Asemic writing using MetaFun in ConTEXt

Keith McKay

Abstract

The article describes ways to generate asemic writing using MetaPost, MetaFun, and LuaMetaFun in ConTEXt. The term asemic means no semantic content and thus asemic writing is writing without semantic content or meaning. It has a thriving community of artists producing many different styles which can best be described as a hybrid form of abstract art. Examples of hand-written and ConTEXt generated asemic writing will be presented along with examples of the code used to produce them.

About five years ago I stumbled across a hybrid form of abstract art known as *asemic writing*. Although I'm not an artist I found the works very appealing and as I looked further into the term asemic, I became even more intrigued when I found that it means something with *no semantic content* or, in layman's terms, *something without meaning*. The writing looks like it has meaning since in many cases it contains gestural lines connecting 'characters' but on closer inspection it has no meaning at all.

Asemic writing was given that name by two visual poets, Tim Gaze and Jim Leftwich, in the late nineties, although it can be argued that it has been around much longer.[1] Chinese characters and Islamic calligraphy to some western eyes may be thought of asemic but not by the cultures who wrote it. Vice versa, western Latin script could be thought of as asemic by those cultures. Going back even further, what are we to make of the cup and ring marks of Neolithic rock art? We don't know what they mean today but they must have had meaning to the people in those times.

Modern day asemic writing comprises a multitude of styles,[2] but my interest was in handwritten asemics because the randomness in the style appealed to me.

In the figures below I show some examples of asemic writing. Figures 1 and 2 show the variation of styles of asemic writing.[3] Figure 3 is one of my own pieces of asemic hand writing.[4]

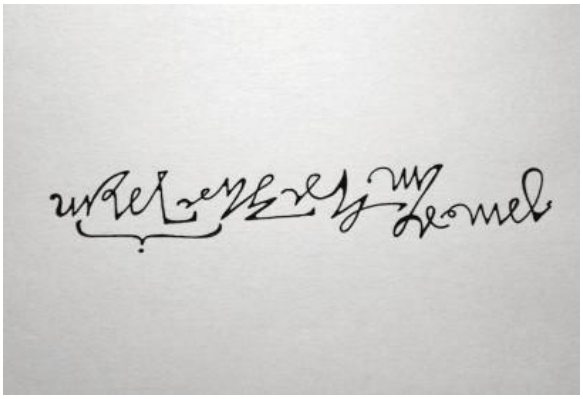

Figure 1: Asemic writing by Geof Huth.

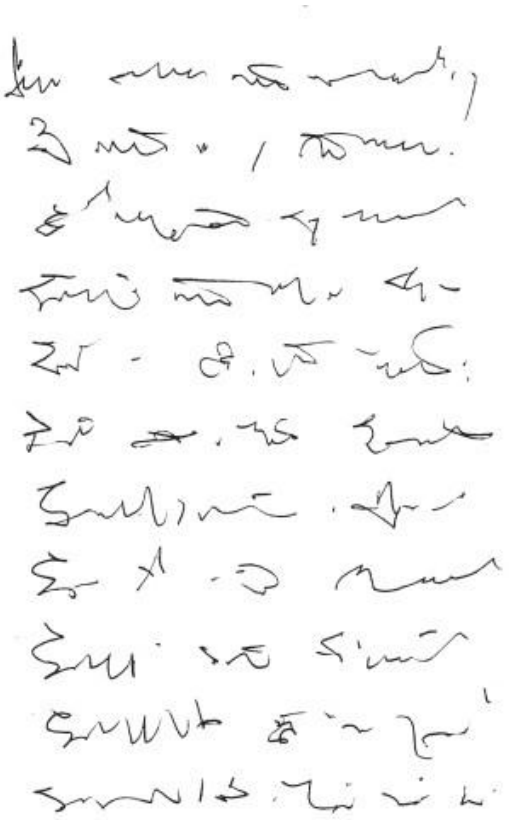

Figure 2: Asemic writing by Jim Leftwich.

At about the same time as I found asemic writing I was reading the MetaFun manual, and I thought maybe I could generate asemic writing using MetaPost and MetaFun code in ConTEXt. After all, the asemic writing looked like a path to which randomness had been applied and then drawn on the page with a pen. MetaFun has a randomized function but it wasn't what I was looking for so I set upon writing one, which was also a good way for me to start to learn some MetaFun coding. My first attempts were,

[1] See *Asemic: the Art of Writing* by Peter Schwenger, University of Minnesota Press, 2019.

[2] See *An Anthology of Asemic Handwriting* edited by Tim Gaze and Michael Jacobson, 2013. It can be downloaded from the Internet Archive.

[3] Both of these examples are taken from *An Anthology of Asemic Handwriting*.

[4] This piece was exhibited at: GRAPHOS — MUESTRA INTERNATIONAL DI ESCRITURA ASÉMICA, 01–18 Nov. 2023, Buenos Aires, Argentina.

doi.org/10.47397/tb/45-3/tb141mckay-asemic

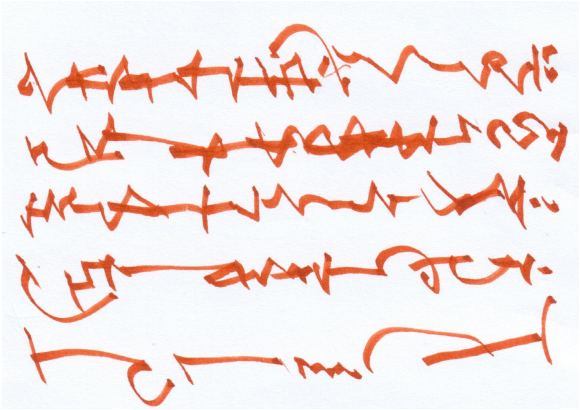

Figure 3: Asemic writing by Keith McKay.

to say the least, a bit clunky, although they worked and I was able to improve them with time.

My logic was as follows. Create a path of a certain length and decide how many points, i.e 'characters', it should contain. Iterate along the path and at each point add a small amount of randomness to the xpart and ypart of the point. Using these new points create a new asemic path.

To implement this I created two vardefs: One to generate a random number within a range of values, usually something like −0.1 to +0.1, and another to create the asemic path as described above.

```
vardef randnumRange(expr mini,maxi) =
  randnum := ((maxi - mini) * uniformdeviate(1))
             + mini;
  randnum
enddef;
```

The asemic path generator vardef has four variables: a path, the number of points in that path and the minimum and maximum for the range of a random number. The last two variables are used within the `vardef randnumRange()`.

```
vardef Asemic(expr aPath,nPoints,minii,maxii) =
path asemicPath;
mini := minii; maxi := maxii; nbPoints := nPoints;
asemicPath :=
  ((((xpart point 0cm on aPath)/72)*2.54)*cm,
   (((ypart point 0cm on aPath)/72)*2.54)*cm);
for x = 1 upto nbPoints:
  ycoord := ((ypart point(x/nbPoints) along aPath)/72)
            *2.54 + randnumRange(mini,maxi);
  xcoord := ((xpart point(x/nbPoints) along aPath)/72)
            *2.54 + randnumRange(mini,maxi);
  asemicPath := asemicPath ...
                {curl 100}(xcoord*cm, ycoord*cm);
endfor;
if ((point 0 along aPath)
    = (point nbPoints along aPath)):
  asemicPath := asemicPath ... cycle;
fi;
asemicPath % return
enddef;
```

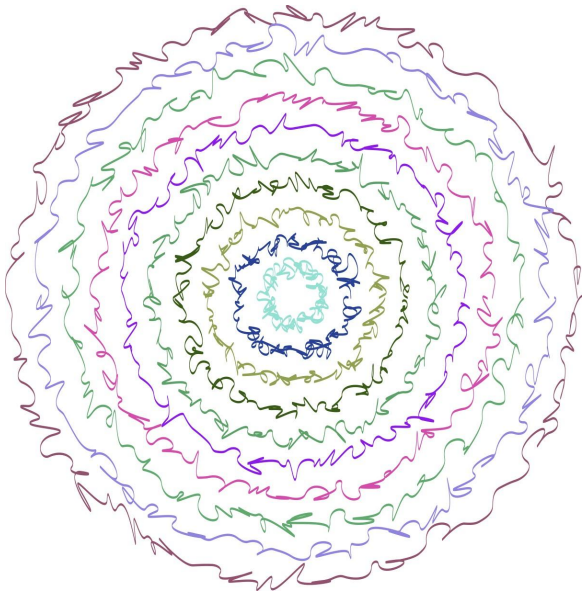

Figure 4: Concentric asemic circles.

In addition to the vardefs which created the asemic line, I wrote a small def to create a calligraphic styled pen with a randomly generated rgb colour.

```
def coloredPath(expr s, t) =
  withpen pencircle xscaled s yscaled t
    withcolor (uniformdeviate(1),
               uniformdeviate(1),uniformdeviate(1));
enddef;
```

Using these three macros I could create a number of asemic lines on a page with randomly generated colours, like so:

```
\startMPpage
path aPath;
numeric nPoints, mini, maxi;
nPoints := 100;
mini := -0.125; maxi := 0.125;
for i = 10 downto 1:
  aPath := (0cm, i*cm) -- (10cm, i*cm);
  draw Asemic(aPath,100,-0.125,0.125)
       coloredPath(.1mm,.4mm);
endfor;
\stopMPpage
```

Lines of text can get boring, so replacing `aPath` with `aPath:= fullcircle scaled(i*cm);` gives a series of randomly coloured asemic concentric circles, as shown in Figure 4. Indeed any path, open or closed, can be made asemic using our vardefs `Asemic` and `randnumRange`.

But we don't need to stop here. A page of text consists of lines of words made up of characters, and, these words are separated with spaces, so how can we use the asemic vardefs to create lines of asemic words? This was a little tricky to solve at first because there are a number of things to keep in mind. We have already decided on a path which has a certain number of points; we decide on the number of spaces between

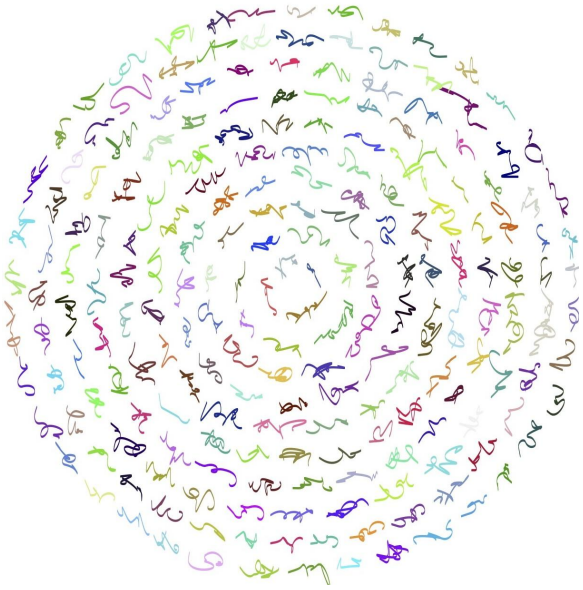

Figure 5: Concentric circles of asemic words.

the words and we can randomly generate the number of characters for each word. However, we must make sure that the last word does not overshoot the end of the line and to do this we must keep a running total of the number of points used along the path. When we get to a word that does overshoot the end of the line we truncate it so that it fits, thus giving the impression of justified text.

```
\startMPinclusions
% Put vardef randnumRange() here.
% Put vardef Asemic() here.
% Put def coloredPath()here
\stopMPinclusions
\startMPpage
path myPatha, myPathb, myPathc, myPathd, asubPath;
numeric nPoints, nchars, mini, maxi, minword, maxword,
        spaces, beginPath, endPath;
boolean ending;
nchars := 400; minword := 5; maxword := 10;
mini := -0.125; maxi := 0.125; spaces := 3;
for i = 10 downto 1:
  myPatha := (0cm, i*cm) -- (10cm, i*cm);
  myPathb := fullcircle scaled(i*cm)shifted(16cm,5cm);
  myPathc := fullsquare scaled(i*cm)shifted(5cm,16cm);
  myPathd := fulltriangle scaled(i*cm)shifted(15cm,
                                              16cm);
  nPoints := (nchars * (i/10));
  for j = myPatha, myPathb, myPathc, myPathd:
  ending := false;
  wordlength := round(randnumRange(minword,maxword));
  beginPath := 1;
  endPath := beginPath + wordlength;
  forever:
    if (nPoints - beginPath) < 18:
      ending := true;
      endPath := nPoints - 1;
      t := (beginPath/nPoints) * length j;
      u := (endPath/nPoints) * length j;
```

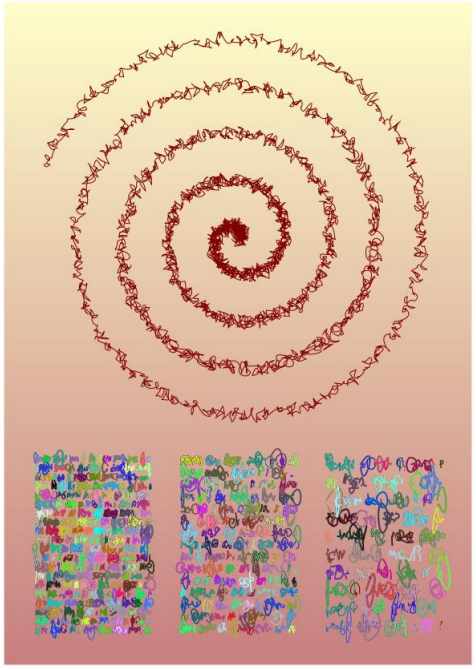

Figure 6: Asemic spiral with asemic writing.

```
      asubPath := subpath(t,u) of j;
      wordlength := (nPoints - 1) - beginPath;
      draw Asemic(asubPath,wordlength,mini,maxi)
           coloredPath(.1mm,.4mm);
    else:
      ending := false;
      t := (beginPath/nPoints) * length j;
      u := (endPath/nPoints) * length j;
      asubPath := subpath(t,u)  of j;
      draw Asemic(asubPath,wordlength,mini,maxi)
          coloredPath(.1mm,.4mm);
      beginPath := endPath + spaces;
      endPath := beginPath + wordlength;
      wordlength := round(randnumRange(minword,
                                       maxword));
    fi;
    exitif ending = true;
  endfor;
  endfor;
endfor;
\stopMPpage
```

Figure 5 shows concentric asemic circles consisting of randomly sized words with the density of points on the original circle adjusted for diameter. This is one of the asemic shapes produced in the above code. Readers are welcome to try it out; you'll need a ConTEXt distribution installed (`contextgarden.net`).

We now have the basic building blocks for pages of computer generated asemic writing. Figure 6 and the following page show some images that I have produced using ConTEXt.

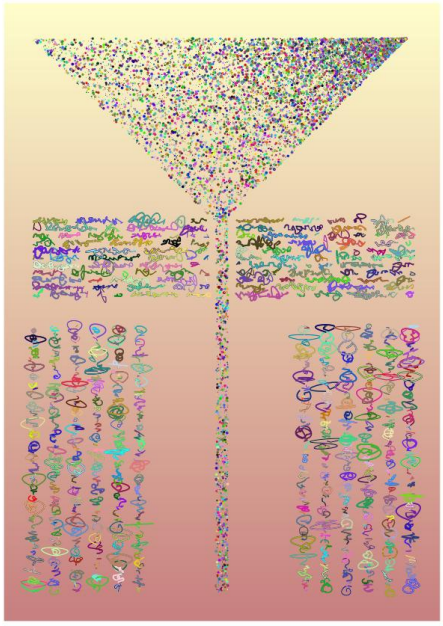

Figure 7: Horizontal and vertical asemic writing.

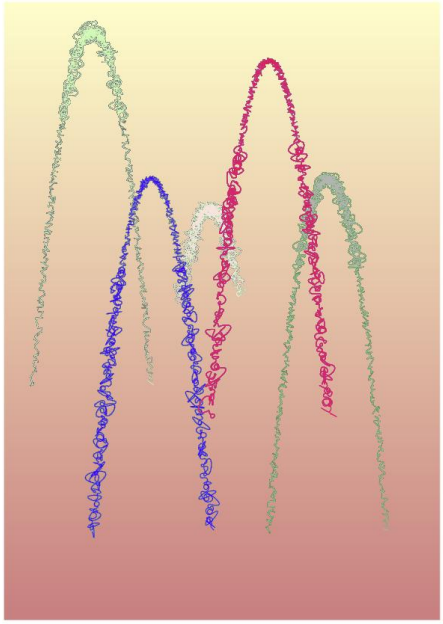

Figure 10: Asemic functions.

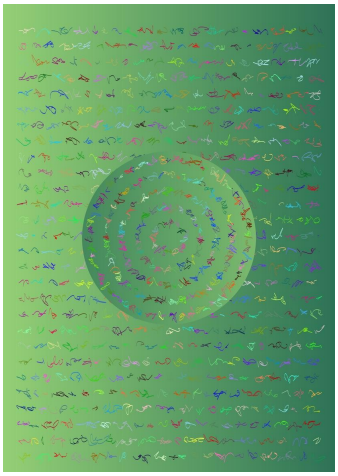

Figure 8: Dimpled Asemic writing.

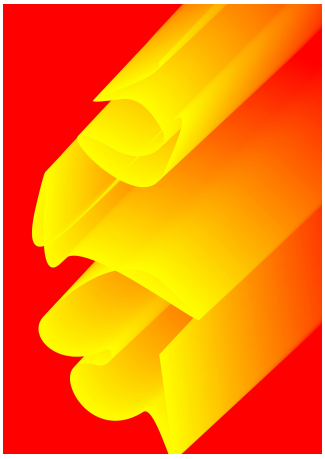

Figure 11: Large blended asemic character.

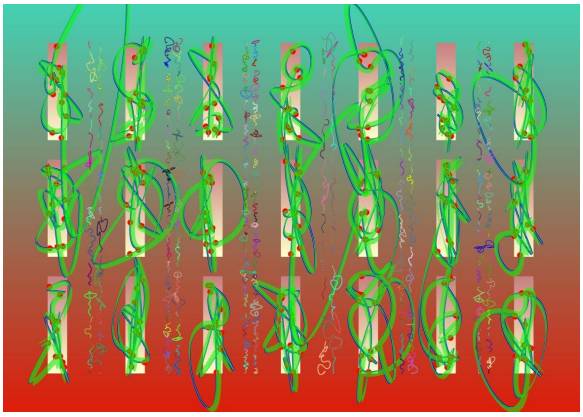

Figure 9: Asemic characters.

Figure 12: Asemic writing.

Recently, Hans Hagen suggested to me some improvements to the code which would speed up the calculation. In the LuaMetaFun manual there is some information about fast loops over paths. Normally if MetaPost has to find a point q on a path, it has to start at 1 and move along the path to find q, but LuaMetaFun has `for i within path : ... pathpoint ... endfor;`. In conjunction with the new primitive `arcpointlist`, this considerably speeds up the computation. The revised code is as follows:

```
vardef Asemic(expr aPath, nPoints, mini, maxi) =
  path aasemicPath; numeric nbPoints;
  nbPoints := nPoints;
  aasemicPath := aPath;
  aasemicPath := arcpointlist nbPoints of aasemicPath;
  aasemicPath :=
    for i within aasemicPath:
      randomiser(pathpoint, mini, maxi)
      .. tension 2.5 ..
    endfor
    nocycle
    ;
  aasemicPath
enddef;
```

with a new `vardef randomiser()` to randomise the point on the path.

```
vardef randomiser(expr p, mini, maxi) =
  (
    xpart p + randnumRange(mini, maxi)*cm,
    ypart p + randnumRange(mini, maxi)*cm
  )
enddef;
```

Along with Mikael Sundqvist, Hans Hagen has also been adding additional functionality to LuaMetaFun in the form of envelopes. I have used these to generate asemic writing. I will not go into detail about envelopes since it is outside my area of expertise, but in simple terms, an envelope is the closed path we get when we run a pen over a path. This may not seem very exciting; however, when used with `withshademethod " "` and `withshadecolors(colora,colorb)` some very interesting effects can be obtained. The LuaMetaFun manual goes into more detail and readers are encouraged to read it.[5]

Here is a code snippet in which I used envelopes to produce the shaded asemic words shown in Figures 13 and 14. The two figures are examples of MetaPost code taken from `tlhiv.org` to which I then added my asemic code.[6] An envelope (`asPathEnvelope`) is created over an asemic path (`asPath`) and two colours are randomly generated which are then used to fill and shade the envelope.

[5] The LuaMetaFun manual is in the ConTEXt distribution.
[6] `tlhiv.org/MetaPost/examples/examples.html`

Figure 13: Trefoil with asemic shaded writing.

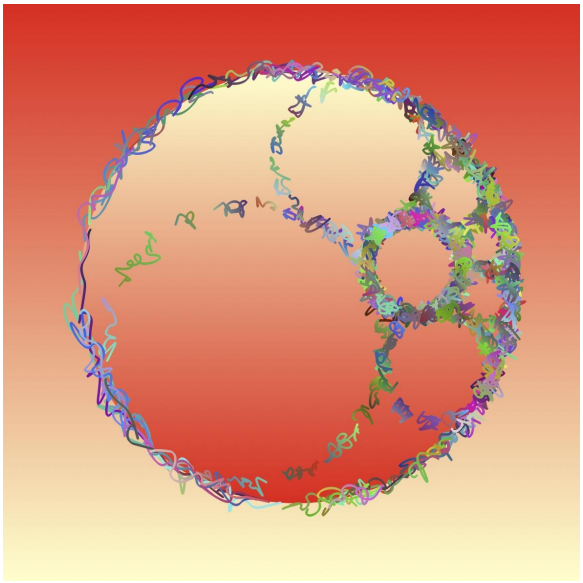

Figure 14: Asemic circles within circles.

```
% snip
asPath := Asemic(aPath, nPoints,mini,maxi);
asPathEnvelope := envelope pensquare scaled 0.3mm
                          rotated 45 of asPath;
definecolor [name = "Mycolor1",
             r = uniformdeviate(1),
             g = uniformdeviate(1),
             b = uniformdeviate(1)];
definecolor [name = "Mycolor2",
             r = uniformdeviate(1),
             g = uniformdeviate(1),
             b = uniformdeviate(1)];
fill asPathEnvelope yscaled 1
        withshademethod "linear"
        withshadecolors ("Mycolor1","Mycolor2");
% snip
```

Finally, I will show some images of the effect of envelopes on a series of paths forming the shape of a spiral with the paths asemic lines, and lines of asemic words.

Figure 15 is coded as follows:

```
pair A,B;
path p, psubPath, asPathEnvelope;
A = (0cm,0cm);B = (5cm,0cm);
p = A{dir 60} .. B;
fill fullsquare scaled 12cm withcolor 0.5[blue,white];
for i = 0 step 10 until 350:
  psubPath := (subpath((0.1 * length p),
                       (1 * length p))
                of p) rotated i;
  asPathEnvelope := envelope pensquare scaled 0.5mm
                    of psubPath;
  fill asPathEnvelope
       withshade define_linear_shade
                 ((cosd(i) * 0.5cm, sind(i) * 0.5cm),
                  (cosd(i) * 5cm, sind(i) * 5cm),
       0.5[blue,white],
       white);
endfor;
```

and Figures 16 and 17 have the asemic vardefs applied to them as shown previously.

This has been a whirlwind tour of asemic writing but I hope it has given the reader a taste of this hybrid art form. As I said at the beginning, I am not an artist and I should add, not a good coder either, but the experience of using ConTEXt with MetaPost, MetaFun and LuaMetaFun in creating asemic writing through code has led me down a new path. In my exploration of generating asemic writing through code, I still ponder if the computer-generated asemic writing is truly asemic since the code to produce it has to be syntactically correct and therefore has meaning. If it is incorrect then there is no output. So, I'll leave you with this thought. Can something with meaning produce something without meaning?

With thanks to Hans Hagen, Mikael Sundqvist, and the ConTEXt mailing list for help in problem solving.

◇ Keith McKay
Maryhill, Glasgow
Scotland, UK
mckaymeister (at) gmail dot com

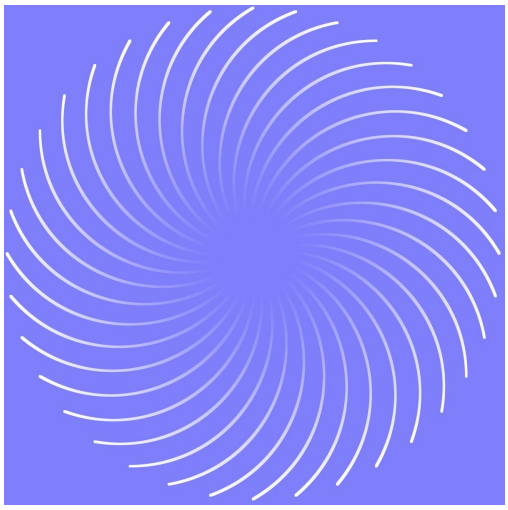

Figure 15: Spiral with envelopes.

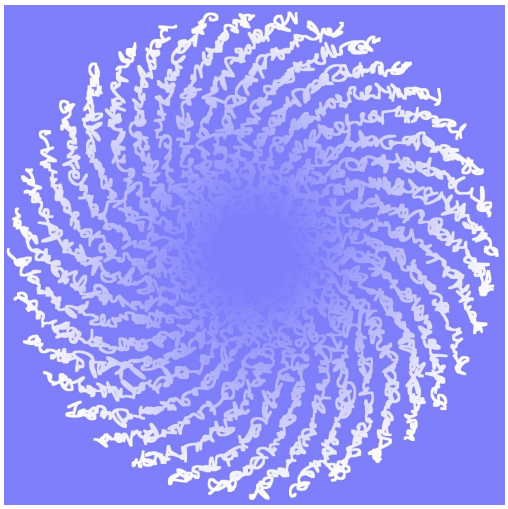

Figure 16: Spiral of asemic lines.

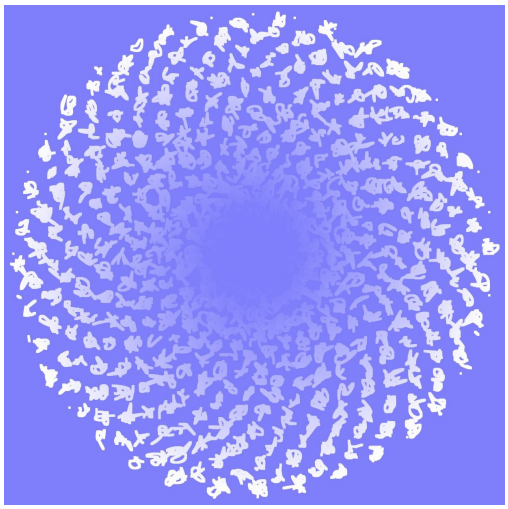

Figure 17: Spiral of lines of asemic words.