Confessions of an Illiterate Programmer

In the article by Hagen on Illiterate Programming (this issue of MAPS), there is a figure which requires a little bit of explanation. Some of you may have thought that it just looks like somebody has scribbled, or doodled on the page, or that it has been written by someone who has very bad hand writing. And what about the squiggly circle, square, and lines, what are they all about? If you're thinking, it makes no sense, or it has no meaning, well you are half way there. In fact what you are looking at is *asemic writing*, a term which is used to describe, *writing without meaning*, but in this case, the asemic writing has been created purely programmatically using MetaPost, MetaFun, and LuaMetaFun, in ConTEXt.[1]

I stumbled upon asemic writing a few years ago, and my initial thoughts were somewhat similar to the above, but that was a bit disingenuous of me. I soon realised that what I was really seeing was a type of abstract art which had no meaning, and that looks purely random, but makes you want to try to find meaning in it. Many asemic artists in fact give their asemic works titles, but in my mind that gives the work meaning and therefore it no longer is asemic, but maybe I'm being too pedantic. In my view it is up to the viewer to decide if they see meaning or not in the asemic writing.

After a while I began to think that, using the graphics capabilities of ConTEXt, I could create asemic writing programmatically, after all, what I was seeing was paths with a bit of randomness added to them, that is made to look like text positioned on a page. So I set to it and started thinking about how to randomise a path to make it look asemic. I started by looking at the `randomized` function in MetaFun thinking that it would be an easy way to randomise a path, however it wasn't what I was really looking for so I wrote a new function which would give a random number between a maximum and minimum number to be added to a point on a path. This was eventually morphed into two functions `randnumRange` and `randomiser` which are used in the function `Asemic`. The `Asemic` function takes a path with a known number of points i.e. characters, and adds a small amount of randomness to the xpart and ypart of each point in the path. So if you have a path which spans the text area of a page it's converted to a line of asemic text, which is fine, however, there is usually more than one word to a line of text.

So how to go about creating a line of text as asemic words. To do that we need to decide on a number of things:

- ☐ The number of points in a line. Points mean characters.
- ☐ The maximum and minimum numbers of characters in a word.
- ☐ The number of spaces between words.

Creating an asemic word from a path, is akin to creating a number of subpaths from the original path, with spaces in between, but we still need to know a number of other things about the position of each asemic word. This is done by checking if it is the first word on the line, the last word or a word in between. It's all a bit complicated but the function `AAAsemic` handles this nicely with the help of the `Asemic` function and returns a line of correctly spaced asemic words.

1. *Asemic writing using MetaFun in ConTEXt* by Keith McKay. TUGboat Volume 45 (2024), No 3.

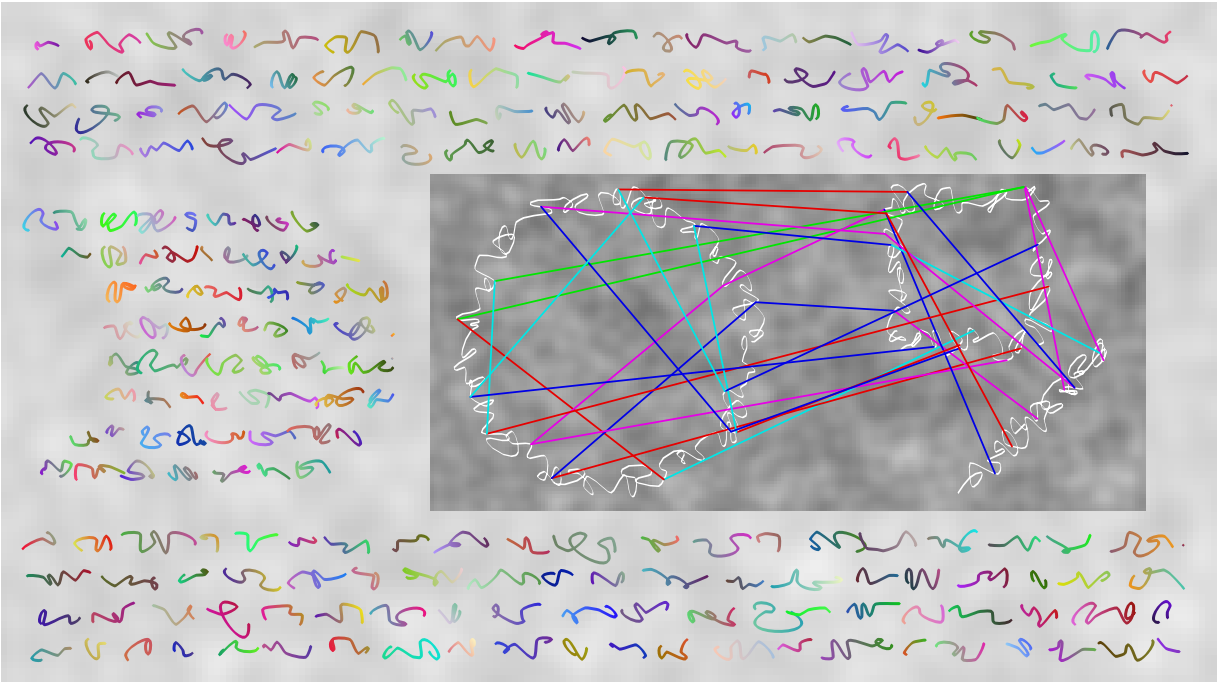
Now we can be a bit more creative. We can vary the length of the line of asemic words, add colour to them or even better we can use the LuaMetaFun function, `envelope`, and create an asemic word which begins with one colour and shades into another. All this is done for any number of lines of asemic words, within the magic of a `for` loop. There is one thing though. Since we don't know how many asemic words have been created we have to use a `forever:` loop to check if there is an asemic word (using `known`) in `asPath[]`. When we get to the place where there isn't an asemic word (using `unknown`) then we exit the `forever:` loop. We can also apply these asemic functions to different shapes to make them asemic as in the figure. I call them *asemic diagrams*, randomly generated asemic shapes connected with randomly coloured lines to randomly selected points on the shapes.[2]

All that is left to do now is to create a background for the figure. Here, ConTEXt provides us with a number of options, such as plain coloured backgrounds or by using the LuaMetaFun function `lmt_shade[]`, shaded coloured backgrounds. We define two randomly generated colours with `definecolor[]` using `uniformdeviate(1)` to calculate the `r g b` components. However, in the figure for the article I created a background using the hot of the press, LuaMetaFun function `lmt_noise[]`, which generates pseudorandom noise known as Perlin noise.[3] Developed by Ken Perlin, this method of creating noise is widely used to generate natural appearing textures on computer generated surfaces in the movie industry. In 1996 Perlin received an Oscar for its development.

You will have realised by now that there is a theme running through this note - *randomness*. Randomness has been used to create something which looks like writing, but has no meaning and turned it into a work of art, or more precisely, Computer Generated Abstract Art. The figure looks like something maybe for a textbook or a magazine, or, dare I say it, an article in a computer software journal, but of course, only an illiterate programmer would think that.

Keith McKay
Glasgow, Scotland, UK.

2. See my book, *Asemic Diagrams: Computer Generated Asemic Writing*. Available from Amazon worldwide. This book was made completely within ConTEXt using its graphics packages and certainly not with AI!
3. *Getting Noisy* by Hans Hagen and Keith McKay. TUGboat Volume 46 (2025), No.3



```
% This graphic is made by Keith McKay who has presented his work in
% \CONTEXT\ meetings and published about it in user group journals.

\startMPinclusions

% Convert any path to an asemic path.

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

% Creates a subpath of asemic words from aPath nchars long with a number of
% spaces inbetween each word. The wordlength is calculated using a randomly
% generated word between minwordlength and maxwordlength. The subpath is
% passed to Asemic(...) along with the wordlength, mini and maxi to
% create the asemic word. The position of each word is checked against
% whether it is the first word, the last word, or inbetween these two
% words. Each word is stored in position n of asPath[] and n incremented
% by 1. Each line should begin at 0 and end at nchars giving the impression
% of justification. Since this is a vardef asPath is returned.

vardef AAAsemic(expr aPath, nchars, spaces, minwordlength, maxwordlength, mini, maxi) =
    path asPath[], asubPath, asemicsubpath ;
    numeric wordlength, apathlength, ncharsused, n ;
    ncharsused := 0 ;
    n := 0 ;
    forever:
        wordlength := round(randnumRange(minwordlength,maxwordlength)) ;
        apathlength := length aPath ;
        % check if this is the last word
        n := n + 1 ;
        if ncharsused >= nchars - maxwordlength :
            wordlength := nchars - ncharsused ;
            asubPath := subpath(
                apathlength * (ncharsused/nchars),
                apathlength
            ) of aPath ;
            asemicsubpath := Asemic(asubPath, wordlength, mini, maxi) ;
```

```
            asPath[n] := asemicsubpath ;
            ncharsused := ncharsused + wordlength ;
            exitif ncharsused = nchars ;
        elseif ncharsused = 0 :
            asubPath := subpath(
                0,
                apathlength * ((ncharsused + wordlength)/nchars)
            ) of aPath ;
            asemicsubpath := Asemic(asubPath,wordlength,mini,maxi) ;
            asPath[n] := asemicsubpath ;
            ncharsused := wordlength + spaces ;
        else :
            asubPath := subpath(
                apathlength * (ncharsused/nchars),
                apathlength * ((ncharsused + wordlength)/nchars)
            ) of aPath ;
            asemicsubpath := Asemic(asubPath,wordlength,mini,maxi) ;
            asPath[n] := asemicsubpath ;
            ncharsused := ncharsused + wordlength + spaces ;
        fi ;
    endfor ;
    asPath
enddef ;

% Calculates a random number in the range mini to maxi used by
% randomiser.

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

% Adds a random ammount to the x and y part of a point on a path
% used with Asemic.

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

% Draws an asemic path using an oval pen with a randomly generated
% colour, although in this case the path is left black.

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

% Draws a coloured line between asemic shapes.

def coloredPathb(expr s) text c =
       withpen pencircle
        scaled s
        withcolor c ;
enddef ;

\stopMPinclusions

\startMPpage

% We define variables as we go.

path aLine, aShortLine, asPathEnvelope, asPath, formatPath[] ;
color c[] ;

c[0] := .9blue ;
c[1] := .9green ;
c[2] := .9magenta ;
c[3] := .9cyan ;
```

```
c[4] := .9red ;

% Create paths to be made asemic.

aLine      := (0cm,0cm) -- (16cm,0cm) ;
aShortLine := (0cm,0cm) -- ( 4cm,0cm) ;

% Create a noise background.

draw lmt_noise [
    bytemap    = 1,
    nx         = 200,
    ny         = 100,
    nz         = 1,
    iterations = 3,
    frequency  = 0.04,
    minimum    = 175,
    maximum    = 255,
    method     = "perlin",
] sized (17cm,9.5cm) shifted (-0.5cm,-0.5cm) ;

% As above but with noise only behind the asemic diagram.

draw lmt_noise [
    bytemap    = 2,
    nx         = 200,
    ny         = 100,
    nz         = 1,
    iterations = 3,
    frequency  = 0.04,
    minimum    = 75,
    maximum    = 225,
    method     = "perlin",
] sized (10cm, 4.7cm) shifted(5.5cm, 1.9cm) ;

% Draw four lines of asemic text at bottom of page;

for i = 0 upto 3 :
    asPath := AAAsemic(aLine, 150, 2, 3, 8, -0.2, 0.1) ;
    j := 1 ;
    forever:
        if known asPath[j] :
            asPathEnvelope := envelope
                pensquare scaled 0.3mm rotated 45
            of asPath[j] ;
            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
                shifted (0cm, i * 0.5cm)
                yscaled 1
                withshademethod "linear"
                withshadecolors ("Mycolor1","Mycolor2") ;
            j := j + 1 ;
        fi;
        exitif unknown asPath[j] ;
    endfor;
endfor;

% Draw eight lines of formated asemic text in middle of page.

formatPath[1] := aShortLine ;
```

```
formatPath[2] := aShortLine xshifted 0.5cm ;
formatPath[3] := aShortLine xshifted 1cm ;
formatPath[4] := formatPath[3] ;
formatPath[5] := formatPath[3] ;
formatPath[6] := formatPath[3] ;
formatPath[7] := formatPath[2] ;
formatPath[8] := formatPath[1] ;

for i = 1 upto 8 :
    asPath := AAAsemic(formatPath[i], 60, 2, 3, 8, -0.2, 0.1) ;
    j := 1 ;
    forever :
        if known asPath[j] :
            asPathEnvelope := envelope
                pensquare scaled 0.3mm rotated 45
            of asPath[j] ;
            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
                shifted (0, (i * 0.5cm) + 2cm)
                yscaled 1
                withshademethod "linear"
                withshadecolors ("Mycolor1","Mycolor2") ;
            j := j + 1 ;
        fi ;
        exitif unknown asPath[j] ;
    endfor ;
endfor ;

% Draw four lines of asemic text at top of page;

for i = 0 upto 3 :
    asPath := AAAsemic(aLine, 150, 2, 3, 8, -0.2, 0.1) ;
    j := 1 ;
    forever:
        if known asPath[j] :
            asPathEnvelope := envelope
                pensquare scaled 0.3mm rotated 45
            of asPath[j] ;
            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
                shifted (0, (i * 0.5cm) + 7cm)
                yscaled 1
                withshademethod "linear"
                withshadecolors ("Mycolor1","Mycolor2") ;
            j := j + 1 ;
        fi ;
```

```
        exitif unknown asPath[j] ;
    endfor ;
endfor ;

% Draw the asemic diagram, setup paths and create asemic paths.

path p, q, l, apPath, aqPath, alPath ;
pair alongp[], alongq[], alongl[] ;

p := fullcircle scaled 4cm shifted ( 8cm, 4.25cm) ;
q := fullsquare scaled 2cm shifted (13cm, 5.25cm) ;
l := (13cm,2.25cm) -- (15cm,4.25cm) ;

apPath := Asemic(p,150,-0.2,0.2) ;
aqPath := Asemic(q,100,-0.2,0.2) ;
alPath := Asemic(l, 50,-0.2,0.2) ;

for p = apPath, aqPath, alPath :
    draw p
        withpen pencircle xscaled 0.05mm yscaled 0.25mm rotated 30
        withcolor white
    ;
endfor ;

% Choose random points along each asemic path.

for i = 0 step 1 until 7:
       alongp[i    ] = point randnumRange(0,0.5) along apPath ;
       alongq[i    ] = point randnumRange(0,0.5) along aqPath ;
       alongp[i + 8] = point randnumRange(0.5,1) along apPath ;
       alongq[i + 8] = point randnumRange(0.5,1) along aqPath ;
       alongl[i    ] = point uniformdeviate(1)   along alPath ;
endfor;

% Sort the points.

sortlist(alongp)(ypart) ;

% Draw lines between points of the asemic paths.

drawoptions(coloredPathb(.25mm) c[floor(uniformdeviate(5))]);

for j = 0 step 1 until 7:
    draw (alongq[j    ] -- alongp[j    ]) ;
    draw (alongp[j + 8] -- alongq[j + 8]) ;
    draw (alongp[j    ] -- alongp[j + 8]) ;
    draw (alongq[j + 8] -- alongl[j    ]) ;
endfor;

drawoptions();

\stopMPpage
```