

Manipulation

Rashmi Sunder-Raj
<http://twitter.com/@HypercubicPeg>
<https://mathstodon.xyz/@HypercubicPeg>

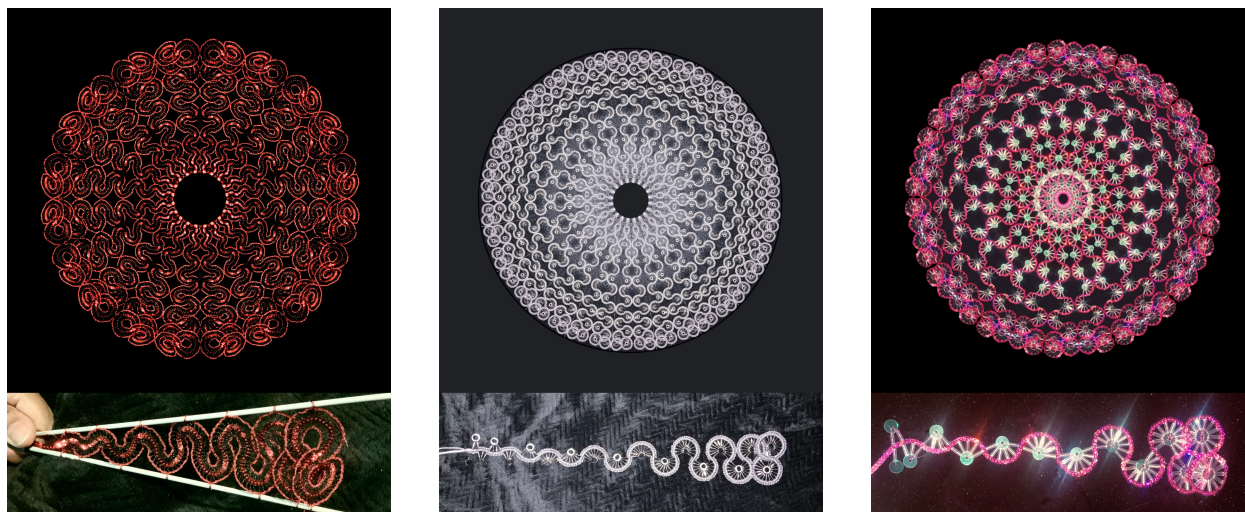


Figure 1: Rotationally-symmetric patterns made by digital manipulation of physical “squiggles” (the physical objects are shown below them).

“Manipulation” is a collection of rotationally-symmetric patterns created by digitally copying, reflecting and modifying physical “squiggles”, which I made using crocheting, tatting, or beading.

These *squiggles* can be loosely defined as figures created by moving along a sequence of circles of constant size, switching direction at each point of tangency. I stumbled upon the idea while trying to fit together rings of polygons in vector graphics programs.

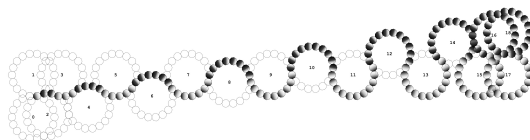


Figure 2: A “linear squiggle” formed from partial rings of polygons.

I wrote about them in the paper ([Physical Representations of Polygon, Wedge, and Arc Squiggles \[1\]](#)) which was presented at the 2023 Bridges Math Art conference. In it I described them as follows:

“Each squiggle is associated with a *sequence* $a_0, a_1, a_2, a_3, \dots, a_n$, which indicates how far to travel around each “circle” before switching to the next one. This distance is defined in multiples of some *unit*, which I usually take to be a polygon, a wedge, or an arc length. The *squiggle angle* is the angle subtended by this unit at the center of the “circle”. If the sequence is of the form $a, a + k, a + 2k, \dots, a + nk$, then I will refer to both the sequence and the squiggle as *linear*.”

The linear squiggles have the interesting properties that they are pretty much bounded by straight lines, and the angles of the triangle forming their convex hull are predictable. These properties can be useful when creating rotationally symmetric patterns. Please see the supplement to the Bridges paper (https://archive.bridgesmathart.org/2023/bridges2023_Supplement_108.pdf) for further details and proofs.

The computer work that I have done related to these explorations has primarily involved the hand-manipulation of figures in a vector graphics program. After hearing about the Art Exhibition at the Maple Conference, I thought that I would have a go at programming Maple to automatically generate some squiggles for me, and perhaps speed up some of my future experimenting in the process.

The version of Maple that I have is rather old (Maple V Version 5.0.2 for Macintosh from 1992), as it is left over from the days when I was making figures for my masters thesis. It is missing many of the features included in newer versions (and even some useful ones that exist in later revisions of Maple V), but the knowledge that I had been able to figure out how to use it at some point in time was a big plus, so I decided to try to use it anyway. After messing around a bit, I realized that the geometry package had the means to rotate points around other points. This would allow me to build what I call “extended wedge squiggles” rather easily once I got the hang of the syntax.

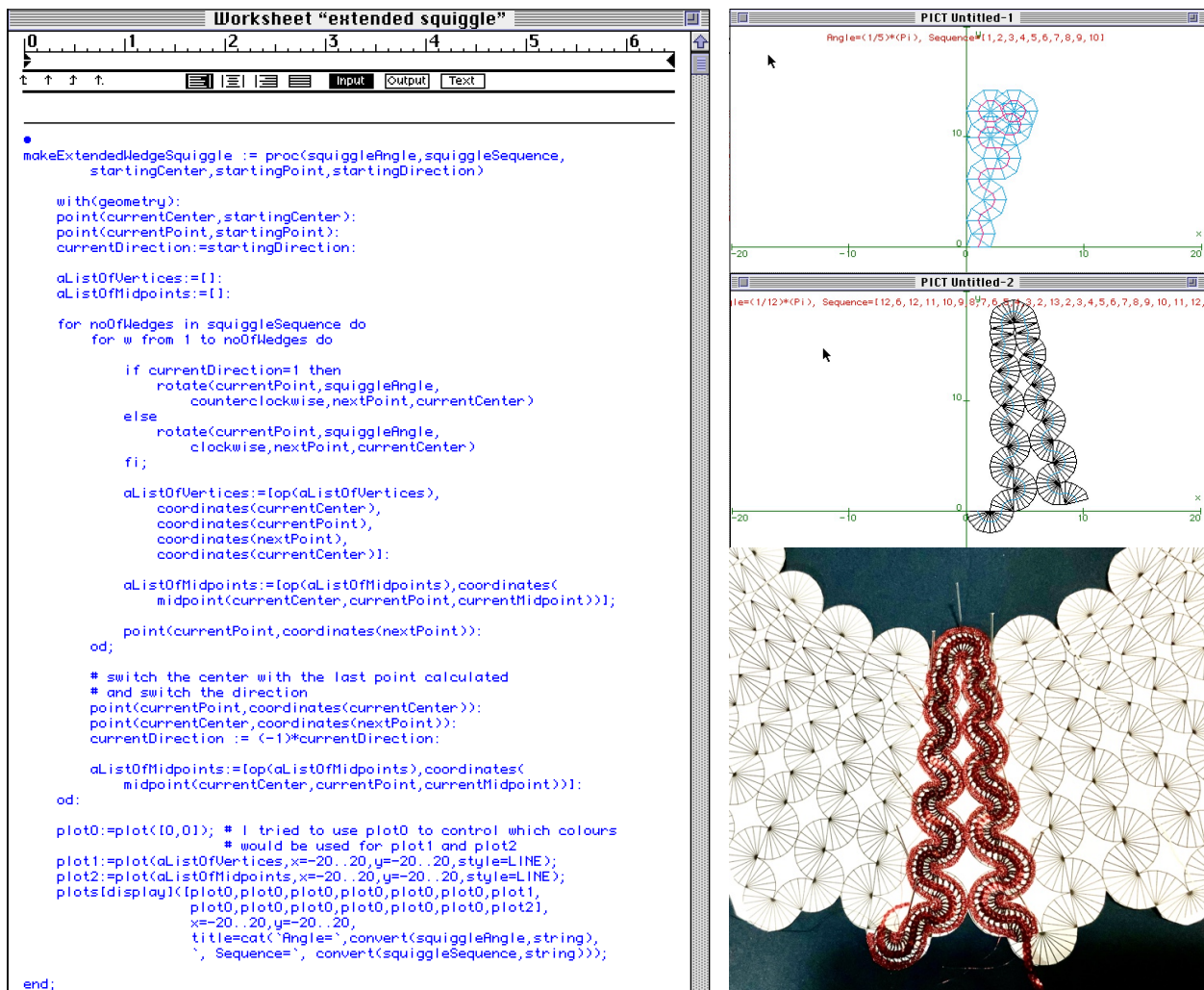
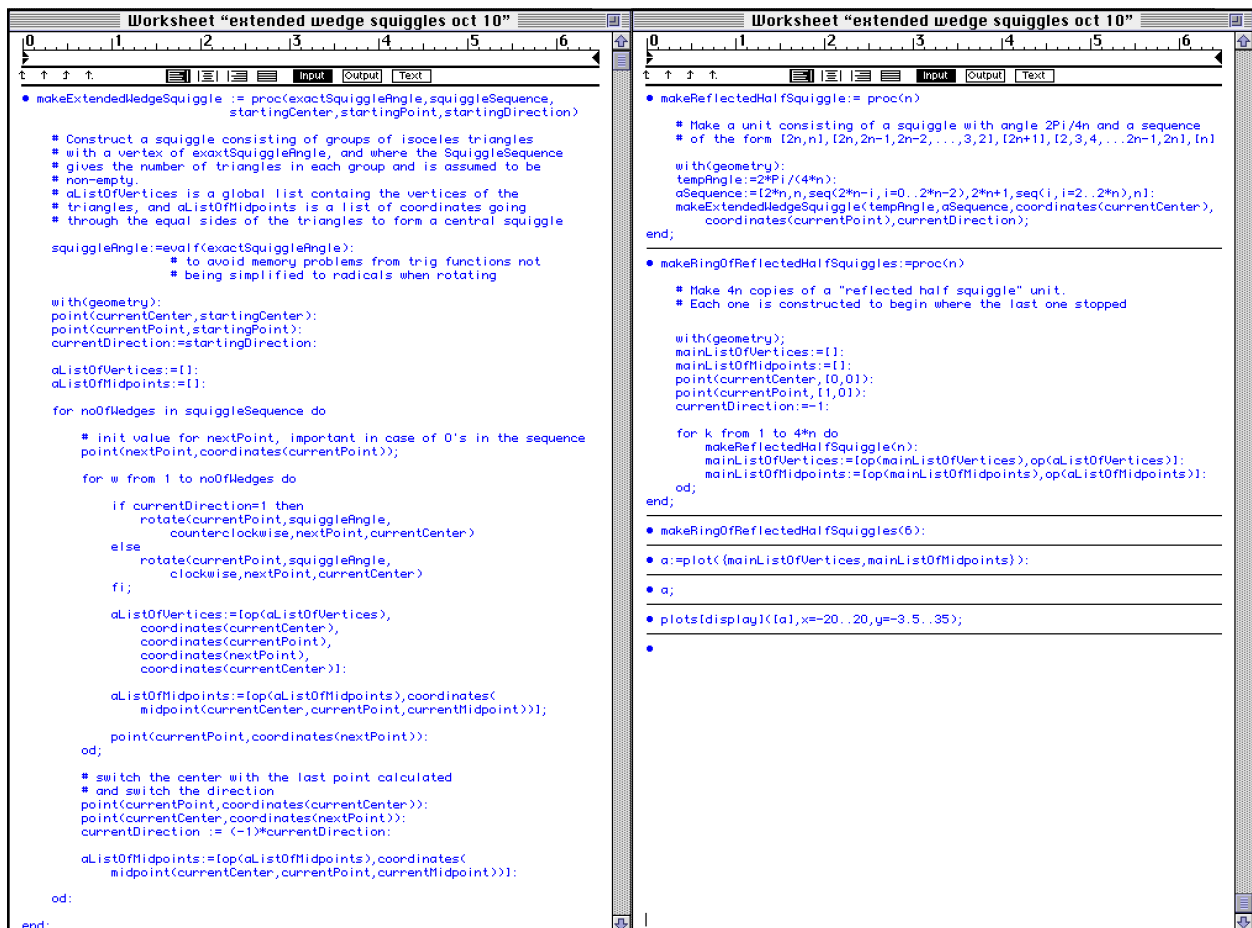


Figure 3: A Maple 5.0.2 worksheet for plotting extended wedge squiggles along with 2 sample plots. The last picture shows a squiggle that I crocheted using metallic thread around a length of sequins. It uses the angle and sequence shown in the second plot. The crochet is shown on top of a printout drawn in a vector graphics program.

While trying out sequences that I had used in the supplement to my Bridges paper, I noticed that I had inadvertently omitted a couple of numbers when typing at least one of them up. Most of the others looked okay, but for some reason there were some that didn't seem to work with my Maple procedure. Initially, I had a trouble figuring out why some of these angle/sequence combinations would execute quickly but others would run out of resources. In some of the cases that I tried, complicated sequences would work for some angles while simple ones would not work for others.

It turned out that for some angles, there were trig functions which were not being simplified, for example $\sin(\pi/8)$ was kept as $\sin(\pi/8)$ throughout the calculations even though $\sin(\pi/5)$ was replaced by an expression involving radicals. Presumably my use of so many rotations in squiggle construction caused this to become a serious problem. Replacing the exact angles with floating point approximations (using `evalf`) before performing any rotations made the system much more responsive, and I ended up being confident enough in it to write a bit of code to generate some squiggly rings that I had previously only tried drawing in vector graphics programs. While I was at it, I also fixed a bug that caused problems when using squiggle sequences which contain zeroes.

Figure 4 shows the updated code, and Figure 5 shows one of these squiggly rings plotted by Maple, as well as a piece of digital art incorporating this Maple plot and the crocheted fragment shown in Figure 3. If you look closely enough, you may be able to see the ghost of a 3.5" Maple install disk exerting its influence on the present.



```

Worksheet "extended wedge squiggles oct 10"
0 1 2 3 4 5 6
t ↑ ↓ ← →
Input [Output] Text
• makeExtendedWedgeSquiggle := proc(exactSquiggleAngle, squiggleSequence,
startingCenter, startingPoint, startingDirection)
# Construct a squiggle consisting of groups of isosceles triangles
# with a vertex of exactSquiggleAngle, and where the SquiggleSequence
# gives the number of triangles in each group and is assumed to be
# non-empty.
# aListOfVertices is a global list containing the vertices of the
# triangles, and aListOfMidpoints is a list of coordinates going
# through the equal sides of the triangles to form a central squiggle
squiggleAngle:=evalf(exactSquiggleAngle):
# to avoid memory problems from trig functions not
# being simplified to radicals when rotating
with(geometry):
point(currentCenter, startingCenter):
point(currentPoint, startingPoint):
currentDirection:=startingDirection:
aListOfVertices:=[]:
aListOfMidpoints:=[]:
for noOfWedges in squiggleSequence do
# init value for nextPoint, important in case of 0's in the sequence
point(nextPoint, coordinates(currentPoint)):
for w from 1 to noOfWedges do
if currentDirection=1 then
rotate(currentPoint, squiggleAngle,
counterclockwise, nextPoint, currentCenter)
else
rotate(currentPoint, squiggleAngle,
clockwise, nextPoint, currentCenter)
fi;
aListOfVertices:=op(aListOfVertices),
coordinates(currentCenter),
coordinates(currentPoint),
coordinates(nextPoint),
coordinates(currentCenter)):
aListOfMidpoints:=op(aListOfMidpoints, coordinates(
midpoint(currentCenter, currentPoint, currentMidpoint)));
point(currentPoint, coordinates(nextPoint)):
od;
# switch the center with the last point calculated
# and switch the direction
point(currentPoint, coordinates(currentCenter)):
point(currentCenter, coordinates(nextPoint)):
currentDirection := (-1)*currentDirection:
aListOfMidpoints:=op(aListOfMidpoints, coordinates(
midpoint(currentCenter, currentPoint, currentMidpoint)));
od;
end;
end;

Worksheet "extended wedge squiggles oct 10"
0 1 2 3 4 5 6
t ↑ ↓ ← →
Input [Output] Text
• makeReflectedHalfSquiggle:= proc(n)
# Make a unit consisting of a squiggle with angle 2Pi/4n and a sequence
# of the form [2n,n], [2n,2n-1,2n-2,...,3,2], [2n+1], [2,3,4,...,2n-1,2n], [n]
with(geometry):
tempAngle:=2*Pi/(4*n):
aSequence:=[2*n,n,seq(2*n-i, i=0..2*n-2),2*n+1,seq(i, i=2..2*n),n]:
makeExtendedWedgeSquiggle(tempAngle, aSequence, coordinates(currentCenter),
coordinates(currentPoint), currentDirection);
end;
• makeRingOfReflectedHalfSquiggles:=proc(n)
# Make 4n copies of a "reflected half squiggle" unit.
# Each one is constructed to begin where the last one stopped
with(geometry):
mainListOfVertices:=[]:
mainListOfMidpoints:=[]:
point(currentCenter, [0,0]):
point(currentPoint, [1,0]):
currentDirection:=1:
for k from 1 to 4*n do
makeReflectedHalfSquiggle(n):
mainListOfVertices:=op(mainListOfVertices), op(aListOfVertices):
mainListOfMidpoints:=op(mainListOfMidpoints), op(aListOfMidpoints):
od;
end;
• makeRingOfReflectedHalfSquiggles(6):
• a:=plot([mainListOfVertices, mainListOfMidpoints]):
• a;
• plots[display](a, x=-20..20, y=-3.5..35);
•

```

Figure 4: Worksheet with updated code as well as extra routines to allow the plotting of a squiggly ring.

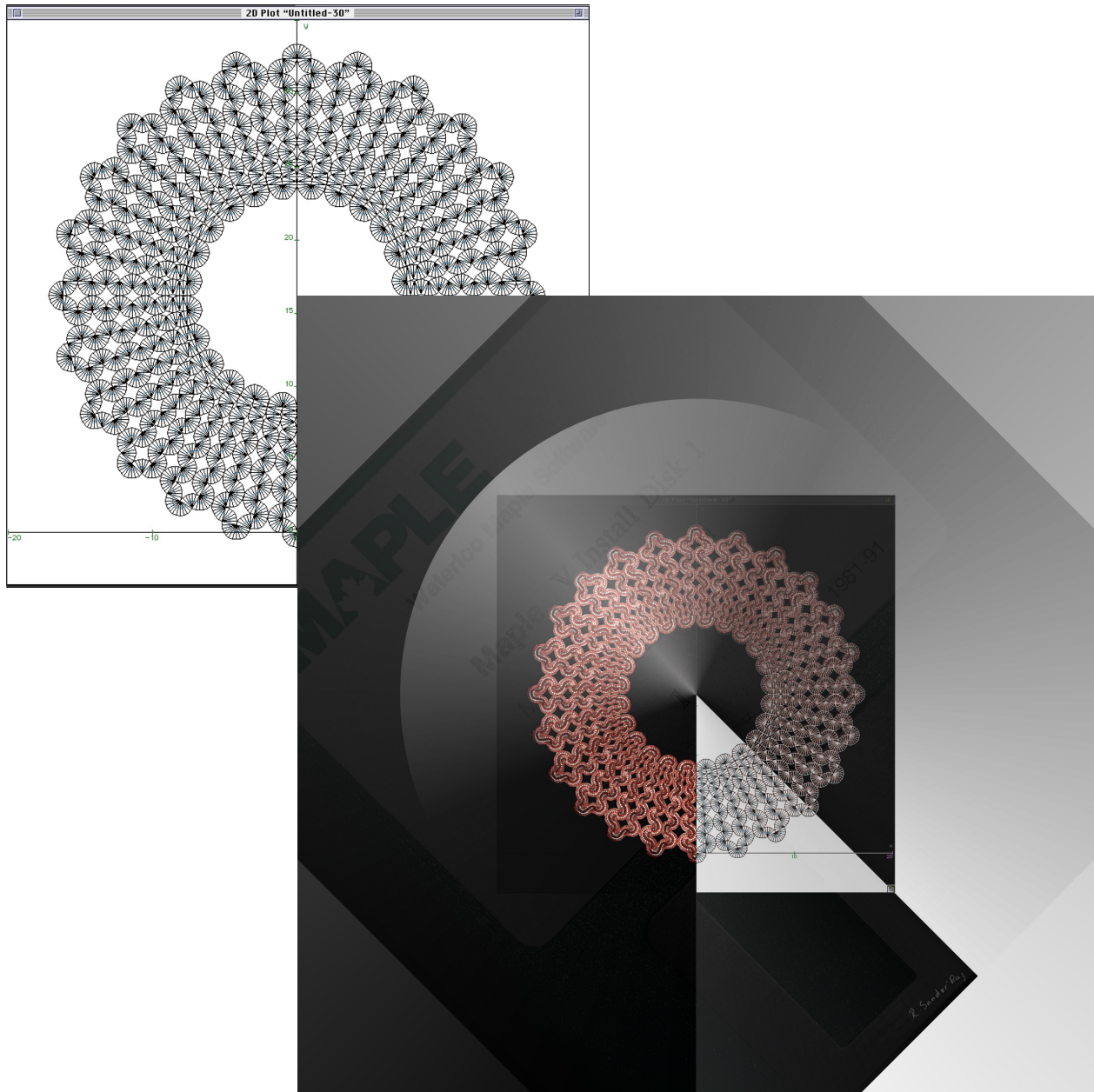


Figure 5: A squiggly ring plotted using the updated code, along with digital art mixing this plot with copies of the fragment of crochet shown in Figure 3.

Reference

- [1] R. Sunder-Raj. "Physical Representations of Polygon, Wedge, and Arc Squiggles." *Bridges Conference Proceedings*, Halifax, Canada, Jul. 27-31, 2023, pp. 469-472. <http://archive.bridgesmathart.org/2023/bridges2023-469.html>