Loss Pequeño Glazier

Code, Cod, Ode: Poetic Language & Programming or: Pequeño Lector/Klein Leser: Es gibt (See)lachsfilet?

2005

https://doi.org/10.25969/mediarep/17669

Veröffentlichungsversion / published version

Zeitschriftenartikel / journal article

Code, Cod, Ode: Poetic Language & Programming or: Pequeño Lector/Klein Leser: Es gibt (See)lachsfilet?

By Loss Pequeño Glazier

No. 34 – 2005

Abstract

Mutation or modulation of words manifest orthographic relations between variants but also sometimes suggest more elusive relations. Much importance can be seen in the specificity of language, especially considering the sum of variations of a single word in different languages. The word itself is a solid object at the center of such a set of permutations. The meaning of a sum of such variants can be likened to an array in programming. An array object can be greater than the sum of its parts, a concept that ties into Cubism as well as to poetry where languages mix. Other array poetry suggests geometric structure; this is poetry that creates meaning from empty space as much as from its solid textual areas. This is similar to the way that architecture creates meaning from empty spaces, as seen notably in uses of the arch. The structural strength of empty space can also be seen in a number of postmodern poems, where such space is integral to their expressiveness.
These poems also use array concepts to inform the poem. It is useful to look at examples of code in my own work, which uses arrays and empty space as solid material in strings. What is of use in this method is the concept of precise poetic analysis, of the relevance of position, location, and structure as crucially important in reading code as poetic material.

"Deep in the mechanism
of the smallest zone
the drunken waltzers
the elephants tango ..."
-- Charles Stein

"Released from inexplicable code."
-- Steve Benson

Somehow I find the words "code", "cod", and "ode" to be an excellent starting place for a consideration of poetic language and programming. There's a way the three words are not only related, but suggested movement through mutation, or modulation. Of course the mind knows these relation is merely an accident, as the relation is only through orthography, not meaning, but the poet in one cannot help but try to find a more elusive relation.

The modularity and mutability of language was especially noticeable at the cafeteria today, when the feature of the day, Seelachsfilet, which is pollack (a fish related to cod) was listed on a poster menu. Apparently, the pollack ran out and they decided to substitute salmon filet instead. So instead of making a new poster they simply crossed out the "See" portion of the word, converting "Seelachsfilet" to "Lachsfilet" and thus "pollack" to "salmon". This was a more meaningful mutation, also orthographic, and one that gives a better sense of movement through changes in, or loss of, letters.

The topic of poetic language and programming always raises some interesting issues. On the one hand are programmers who insist that programming is a skill and simply a less-than-compelling means to an end. On the other hand, are programmers who are skilled and who have poetic intentions but who find such skill to be not quite enough on their own. Once you know how to write code, randomize, and create programs, what do you want to say? One programmer I know has said of code, all he sees is structure, or the architecture of the statements. Where does the connection between the code and the poetry of the coded object lie?

Specifically, as to language, how does programming "make it new"? How is it language? What specifically do we mean by "language" and how is language specific?

To look at how language is specific, I'd first like to consider a single word, say "mother", and look at how it appears in different languages.

mother	English	mama	Rumanian
mam	Welsh	nënë	Albanian
máthair	Gaelic	meter	Greek
mère	French	mat'	Russian
madre	Spanish	motina	Lithuanian
mãe	Portuguese	mayr	Armenian
madre	Italian	madar	Persian
mater	Latin	matar	Sanskrit
Mutter	German	ama	Basque
moeder	Dutch	mare	Catalan
móoir	Icelandic	ä iti	Finnish
moder	Swedish	anya	Hungarian
matka	Polish	anne	Turkish
matka	Czech		

There are similarities, of course, due to historical reasons and because of affinities between language families. But I'd like to ponder for just a moment, in languages that are so different, how a single word can appear so similar. I would suggest that a single word can have many modulations, variations across languages, without losing meaning. That is to say these words form a set or series of variants of a lexical unit. For humans, the concept of "mother" is, rather than its spelling in one language, the sum of the variations across all languages, remarkably motile and supple in meaning, mother is the sum of the differences of its fixed form.

"Mother" is a fixed object. As solid an object as Ezra Pound describes:

"AN OBJECT

This thing, that hath a code and not a core,

Hath set acquaintance where might be affections,

And nothing now

Disturbeth his reflections."

Or an object as Joan-Elies Adell writes in "For Every Object":

"I hold them,/invisible to my eyes, and play with them unafraid,/feeling the pointed tips, the sharp edges,/like fragments of language, hiding behind life/ready to cut."

Or otherwise, it is like a moment in motion frozen in time. As Pound sketches:

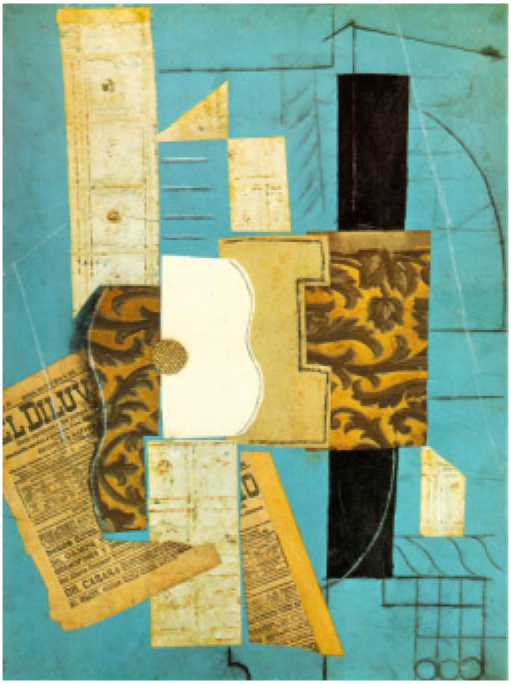
"IN A STATION OF THE METRO

The apparition of these faces in the crowd; Petals on a wet, black bough."

Such a language object is solid and fixed but at the same time it captures motion. It is a specific language item consisting of a number of permutations. It is and is not all of the permutations, depending from where you are looking in language. In programming terms, this would be called an array. An array is a collection of objects that share a single variable name, differentiated only by where they are located in the collection, a collection of parts whose sum is greater than the whole.

A Structure of Parts

Such a vision of the sum of the parts being greater than the whole is, coincidentally, not unlike Cubism. If we look, for example, at a work like Pablo Picasso's Cubist collage "Guitar", we see the following.



Here, fragments of "guitar" material, frets, neck, halves of guitar bodies, and other related images, lie scattered on a on a mostly blue background. There is a sense to which the different parts of a guitar can be shown and it is the collection of these parts, the "guitar array", one could say, which better defines a guitar than a photo-realistic image. (This is, perhaps, because a photo realistic image must always suffer from the subjectivity, or tyranny, of perspective. It can only be viewed from one viewpoint. The array solves this by providing a multitude of viewpoints in one plane.)

At the core of these collections of variants, I would like to look at how language mixes in a given work.

For example, observe how languages mix in the following poem extract. By using more than one language, for example, the expression "Cuban sol", which is "sun" but also carries the suggestions of "soul", one sees a work, by using multiple languages, where the sum of the parts is greater than the whole.

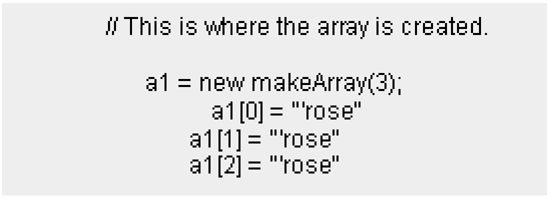
> "ON YOUR MARX. 'She is tropical, a Caribbean breeze, Cuban sol.' The radiant birds exist in cages in two's -- parakeets, lovebirds, and sulfur-crested cockatoos. The corner for birds, amor. Half-shell fountain, miniature orange trees form walkways, immensely pleasing to the self sense even as there are sones o salsa in waiting rooms. An effusively tiled Sevilla. Socialist radio of the Revolución."

A similar sense of the parts being greater than the whole occurs in Gertrude Stein's famous utterance:

"A rose is a rose is a rose."

Here, the parts that are greater than the whole are all the same, "rose", and yet placing them as an array seems to almost geometrically increase the overall meaning of them.

As an array, this would appear as something like:



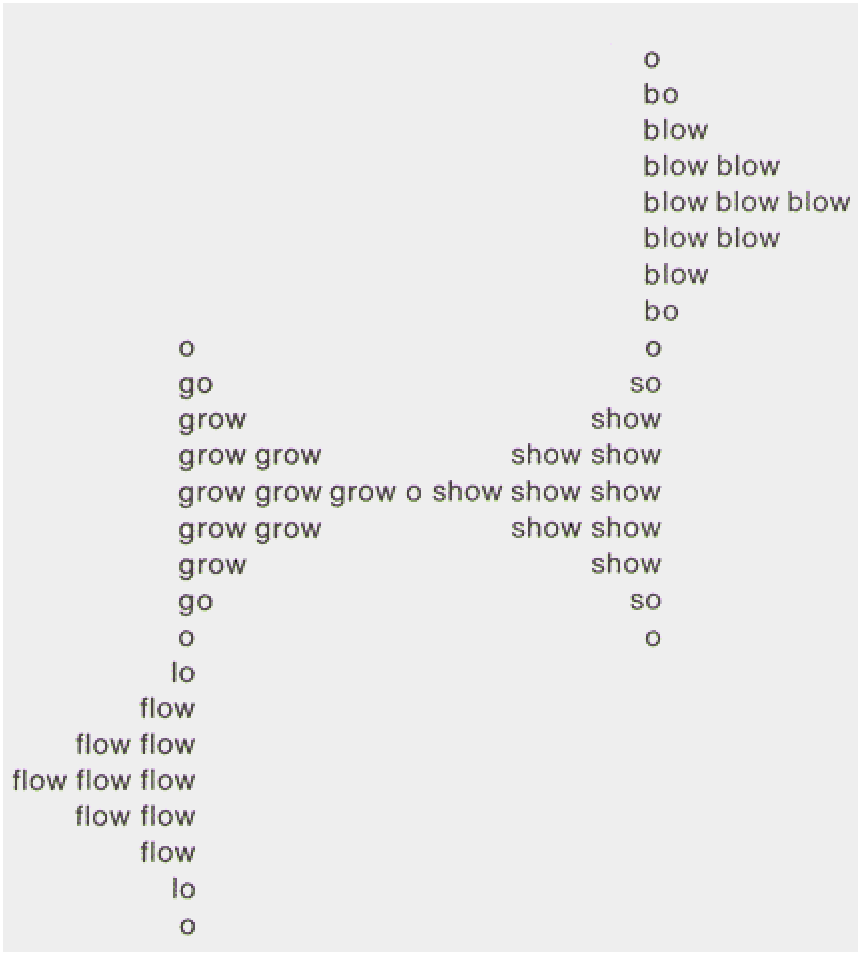
```
// This is where the array is created.

a1 = new makeArray(3);
a1[0] = "'rose"
a1[1] = "'rose"
a1[2] = "'rose"
```

In this case, though I am referring to this utterance as an array, we can't strictly consider the phrase an array since the elements ("rose") are all identical. This is a good instance for us to see how such elements can begin to take on geometric value. In other words, the words rose are like pillars in a colonnade where "rose" is the anchor for each pillar and the intervening explanation is empty. ("Rose" is never defined other than being "rose".)

How then is language capable of suggesting a structure? How is it geometric, structural? How does the visual produce meaning?

To answer this, I would like to look at Gomringer's "O". A poem which like the arches suggested by "a rose", creates meaning from empty space as much as from its solid textual areas.



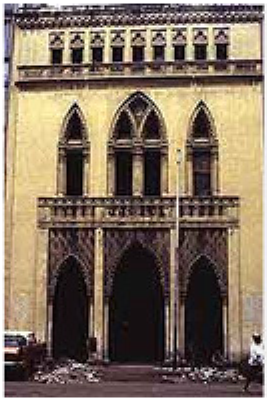
```
                                           o
                                           bo
                                           blow
                                           blow blow
                                           blow blow blow
                                           blow blow
                                           blow
                                           bo
             o                             o
             go                           so
             grow                       show
             grow grow             show show
             grow grow grow o show show show
             grow grow             show show
             grow                       show
             go                           so
             o                             o
           lo
         flow
    flow flow
flow flow flow
    flow flow
         flow
           lo
             o
```

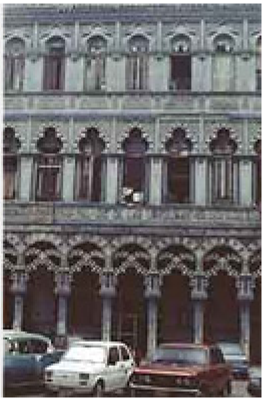
In this poem, we see a similar empty space of meaning which, like the arches suggested in "A rose" create a situation where nothing is actually stronger than textual content.

Architecture as Meaning

"There is nothing!"
-- William Carlos Williams

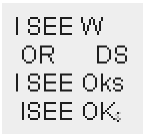
Arches express the ultimate in a play between structure and empty space, as if the architectural were paradigmatic of a kind of structural view of textual construction. With arches as well, one sees not a single item but a totality, a structure of parts. Here are some arches, examples of Havana's marvelous Moorish architecture.





The same triangle shapes one sees in the poem "O" are present above the columns in the arches. Indeed, such figurations create structural strength in order to allow spaces to exist. In this context, the spaces could be seen as the point, the solid matter of the architectural conversation.

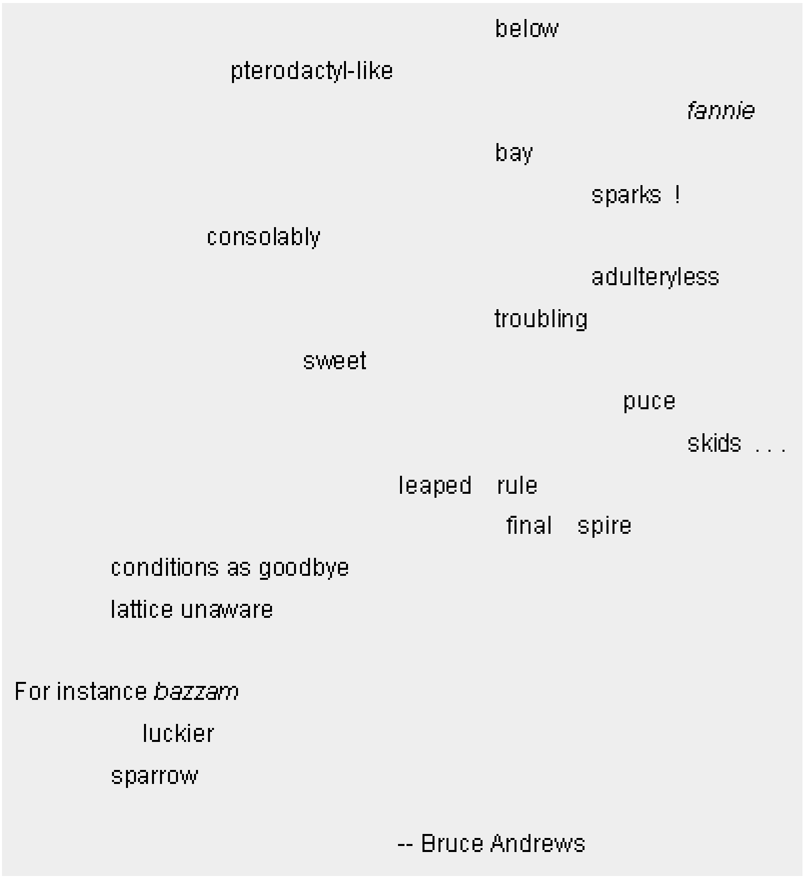
We see a similar engagement of the structural strength of empty space in Hannah Weiner's "I See Words".



```
I SEE W
 OR     DS
I SEE Oks
 ISEE OKs
```

Here, the linebreak after the first "W" and the space between "or" and "ds", as examples, create an archway, opening, or structural frame in which the simple "I see words" takes on much more meaning in its altered form than in its poetic representation.

A similar solidity of empty space is evident in "I Can't Resist the Craving for Another Inhalation" by Bruce Andrews:



below

pterodactyl-like

fannie

bay

sparks !

consolably

adulteryless

troubling

sweet

puce

skids . . .

leaped rule

final spire

conditions as goodbye

lattice unaware

For instance *bazzam*

luckier

sparrow

-- Bruce Andrews

Here we see an example of a text where the empty spaces are as important as the words themselves, the spaces offering pauses, timing, and a real counterpoint to the meaning the words express. This poem would not be the same if the words were all compressed; the spaces give the poem its expressiveness.

In "Fidel" by Andrews, the words seem to form an arch by virtue of their placement on the page. This combines the best of the moorish arch with the sensibility of Gomringer:

```
                                                  pee axe eel map sex am
                                                case ape pal lax
                                             pal leap lax ease
                                          ale leap seam
                                      alm amp ease sax lamp lax eel
                                 amp alm pax eel sex
                            pax peel map pea ma am ale same amp
                       seam ape amp sax sex peel seam leap
                  sex amp seam leap sax pee ape alm
              lap lamp pal ale ma pee alm axe
           ma lax pal ape ease am amp sax lap
pax peel lamp sax lap amp alm seam
  leap lax
    ale lamp leap
        lamp eel alm
           sex peel ape am map axe
             ale pal ease am eel pee seam
              sax peel pea axe map sex ape amp
              lax leap alm lpa lamp pax ma ale ape
              am axe pea lamp amp leap
             amp alm leap axe pea lamp
           pal ale am
              sex sax
                alm am map eel sex axe
                   sex alm am ale lamp pal
```

Empty space is crucial in this poem, of course, since its tight shape would not exist if it weren't juxtaposed against the white space to the left. But also, this poem is also an example of permutation, since it solely employs a finite number of letters in different combinations, with only one letter, "e", used more than once in a word. In fact, the poem could be considered an array since all the words used come from the letters of the one word, "example". (The "e" can be repeated in these permutations because "example" also contains two e's).

At this point, I would like to look at two examples of code in my own work. One from "Bromeliads" and the other from "Io Sono At Swoons", both of which are available on my EPC Author Page (http://epc.buffalo.edu/authors/glazier/).

In "Bromeliads" variant line possibilities are used to produce alternate readings of the text. These are technically arranged in arrays as follows:

```
// This is where the array is created.

     a1 = new makeArray(2);
             a1[0] = " 'Three simple words: crack dot com.' Ambience or
                    confrontation. "
              a1[1] = "I'm sure there is a white house hand in there somewhere.
                    'Colonia' "
```

I would like to note the structural and visual shape of this two-element array, and to suggest that the array here is not only content-based but also a visual way of writing. This structural arrangement is a way to organize the possibilities of the onscreen event and as such, the holes here are solid and meaningful, ways of arranging such possibilities. These are possibilities which exist as manifest in the code but can only be realized in the onscreen event of the code's execution.

I would compare this to another work, "Io Sono At Swoons", a work that assembles texts from a collection of strings, or data elements. The aesthetic contours of "Io Sono" was created by adding spaces, tangible nothing, or holes, into the strings. The result of adding such numbers of spaces into strings could not be anticipated. Thus the only way to work was to insert spaces, compile, then observe the displayed results of the spaces in the onscreen event. Working with the holes, or empty space, became a means to work with the poem.

What is of use in this method is the concept of precise poetic analysis, of the relevance of position, location, and structure. The arrangement of words in a poem or in a program have intense value. Like the empty space beneath arches in a colonnade, writing and programming is a dance around the architectural spaces of its parts.

Words themselves are modular and programmable. As smoothly as "code" flows to "cod" and then to "ode", poetic language is always engaged in the play of its parts.