The new Western Canada Protocol for Mathematics requires high school students to be familiar with fractals, which are a type of geometric object with self-similarity and recursive properties. Many school teachers and their students already use software tools to demonstrate and explore geometric concepts on the computer. We describe how to build fractals using the familiar Geometer’s SketchPad™ software.

**Fractals and Geometer’s SketchPad**

A fractal is a geometric shape that has a basic property of self-symmetry: roughly speaking, parts of the shape look like small copies of the whole. It is a strange notion when you first hear of it, but when you see a few examples the concept becomes clear. One well-known example is the Sierpinski gasket, a very beautiful simple fractal, shown in Figure 1 below.

![Figure 1: The Sierpinski Gasket](image)

You can quickly see large triangles in the shape, with repetitions of smaller triangles inside. It is not hard to notice the top half of the gasket is an exact copy of the whole thing, at one-half the size. Indeed, the gasket is repeated three times in itself, once at each corner, each exactly one-half the size of the whole.

Another example, shown in Figure 2, is called the Koch Curve. Here the self-similarity may not be immediately obvious, but notice the basic shape of the whole curve as one large bump surrounded by two smaller bumps. This pattern is repeated throughout the curve, at various smaller sizes. It shouldn’t take long before you notice the whole curve is really four copies of itself, at exactly one-third the size.

![Figure 2: The Koch Curve](image)

Elegant as these figures are, they can be a challenge to draw. Their fractal properties make them well suited to construction on a computer, but students can quickly become mired in programming details if they try to build a computer program to create these forms. A solution is to use a computerized drawing package that has all the necessary commands to build a fractal from scratch. Geometer’s SketchPad is just such a package.

Geometer’s SketchPad (or GSP) is a handy tool widely used in high schools and colleges for exercises in geometry and for explorations of geometrical constructions. This software is akin to a “word processor” for geometry, including such basic objects as points, lines, and circles, and provides a variety of point-and-click tools to manipulate those objects in geometrically useful ways. The software “knows” how to do many standard straight-edge and compass operations, transformations, and constructions.

What GSP also provides is a simple scripting tool that allows a student to record a series of geometric constructions, then repeat them over and over again. This repetition, or iteration, of basic commands becomes the tool for building self-similar fractals.

This article provides a brief tutorial on how to create fractals in Geometer’s SketchPad. I’ll assume the reader is familiar with the basic operations of this software—even if you’re not, an hour’s review should be enough to become comfortable with the basics. A number of fractals will be constructed using a form of iteration as a basic construction step. In each case, the “looping” instruction in a GSP script is used repeat some sequence of elementary operations and create the fractal. We will begin with a simple circle construction to demonstrate how scripting and iterations work, and progress to more and more
complex examples. This will provide a basis for further explorations on your own.

A key step in every fractal construction is the doubling (or tripling, or more) occurring in each recursion. This leads to exponential growth of the number of geometric elements on the screen—so don’t iterate too deeply or your computer may have difficulties. It also leads to the interesting properties (visually and otherwise) of fractal constructions. Once the fractal is built, it is an entertaining challenge to try to measure geometric properties of the resulting shape: length of the perimeter, area, or number of lines/circles/points in the fractal. But this is an exercise for another day.

A Simple Fractal With Circles

Since GSP uses circles as one of its basic constructions, it is not surprising that one of the easiest fractals to construct is a nest of circles, as shown in Figure 3.

To build the GSP script that accomplishes this, begin with a pair of points. The sequence of steps will be as follows: join two points with a segment, then draw a circle with this segment for its diameter. Use the midpoint and endpoints of the segment for the iterations.

We create a script first by opening a new “Sketch” in GSP, then opening a new “Script.” Each of these two commands are found under the “File” menu, and each will open its own window.

In the “Script” window, click on the “Record” button to begin transcribing the graphic operations that will be done in the “Sketch” window.

Now the constructions. Switching to the “Sketch” window, create two points on the screen using the “Point” tool. Draw a segment to connect the two points, either using the “Segment” tool, or using the command under the “Construct” menu. Under the “Construct” menu, create a midpoint for the segment. Now draw the first circle using the circle tool, with the center at this new midpoint, and the width set to span the segment. (The circle tool will click automatically to the correct size as you drag the mouse towards the segment’s endpoint.)

Now for the iterations. Shift-click to select an endpoint and the midpoint; on the “Script” window, click on the “Loop” button, to tell the script to iterate on these two points. Then Shift-click to select the other endpoint and the midpoint, and again click “Loop” to set another iteration.

Finally, to clean up the picture, click on the line segment and hide it with the “Hide Line” command on the “Display” menu. Then click on the midpoint and hide it as well.

The script is done. Click on the “Stop” button in the “Script” menu, and the script is ready to run. First, clear the “Sketch” window, put on two new points, select them, then click on the “Play” button to start the script. The computer will ask you how many iterations you want to run; try just 1 iteration the first time, to see that the script works as expected. Try again with 5 or 10 iterations, to see the fractal form.
Given:
1. Point A
2. Point B
Steps:
1. Let \([j]\) = Segment between Point A and Point B (hidden).
2. Let \([C]\) = Midpoint of Segment \([j]\) (hidden).
3. Let \([c1]\) = Circle with center at Midpoint \([C]\) passing through Point A.
4. Recurse on \([C]\) and A.
5. Recurse on \([C]\) and B.

Figure 5: The Circle Script

If this is not working for you, look over the script recorded in Figure 5. Notice that it is only five lines long, and uses only two points as initial data. Your script should look something like the one in the figure. Be sure to have only two points selected when you “Play” the script. Unfortunately, there is no way to edit a script once you have recorded your actions. It is best to start from scratch if you are having problems.

Once the fractal is made, try dragging around the initial two points—the whole fractal will follow them around. This is part of the power and attraction of using GSP in fractal studies.

A Four-Circle Fractal

The last example with circles can be extended by setting four smaller circles inside each large circle. The resulting fractal gives a wonderful geometric figure reminiscent of a Pysynka, or Ukrainian Easter egg, as shown in Figure 6.

Figure 6: Four-Circle Fractal

The basic iteration is shown in Figure 7, where the initial circle is filled with four overlapping, smaller circles at right angles to each other.

Figure 7: Four-Circle Iterations

Again, the circles and resulting fractal are based on an initial selection of two points. The only new feature used here is GSP’s construction tool which makes a perpendicular line to the circle’s diameter. This is then used to construct the third and fourth inside circles.

As before, open a new “Sketch” in GSP, and open a new “Script.” Click on “Record” to begin the creation of the script.

Keeping in mind the steps are being recorded, create two new points and draw a segment to connect them. Under the “Construct” menu, create a midpoint for the segment. Draw the first circle with the circle tool, with center at this new midpoint, and width set to span the segment. (Again, the circle tool will click automatically to the segment endpoint as you drag towards it. You may prefer to use the menu command that draws a circle automatically from two points.)

Select the segment again, and the midpoint, then build a perpendicular line by selecting “Perpendicular Line” under the “Construct” menu. Now select the perpendicular line and the circle, then choose “Point at Intersection” under “Construct” to create the two points of intersection of the circle and line.

Now the iterations. There are four pairs of points on which the script will iterate. Shift-click to select an endpoint of the initial segment and its midpoint; on the “Script” window, click on the “Loop” button, to tell the script to iterate on this pair of points. Then Shift-click to select the other endpoint and the midpoint, and again click “Loop” to set another iteration. Then repeat this for the midpoint and one intersection point of the line and circle; then the fourth iteration using the other intersection point.

Figure 8: Four–Circle Script

Select the segment again, and the midpoint, then build a perpendicular line by selecting “Perpendicular Line” under the “Construct” menu. Now select the perpendicular line and the circle, then choose “Point at Intersection” under “Construct” to create the two points of intersection of the circle and line.

Now the iterations. There are four pairs of points on which the script will iterate. Shift-click to select an endpoint of the initial segment and its midpoint; on the “Script” window, click on the “Loop” button, to tell the script to iterate on this pair of points. Then Shift-click to select the other endpoint and the midpoint, and again click “Loop” to set another iteration. Then repeat this for the midpoint and one intersection point of the line and circle; then the fourth iteration using the other intersection point.
To clean up the picture, click on the line segment and hide it with the “Hide Line” command on the “Display” menu. Then click on the midpoint, the perpendicular line, and its two intersection points, and hide them all as well.

The script is done. Click on the “Stop” button in the “Script” menu, and the script is ready to run. First, clear the “Sketch” window, and put on two points, select them, then click on the “Play” button to start the script. The computer will ask you how many iterations you want to run; try just 1 iteration the first time, to see that the script works as expected. Try again with 5 or 10 iterations, to see the fractal form.

The resulting script is only a little more complex than the first example. Figure 8 gives an example from a test recording.

The Broccoli Fractal

Figure 9 shows a sample of the well-known broccoli fractal, so-called because of its similarity to a head of real broccoli. Notice the branching bushes of polygons—this is a useful fractal for demonstrating to students methods for computing areas, perimeters, and dimensions of iterated geometric figures.

The basic iterated figure is a five-sided polygon, essentially a square with a right-angled roof perched on top. Two smaller copies of the polygon get attached to the short sides of the top, as shown in Figure 10. Creating the script for this figure is somewhat more complicated, because making a square takes several steps in GSP, as does making triangles.

To record the script, make two initial points and join them with a horizontal segment. Rotate the segment and its endpoint by 90 degrees around the other endpoint, to obtain one vertical side of the square. Reverse endpoints to get the other side of the square. Rotate the sides up by 135 degrees to get the right triangle on top, with extensions past the top vertex of the triangle.

Given:
1. Point A
2. Point B

Steps:
1. Let \([j]\) = Segment between Point A and Point B.
2. Let \([B']\) = Image of Point B rotated 90 degrees about center Point A (hidden).
3. Let \([j']\) = Image of Segment \([j]\) rotated 90 degrees about center Point A.
4. Let \([A']\) = Image of Point A rotated -90 degrees about center Point B (hidden).
5. Let \([j'']\) = Image of Segment \([j']\) rotated 135 degrees about center Point \([A']\) (hidden).
6. Let \([j''']\) = Image of Segment \([j'']\) rotated 135 degrees about center Point \([B']\) (hidden).
7. Let \([C]\) = Intersection of Segment \([j''']\) and Segment \([j''']\) (hidden).
8. Let \([x]\) = Segment between Point \([C]\) and Point \([B']\).
9. Let \([l]\) = Segment between Point \([C]\) and Point \([A']\).
10. Recurse on \([B']\) and \([C]\).
11. Recurse on \([C]\) and \([A']\).

The Tree

Trees are one of the most basic fractal shapes: the form of a tree starts with a main trunk, the trunk splits into a number of branches, the branches extend and split into sub-branches, and so on. Surprisingly, this can be a tricky fractal to create in GSP, because of the variety of transformations needed to create sub-branches from the original trunk: translations by vectors; rotations; even dilations can be used.

Figure 12 shows a simple tree: each branch splits into two at the joints, and each sub-branch is the same length.
as the original. For simplicity, we avoid dilations so that each branch is the same length as the trunk.

Figure 12: Simple Tree

![Simple Tree Diagram]

Figure 13: Simple Tree Iteration

Two repeats of the basic iteration are shown in Figure 13. Note in the first iteration, only one segment is drawn, and two new endpoints for the branches are created. The branches themselves don’t get drawn until the next loop of the iteration.

The steps of the construction are as follows. After beginning the recording, create a vertical line segment from two points, then translate the top point vertically using the “Transform” menu, with the endpoints of the segment defining the translation vector. Rotate this new point left and right by, say, 8 and 10 degrees. Iterate on the new top and bottom points created (which will form the endpoints of the new branches) by selecting the top point of the trunk and the new top point of the branch (again, order is important), then click on the “Loop” button. Do the same for the other branch point. Don’t forget to hide the one extra point created in the construction. Click “Stop,” then test out the script. A sample script is shown in Figure 14.

Given:
1. Point A
2. Point B
Steps:
1. Let \[j\] = Segment between Point A and Point B.
2. Let \[B'\] = Image of Point B translated by vector A→B (hidden).
3. Let \[B''\] = Image of Point \[B'\] rotated 10 degrees about center Point B (hidden).
4. Let \[B'''\] = Image of Point \[B''\] rotated -8 degrees about center Point B (hidden).
11. Recurse on B and \[B''\].
12. Recurse on B and \[B'''\].

Figure 14: Simple Tree Script

Exercises

Try a couple of variations on the above fractals. For instance, create a new circle fractal by inscribing three (or four) circles inside an initial circle, with no overlap in the circles. Make a flexible broccoli fractal where the top triangle is adjustable—that is, it is not necessarily a right triangle. Try a tree that has three branches at each joint, or four. Make a tree where each sub-branch is shorter than the originating branch by some fixed ratio. Make the angles and ratios in the tree adjustable.

Examine some fractals you have seen before, and determine how to make them as iterative structures. The Sierpinski gasket is a good place to start, as is the so-called Sierpinski carpet, which uses squares rather than triangles. The Koch Curve and Koch Snowflake are a bit more challenging, but quite do-able in GSP. Check out some fractal examples you’ve seen in books, and try to reproduce them in GSP. Finally, see if you can make your own new and interesting fractals.

References

A good source of information on Geometer’s SketchPad is the publisher’s web site at http://www.keypress.com There one can find demo versions of the software, documentation, many examples of scripts and interesting sketches, and Java implementations of the software. One can also purchase individual or classroom versions of the code at the site. There are many, many books on fractals—Mandelbrot has written very readable ones with plenty of beautiful pictures. Rather than recommending any particular book, let me suggest you see what you can find at your local library.