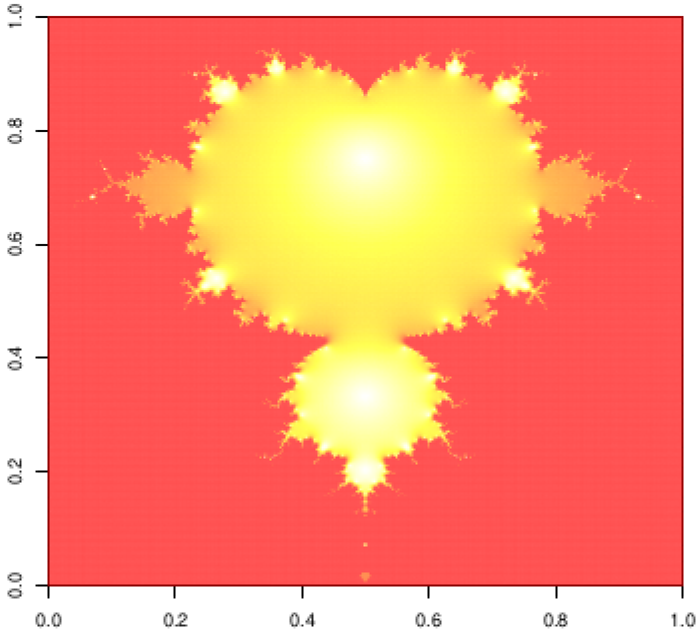
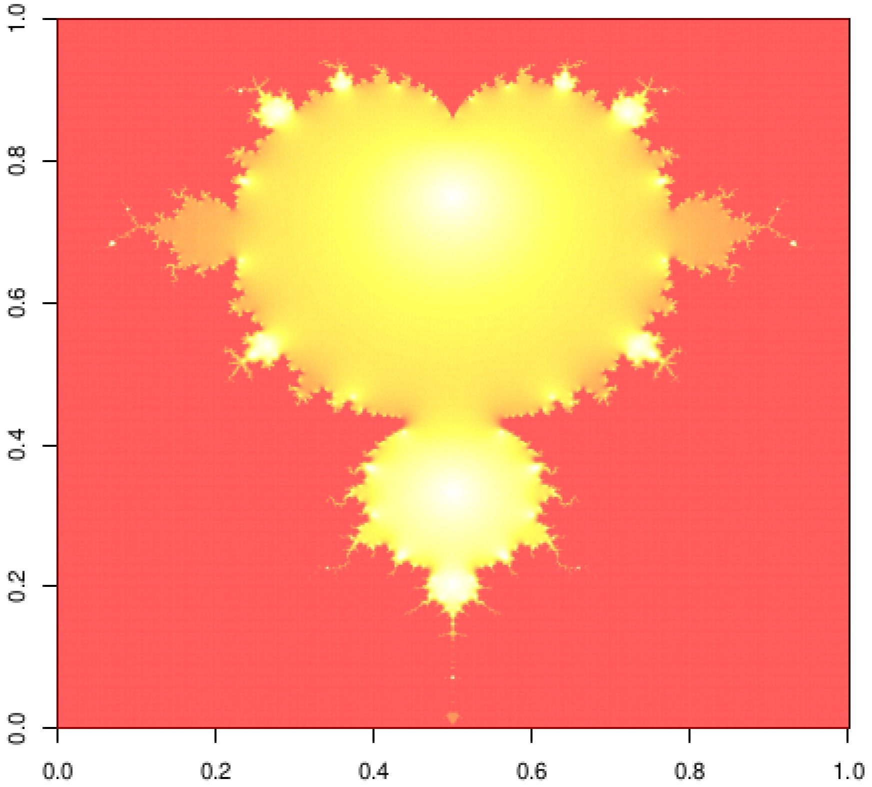
ORIGINAL:



The original image looks to have a good center of the largest node of the system. Then, we can still see the vein-like spread of the fractal design across the ‘default’ parameters.

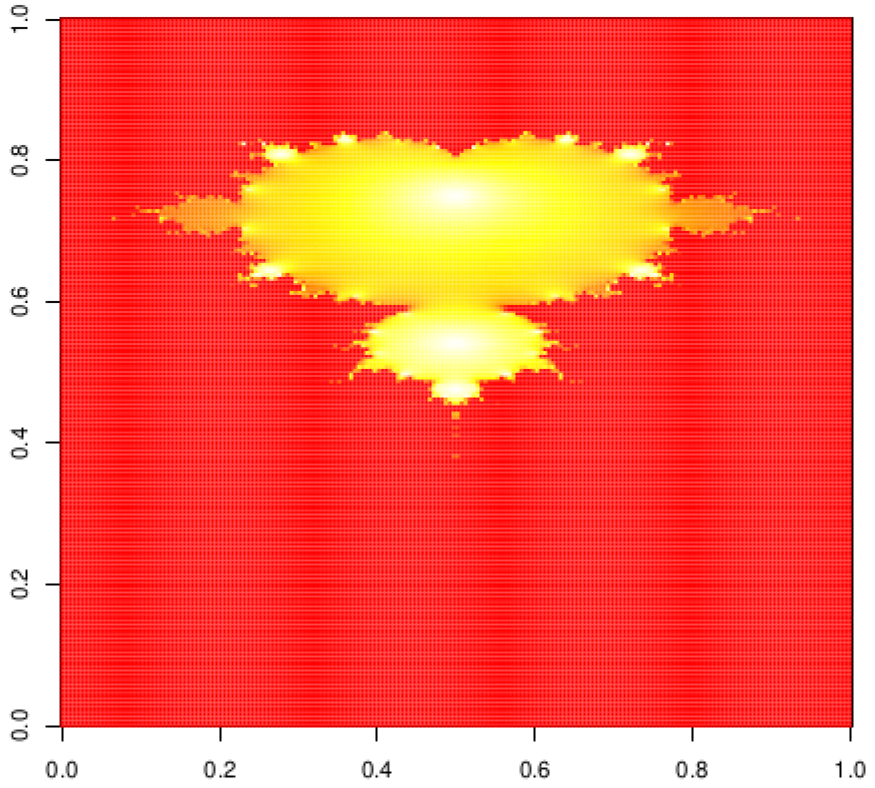
Dim = 1000

This seems to have added a lot more depth to the original fractal design. Giving more degrees to filter through before the image converges for the given resolution. I see more vein like structures emanating off the design.



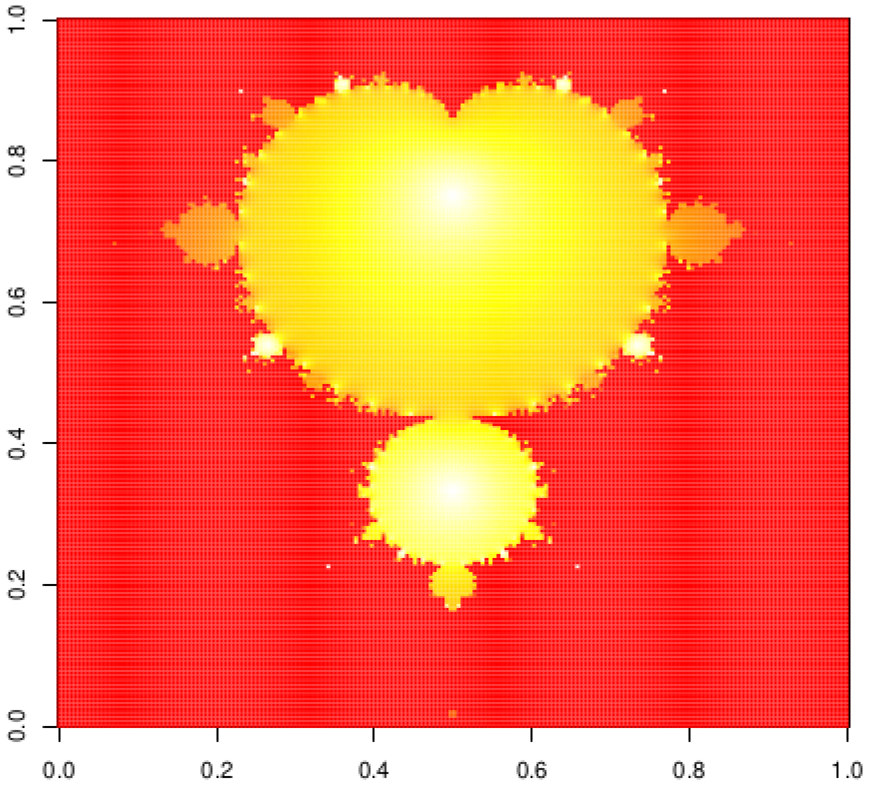
Change Xlo Xhi

Squished or fit the original fractal into this x and y bound fit. Did not alter the actual internals of the fractal generation.

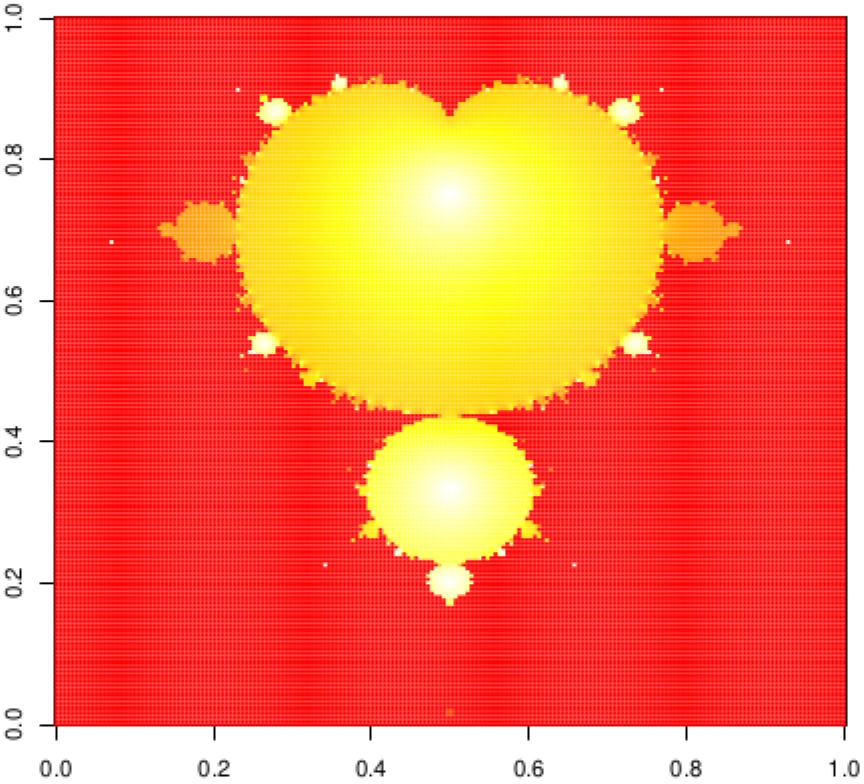


Iteration Changes:
This increased iterations causes the fractal design to steadily approach the basic cardioid pattern as the number of counts increases. A good word to describe this is the aliasing decreases as we lend more iterations to the model. As seen in 1000 iterations for this resolution, the cardioid and resulting circles are pretty smooth. While the artifacts in the other low iterations are jagged and unsmoothed.

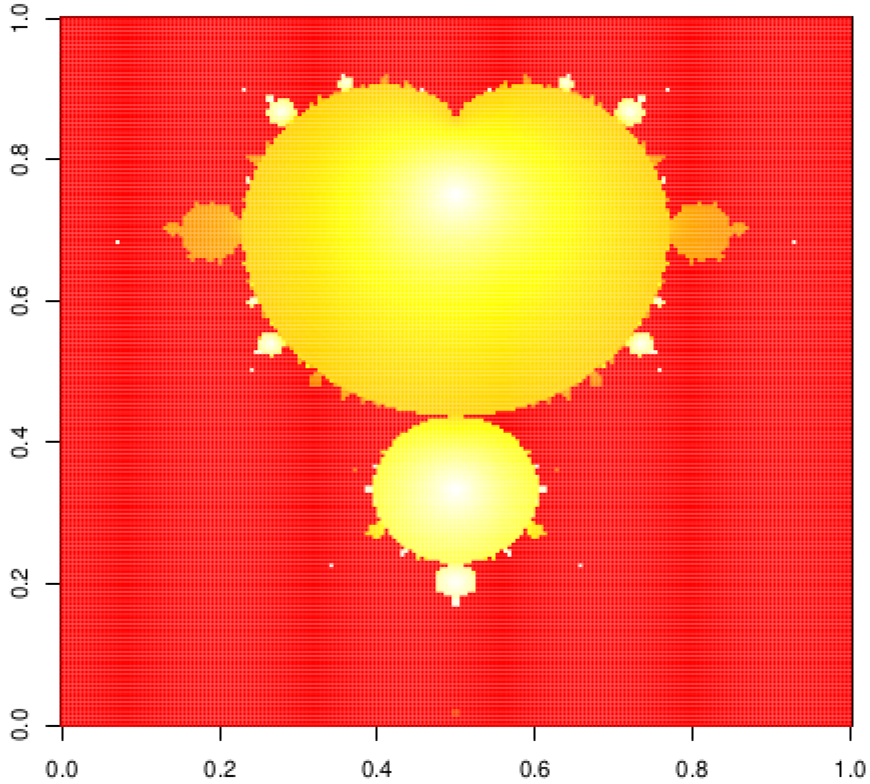
Iters 50



Iters 100


0.0
0.2
0.4
0.6
0.8
1.0
0.0
0.2
0.4
0.6
0.8
1.0

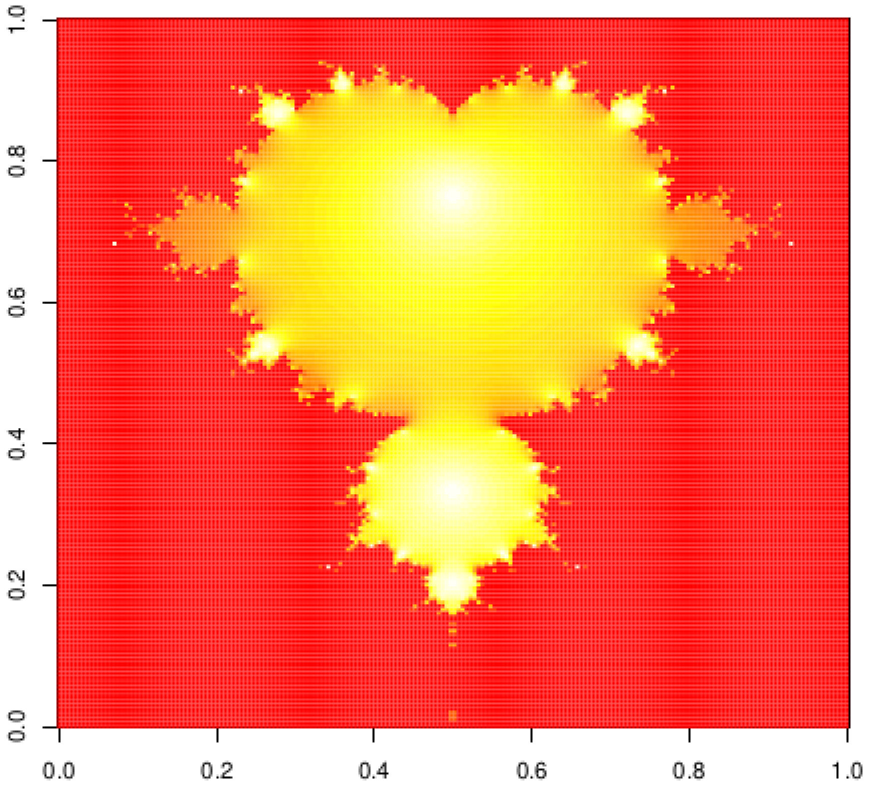
Iters 1000



Topology and Color Changes:
It seems that most of these code changes are just to show the different ways to visual the data through different color lenses. This is very similar to matlab and their plot renderers. Usually for images processing I default to heat but I'm guessing that the color settings are mostly down to context and personal interest.

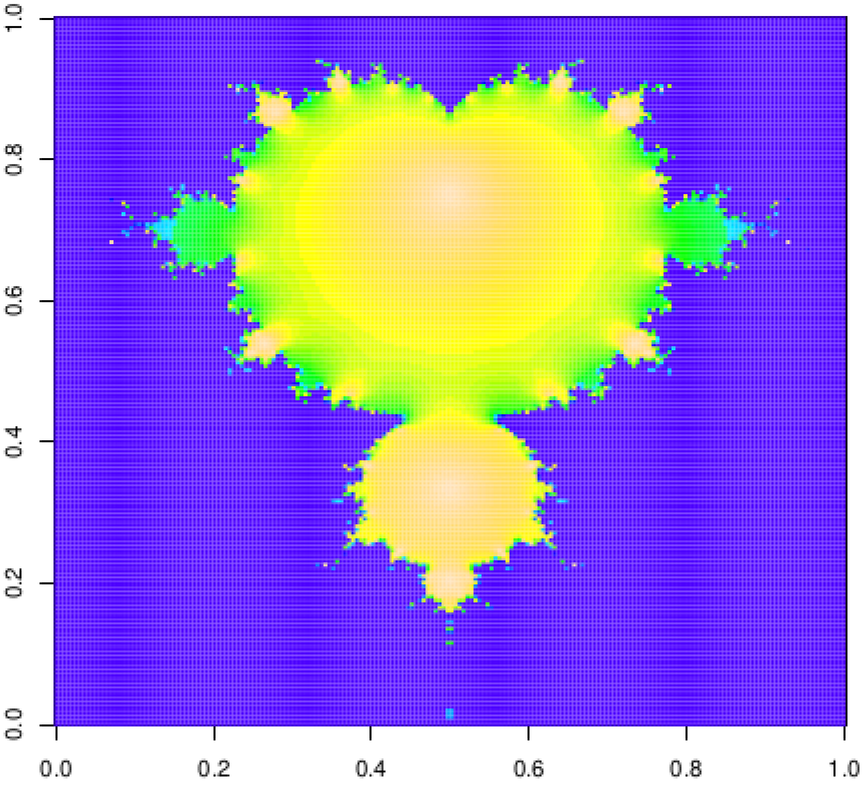
With heat 50 and colors 50 we can see how they each show the intensity information at the pixel value with a set color. Normally it would be best to plot these with an intensity guide or bar title.

Heat 50:

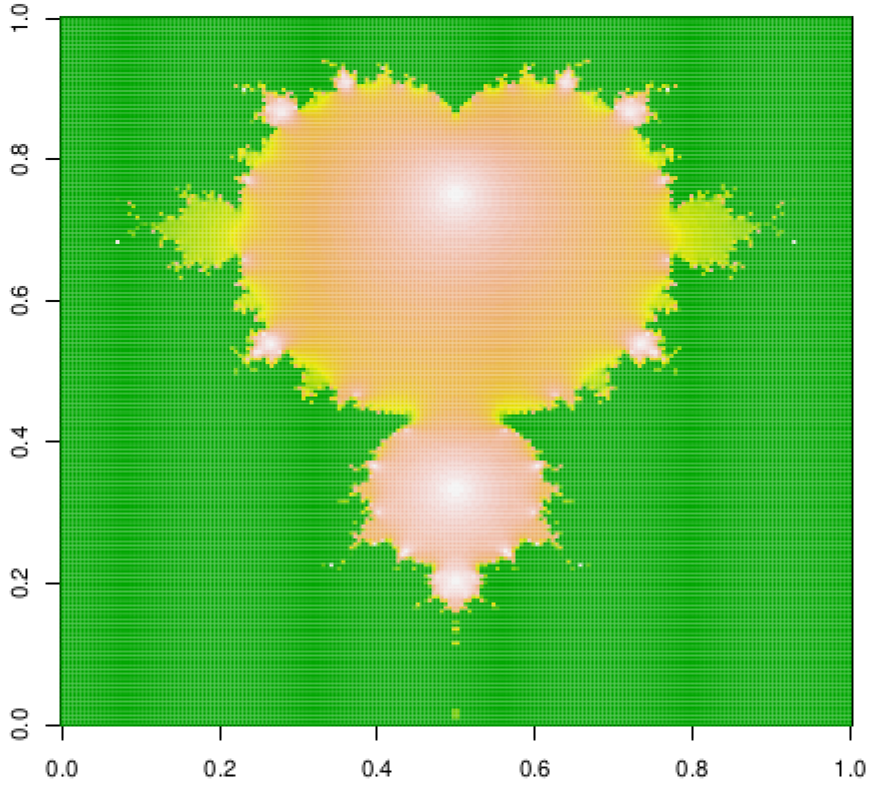


It seems like these other settings just a lend a hand to those presenting these plots and not needing to change the functionality of the design.

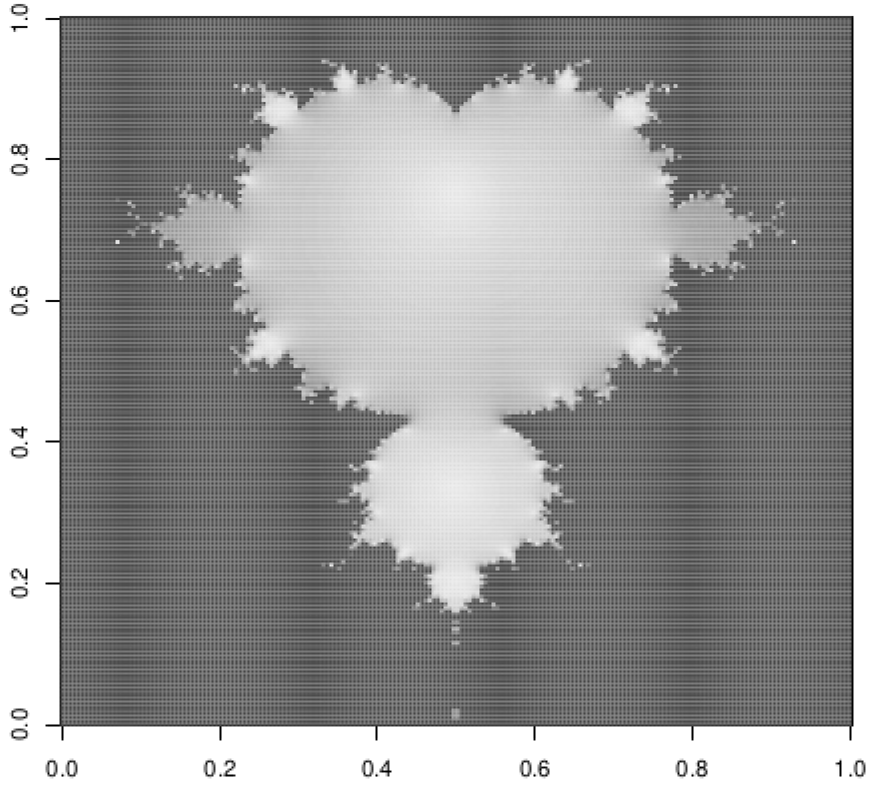
topo.colors(50):


1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

Terrain.colors:

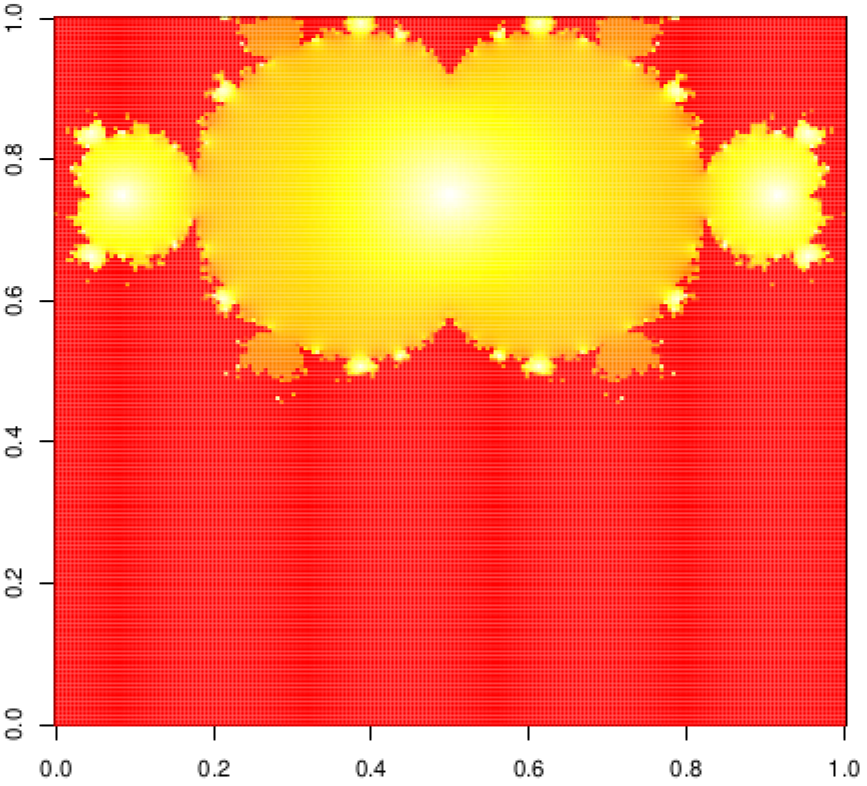


Gray:



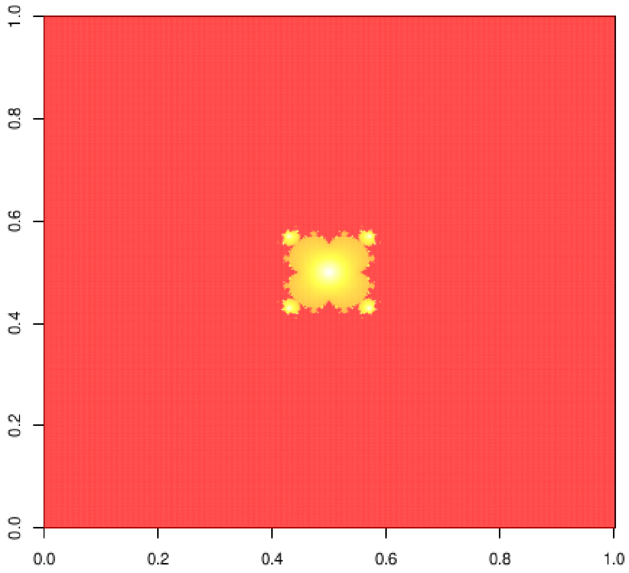
^3 Change

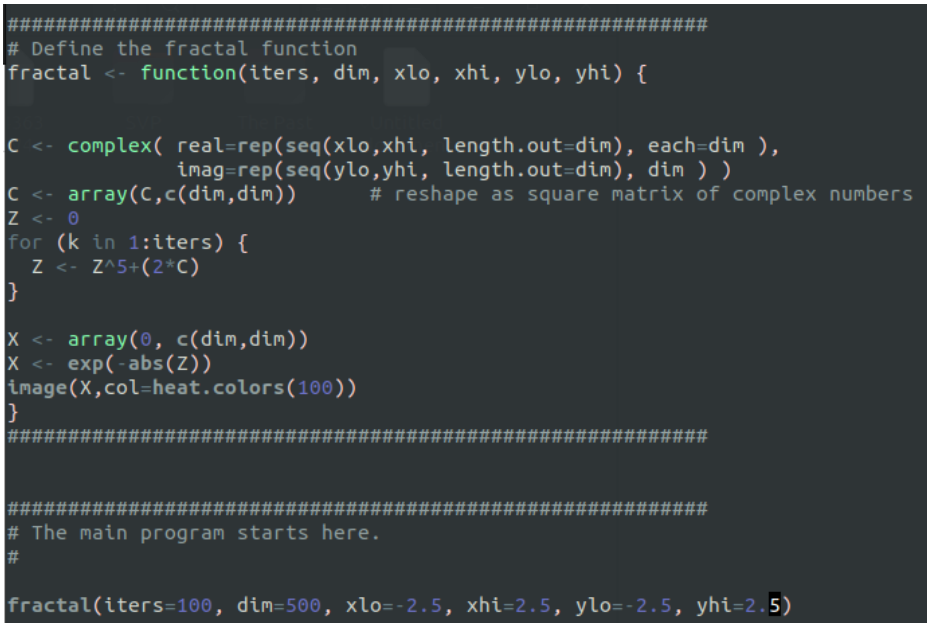
Adding a cubic unit to the iterative equation seems to have drastically changed the way the fractal is built and how it follows its path of emanation. It also looks to have turned the initial cardioid node into a figure 8 of sorts. Also, not sure if the gradients inside the largest nodes are a moire effect of the renderer or if it leads us to a property of the fractal design.



0.0
0.2
0.4
0.6
0.8
1.0
0.0
0.2
0.4
0.6
0.8
1.0

MAKE YOUR OWN:
I decided to just change each parameter including making the squared into a pentic. The dimensions of x and y lo hi were also greater causes less resolution of the resulting system. Also the iters was fairly low so there should be some jagged results not caused by only the low area of display.





```
###########################################################
# Define the fractal function
fractal <- function(iters, dim, xlo, xhi, ylo, yhi) {

C <- complex( real=rep(seq(xlo,xhi, length.out=dim), each=dim ),
              imag=rep(seq(ylo,yhi, length.out=dim), dim ) )
C <- array(C,c(dim,dim))      # reshape as square matrix of complex numbers
Z <- 0
for (k in 1:iters) {
  Z <- Z^5+(2*C)
}

X <- array(0, c(dim,dim))
X <- exp(-abs(Z))
image(X,col=heat.colors(100))
}
###########################################################


###########################################################
# The main program starts here.
#

fractal(iters=100, dim=500, xlo=-2.5, xhi=2.5, ylo=-2.5, yhi=2.5)
```

I’m glad to say that using the CLI only with R worked pretty well. Also I think adding further plugins (for the final proj or something similar) shouldn’t be an issue. I’ll just have to make sure to set it up before the lab or something.