

Mandelbrot Plot

A web-based Mandelbrot set plotter built in Adobe Flash, available at <http://www.flashandmath.com/advanced/mandelbrot/index.html>

Quick Start Guide

To get started with generating some beautiful fractal images, simply use the mouse to draw a rectangle on the fractal plot, and hit the **zoom selection** button to zoom the plot in to your selection.

If you zoom in very close on the image, you may find that it becomes less detailed, with more black regions. To see more detail in the image, change the **max iter** parameter to a higher number (start with 1000). The higher this number, the more detail you will see (but the fractal will take longer to generate).

The easiest way to change the colors is to use one of the preset gradients, obtainable by clicking on the **choose a preset gradient** button at the bottom of the gradient editor. After selecting a gradient, click on the **update** button on the left of the fractal plot to change the coloring.

Smoother, better quality images can be obtained by checking the **4x oversampling** box. Be aware that this will increase the rendering time by a factor of four.

Full Instructions

To understand all of the controls and parameters available in this plotter, it is helpful to have a basic understanding of how the Mandelbrot set is defined and plotted. You may skip this background information and scroll down to simply read instructions on the individual controls.

Background information

The Mandelbrot set is the result of *iterating* a function which uses *complex numbers*. A complex number is a number of the form $a + bi$, where a and b are real numbers, and i is the square root of -1 . These numbers are situated in a two-dimensional plane: a single point in the plane with coordinates (a, b) represents a single complex number $a + bi$.

Here is how the plot of the Mandelbrot set is created: a point in the plot represents a complex number c , and for each of these points we will do a long calculation using the value of c to determine how to color the point. For each number c we generate a long list of complex numbers $z_0, z_1, z_2, z_3, \dots$ as follows. We begin with the number $z_0 = 0$, and we set the value z_1 equal to the result from plugging z_0 into the function $z^2 + c$. We get the next number z_2 by plugging z_1 into the same function, and so on. That is:

$$z_1 = z_0^2 + c,$$

$$z_2 = z_1^2 + c,$$

$$z_3 = z_2^2 + c,$$

⋮

etc.

We say that we are *iterating* the function $z^2 + c$, and we call these numbers z_i the *iterates*. As it turns out, one of two things can happen with this list of numbers, depending on what value of c is used in the function: either the iterates stay small in size (we say they are *bounded*), or they get larger and larger in magnitude (we say their *magnitude approaches infinity*, or that they are *unbounded*). If the iterates stay bounded, we say that the point c is in the Mandelbrot set and we color this point black. If the numbers are unbounded, this point c is not in the Mandelbrot set and we will color this point according to a colorful gradient that we have selected.

It turns out that once these numbers z_i get at least 2 units away from the center point 0, we can be certain that they will then run off to infinity. (We call this number 2 the *bailout radius* for the iteration.) So we can stop calculating these iterates when this happens. The color we choose depends on how many iterates were required before the numbers reached the bailout radius. This coloring method produces a discrete (stepped) coloring, but we can employ a more sophisticated method to produce a continuous (smooth) coloring.

Deciding that the iterates are in fact bounded (and thus the point c should be colored black) is done this way: we simply calculate the iterates some maximum number of times, and if the iterates still have not gone beyond two units away from 0, then we assume that they stay bounded forever. Of course, this assumption is sometimes incorrect. The larger we make this maximum iteration amount, the more accurate the Mandelbrot plot becomes.

The Controls

Drawing rectangles

To select a rectangular portion of the graph for zooming, simply draw with the mouse. After a rectangle is drawn, you can resize it by dragging the edges, or move the whole rectangle by clicking and dragging in the interior of the rectangle. If your rectangle is too small to be able to click inside, click on the hand icon at the top left of the plot to change to a moving only mode. Click on the crosshairs to return to general rectangle drawing.

Left panel buttons

update : Any time you make changes to the parameters (color gradient, maximum iteration amount, etc.) click on the update button to redraw the fractal using these new settings.

zoom selection : Click here to zoom in on the rectangular selection you have drawn. Note that the plot will always use the smallest square which contains your rectangle.

zoom out : Click here to zoom out by a factor of three.

back : Click here to return to the last plot. The plot will be regenerated instead of recalled from memory.

start over : Click here to return to a plot showing the entire Mandelbrot set. The maximum iteration amount will be reset to 100.

Bottom panel

color period : Changing this parameter will affect the rate at which the colors change. Set this to a higher number to slow down the color change, and a smaller number to make the colors change more rapidly. This value should be set to something larger than 0.

color phase : Changing the color phase will cause a shift in the coloring, without changing the rate at which the colors change. This value should be set to something between 0 and 1.

max iter : This stands for maximum iterations. This is the greatest number of iterates that will be computed before the iterates are assumed to be bounded. You will need to set this number to a higher value when zooming in on the small-scale details of the Mandelbrot set, where it takes longer for iterates to escape the bailout radius 2. Be aware, however, that larger values of the maximum iteration amount will cause the fractal to take longer to render.

coloring method : **log, log log, linear, discrete**. Experiment with these different coloring methods to see the effect on the plot, while also adjusting the color period to change the rate at which the colors change. The simplest method of coloring is **discrete**, which will use a short list of colors sampled from your gradient to represent the number of calculations that were required before the iterates escaped the bailout radius. When using discrete coloring, set the **steps** parameter (which will appear if you select the discrete coloring option) according to how many colors you want to use before they are cycled through again. Also be aware that the color phase will shift the selected colors. The other coloring methods available in this list employ a more sophisticated technique to produce continuous (smooth) coloring. The **linear** method roughly matches the discrete method, but with smooth transitions of colors rather than stepped values. Choosing the **log** coloring method will slow down the transition of colors as the boundary is approached, which may look nicer because with linear coloring the colors tend to change very rapidly near the boundary. The **log log** method will have a more pronounced slowing effect.

4x oversampling : A much higher quality plot can be achieved by *oversampling*: that is, a plot twice as long and wide (hence requiring four times the data) is calculated internally, then smoothly resized down by a factor of two for the display. This method produces much nicer looking fractals, but of course takes four times the amount of time to compute. So you may wish to turn this off until you've found a fractal window that you like, then turn it on and click on update to regenerate a smoother picture.

The gradient editor

The gradient editor provides two methods for creating a gradient for the coloring of the fractal: you can either draw red, green, and blue curves to show how these color components change over the gradient, or type in color values in the text-based editor.

Red, green and blue color curves. Since every color that a monitor can display is the result of combining red, green, and blue light, a gradient represents a change in these three light components. You can draw different red, green, and blue curves by hand to show how these color components change over the gradient. To draw straight lines, hold down the Control key while you draw.

smooth, **auto** : The smooth button will smooth out your rough hand-drawn curve. Press and hold this button to apply continual smoothing. The longer you hold the button, the smoother

the graph will get. If the auto button is on (highlighted), then the curve will automatically be smoothed every time you draw a new portion of it. Click on the auto button to turn on or off auto-smoothing.

↑, **↓**: **up, down arrows**. You can scale the graphs up and down with these arrows. This is helpful if you like the overall quality of the gradient but want to decrease, say, the red amount.

c, **p**: **copy and paste buttons** You can copy one gradient curve to another with these buttons.

↔/**⇒** **cycling or one-direction coloring**: When coloring the exterior of the Mandelbrot set, colors must be continually reused. Clicking on the little arrow button to the right of the gradient, you can choose whether the colors move back and forth through the gradient (**↔**) or move in only one direction (**⇒**).

choose a preset gradient: Click here to open up a window which shows a list of nice preset gradients to use. Click on any gradient to select it.

input specific colors by value: You may wish to input very specific colors for use in your gradient. Click on the input specific colors by value button to open up a window which will allow you to input up to eight colors. The colors must be entered in *rrggbb* hexadecimal form representing the red, green, and blue amounts for the color. After typing in your desired colors, map them to the gradient by clicking on either **map colors (linear)** or **map colors (cosine)**. Linear mapping will transition between the colors with straight lines, cosine mapping will use cosine curves instead, which will make the gradient seem to "pause" somewhat at each of your chosen colors. You can also read colors from your gradient into these text fields by clicking on **read colors from gradient**. The number of colors sampled from your gradient depends on how the Number of colors amount has been set.

Export panel

PNG and **JPG**, **export side length, show info**: Clicking on the PNG or JPG export buttons will allow you to export your fractal in one of these two image formats. You can add an information bar at the bottom of your image showing the plot information (as shown above the plot in the application) by checking **show info**. You can make the fractal have any size (up to the maximum amount) by changing the **export side length**. Be aware, however, that if you set a large size and use 4x sampling that the fractal may take a minute or two to render. The rendering time is highly dependent on the maximum iteration setting along with how much of the interior (black portion) of the Mandelbrot set is displayed in your plot. But if you're patient, some very nice large fractal images can be exported with this application.