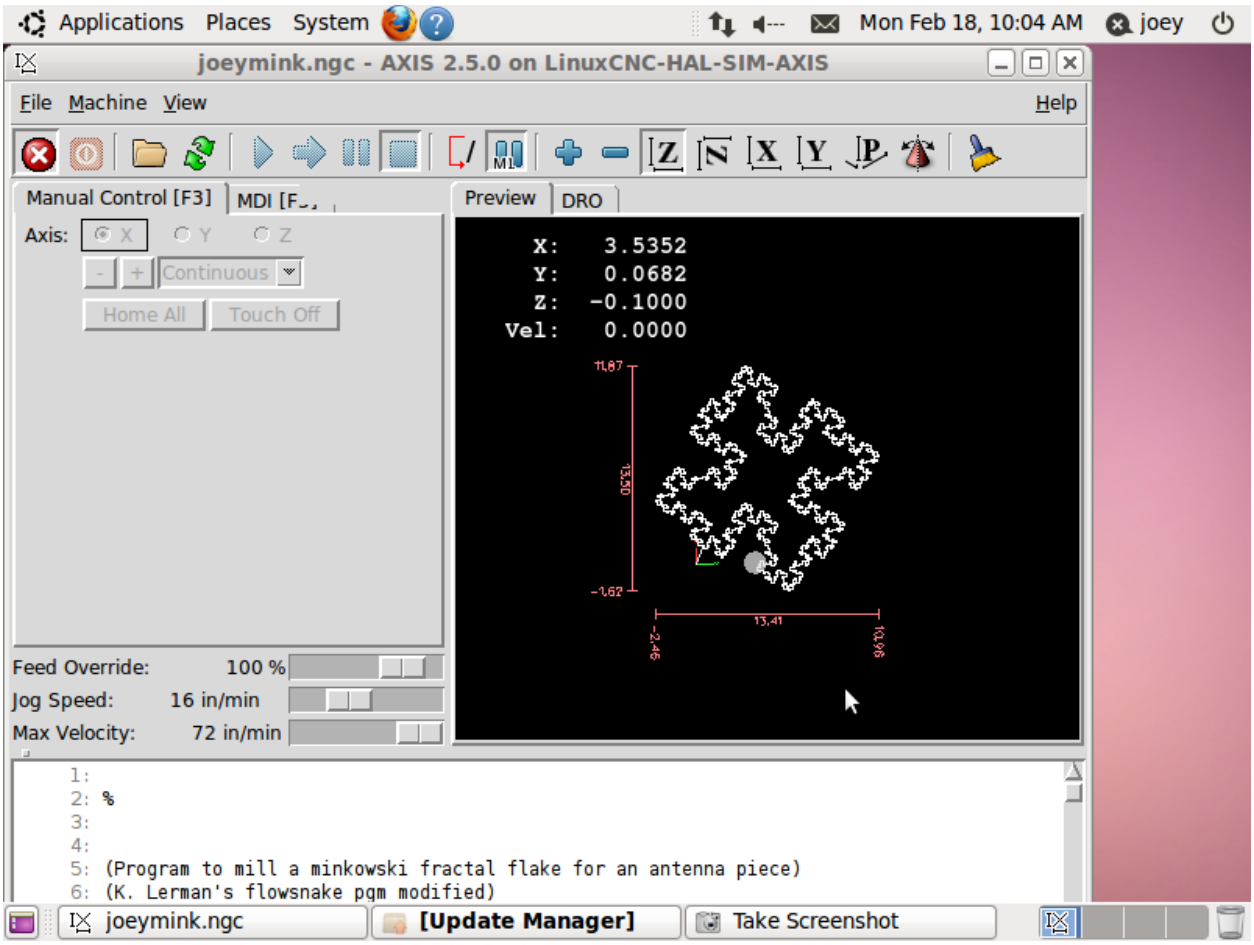
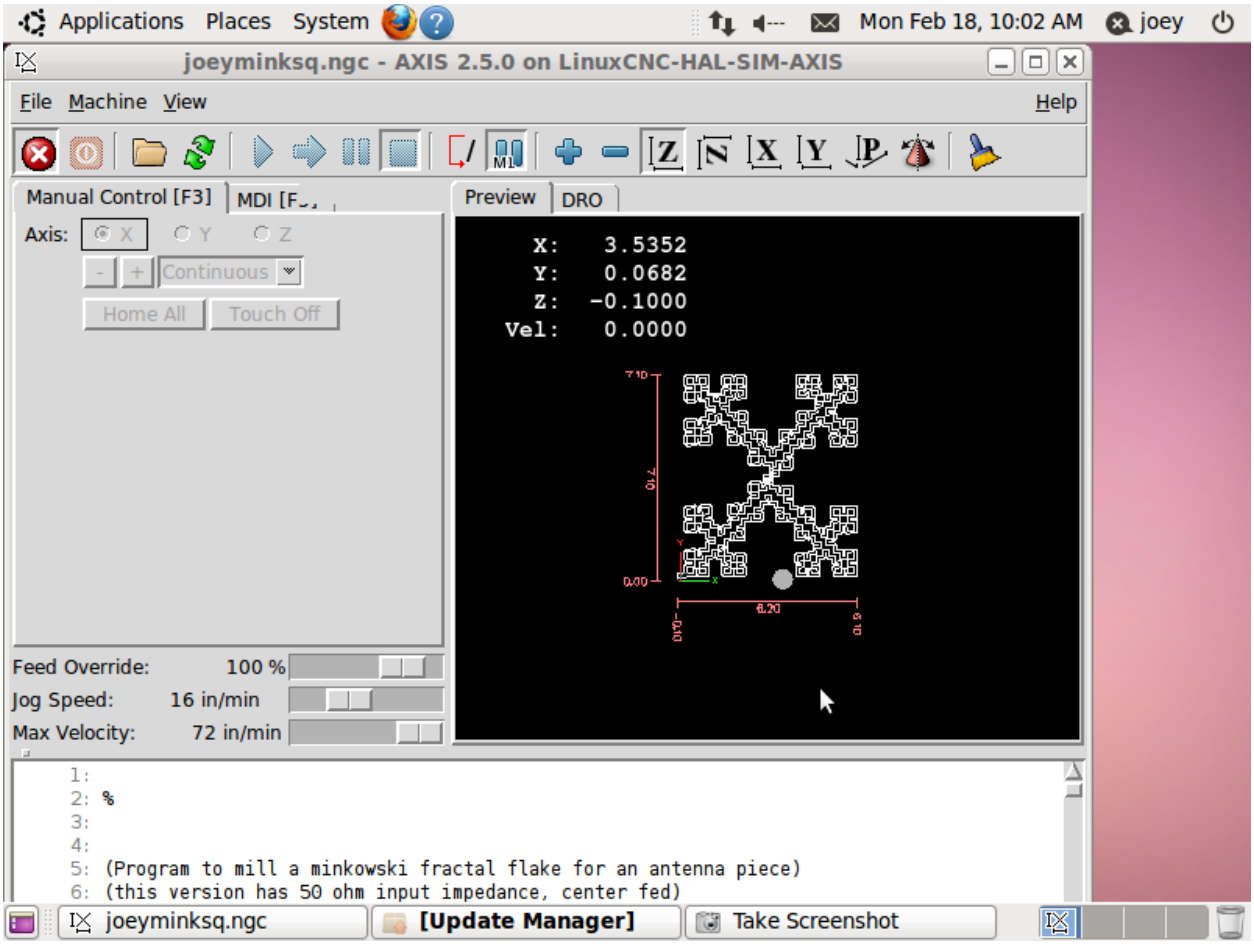


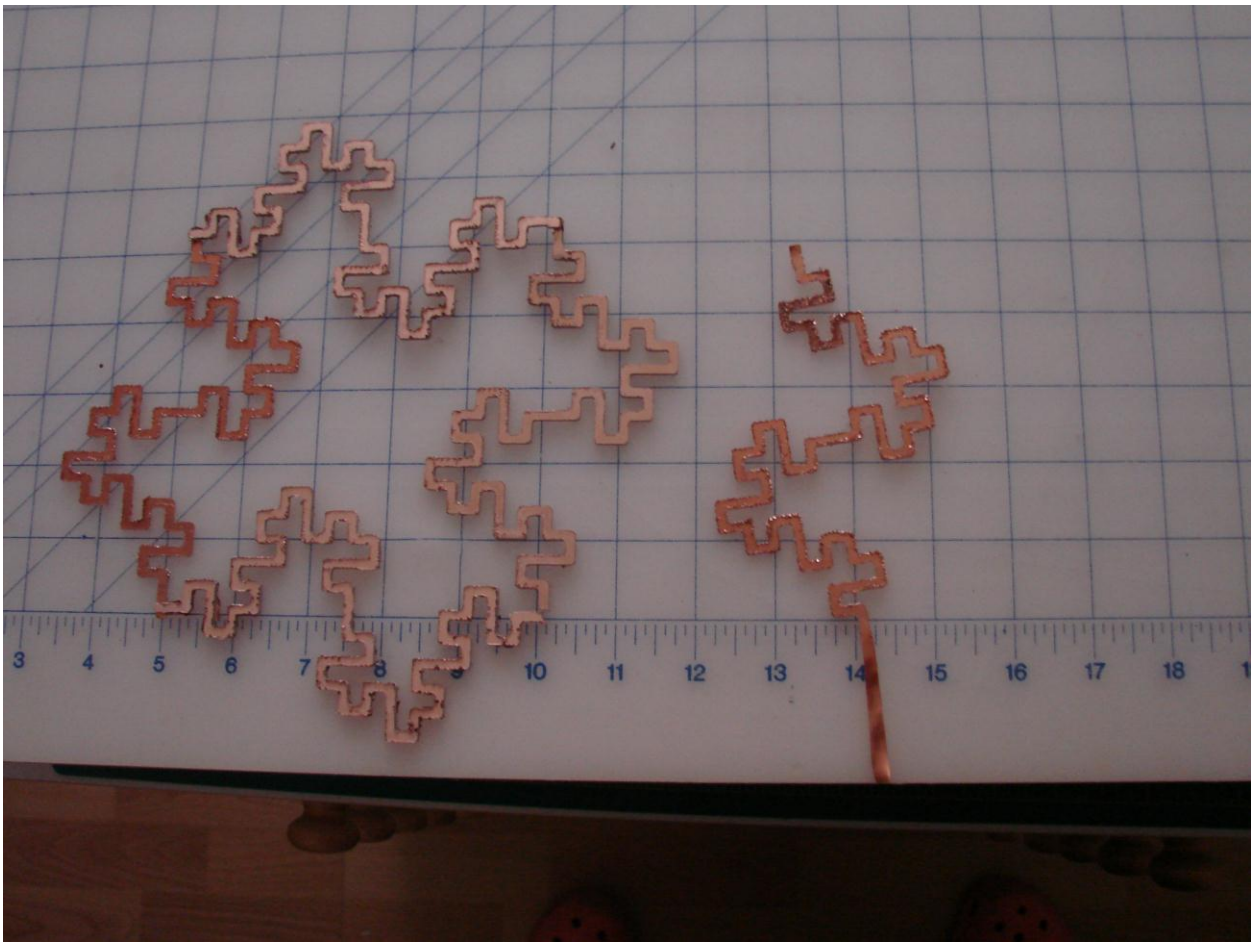
This is a stab at making a recursive figure based on “Minkowski islands” It could be used for making a fractal antenna. The antenna could consist of wire or other conductive material assuming that shape. It is claimed to be the most efficient means of producing lengthy perimeter of a closed figure inside a fixed space. The Wikipedia entry for fractal antenna has more info. I’ve found that just about all information about fractal antennas is copyrighted so they make pretty pictures but you’ll need lawyers before you can sell one. This figure is derived from a square as follows: divide one side of the square into four pieces. Slide the second piece inward by its length and the third piece outward by its length. Connect the outer endpoints of the moved pieces to the inner endpoint of the fixed pieces. Connect the inner endpoints of the moved pieces to each other. Do this recursively to each side , then repeat the process for each line segment in turn. The perimeter of the (four sided) figure “grows” by a factor of eight each pass and

supposedly the resonant frequency is related to the perimeter. For 300Mhz, wavelength is 39”, the above process yields 32” perimeter after one pass starting from a 4” square. I’m in process of trying to mill this shape out of double sided copper clad printed circuit material using the makeitlabs sherline and haven’t finished yet.



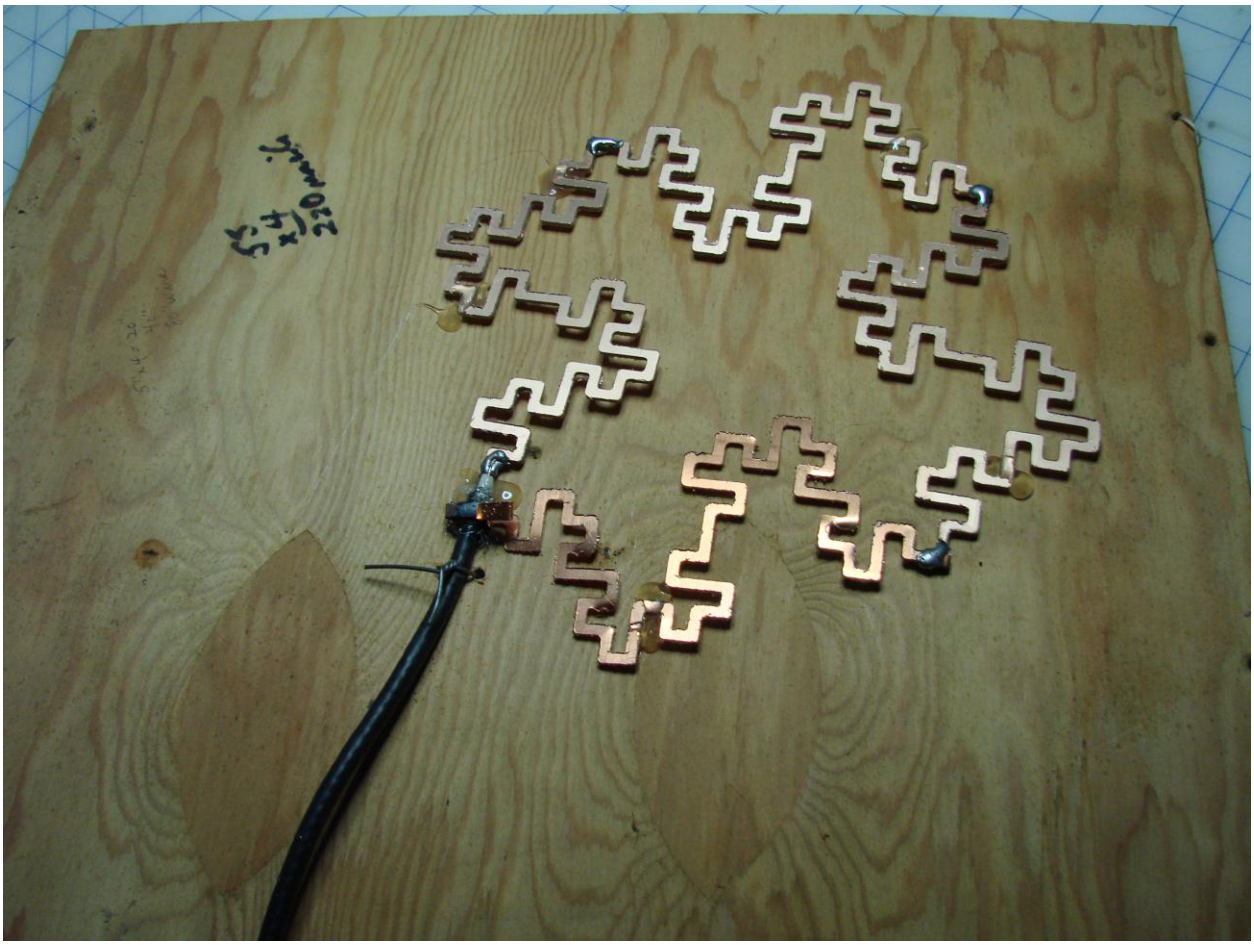
This is another version of Minkowski shape. (this screen dump shows the part with cutter compensation turned on to make a finished part. This figure overwrites itself and would need to be “repaired” before use). Start from a square, divide one side by three, slide the center third inward by its length and connect its endpoints to the inner endpoints of the un-slid lines. Perimeter multiplies by 20/3 each pass assuming a four sided starting figure. For comparisons, a Koch snowflake antenna grows as a multiple of 4/3 on each leg

and they're typically derived from a three sided figure to start. The Serpinski antenna(not shown) is supposed to resonate when triangle height is wavelength / 2. (That gives 19 inches in my case making the Serpinski antenna to big to mill on the Sherline, besides being to difficult to break into subordinate parts.



Here's a picture of the pieces laid out for trial fit, I cut one spare arm for good luck. The next picture shows the assembly after I blob soldered it together. A matching transformer wasn't needed (although I didn't measure for this) and I proceeded to test the thing out. The results were less than spectacular, this antenna gets only 2/3 as many channels as my snowflake antenna does. The perimeter of the figure is 80", I'm planning on making a jig for bending 20 gauge copper wire around. (use one nail for each angle in a sheet of plywood, 220 nails needed) and try again in case the width of the arms is wrong). More likely the

length is wrong, I really need 30 inches, not 80 if I intend to match the meager documentation I've found so far. This was more an exercise in writing gcode and the program is included at the end of this doc.



Program:

```
%


(Program to mill a minkowski fractal flake for an antenna piece)
(K. Lerman's flowsnake pgm modified)
(a trigonometric change is incorporated here)

o1000 sub
#<level> = #1
#<startx> = #2
#<starty> = #3
#<endx> = #4
#<endy> = #5
```

```
(the length of a subsegment, 1/4th of the distance)
#<delta> = [[SQRT[[#<endx>-#<startx>]*[#<endx>-#<startx>] + [[#<endy>-
#<starty>]*[#<endy>-#<starty>]] ]/4]]
(the figure teeters around the first endpoint given)
#<tilt> = [atan [#<endy> - #<starty>]/[#<endx> - #<startx>]]
  o1001 if [#<level> EQ 0]
    o1004 if [[#<endx> LT 14.35] AND [#<endy> LT 14.65]]
     g1 f100 x#<endx> y#<endy>
    o1004 endif
  o1001 else

#<p1X> = [#<startx> + #<delta>*[cos[#<tilt>]]]
#<p1Y> = [#<starty> + #<delta>*[sin[#<tilt>]]]

#<p2X> = [#<startx> +[SQRT[2*#<delta>*#<delta>]]*[cos[#<tilt>+45.0]]]
#<p2Y> = [#<starty> +[SQRT[2*#<delta>*#<delta>]]*[sin[#<tilt>+45.0]]]

#<p3X> = [#<startx> +[SQRT[5*#<delta>*#<delta>]]*[cos[#<tilt>+26.565]]]
#<p3Y> = [#<starty> +[SQRT[5*#<delta>*#<delta>]]*[sin[#<tilt>+26.565]]]

#<p4X> = [#<startx>+2*#<delta>*[cos[#<tilt>]]]
#<p4Y> = [#<starty>+2*#<delta>*[sin[#<tilt>]]]

#<p5X> = [#<startx> +[SQRT[5*#<delta>*#<delta>]]*[cos[#<tilt>-26.565]]]
#<p5Y> = [#<starty> +[SQRT[5*#<delta>*#<delta>]]*[sin[#<tilt>-26.565]]]

#<p6X> = [#<startx> +[SQRT[10*#<delta>*#<delta>]]*[cos[#<tilt>-18.434]]]
#<p6Y> = [#<starty> +[SQRT[10*#<delta>*#<delta>]]*[sin[#<tilt>-18.434]]]

#<p7X> = [#<startx>+3*#<delta>*[cos[#<tilt>]]]
#<p7Y> = [#<starty>+3*#<delta>*[sin[#<tilt>]]]

    o1000 call [#<level>-1] [#<startx>] [#<starty>] [#<p1X>] [#<p1Y>]
    o1000 call [#<level>-1] [#<p1X>] [#<p1Y>] [#<p2X>] [#<p2Y>]
    o1000 call [#<level>-1] [#<p2X>] [#<p2Y>] [#<p3X>] [#<p3Y>]
    o1000 call [#<level>-1] [#<p3X>] [#<p3Y>] [#<p4X>] [#<p4Y>]
    o1000 call [#<level>-1] [#<p4X>] [#<p4Y>] [#<p5X>] [#<p5Y>]
    o1000 call [#<level>-1] [#<p5X>] [#<p5Y>] [#<p6X>] [#<p6Y>]
    o1000 call [#<level>-1] [#<p6X>] [#<p6Y>] [#<p7X>] [#<p7Y>]
    o1000 call [#<level>-1] [#<p7X>] [#<p7Y>] [#<endx>] [#<endy>]
  o1001 endif
o1000 endsub

(S1M3)
(protocol leadin)
g17 g20 g40 g49 g54 g80 g90.1 g94
g10 l2 p1 x0 y0 z0 (make axis set 1 be absolute)
t1 m6 (tellit we're using a tool, 3/16 endmill used here hardcoded)
f 100
G0 x0 y0 z0
#<level> = 2.0
#<depth> = 0.0
o2004 while [#<depth> GT -0.020]
#<depth> = [#<depth> -0.021]
```

```
G40
g0 z 0.25
x0 y0
G41.1 D0.20
g1 x0.25 y0.25
g1 f10 z[#<depth>]
o1000 call [#<level>] [0.25] [0.25] [5.25] [0.25]
(o1000 call [#<level>] [5.00] [0.25] [5.00] [6.00])
(o1000 call [#<level>] [5.00] [6.00] [0.25] [6.00])
(o1000 call [#<level>] [0.25] [6.00] [0.25] [0.25])
G1 z 0.25
G40
g0 x0 y0
G42.1 D0.20
g1 x0.25 y0.25
g1 f10 z[#<depth>]
o1000 call [#<level>] [0.25] [0.25] [5.25] [0.25]
(o1000 call [#<level>] [6.00] [0.25] [6.00] [7.00])
(o1000 call [#<level>] [6.00] [7.00] [0.25] [7.00])
(o1000 call [#<level>] [0.25] [7.00] [0.25] [0.25])
G1 z 0.25
G40
g0 x0 y0
o2004 endwhile
g0 x0 y0
m2
%
```