables students to create and explore the Mandelbrot set. The sketch that accompanies that activity (**Mandelbrot.gsp**) shows for a more detailed rendering of the Mandelbrot set and allows students to zoom in on any portion of the rendering to see the fractal boundary in more detail.