The Aesthetics of Generative Code

Geoff Cox, BA, MA (RCA).
CAiiA-STAR (Science Technology Art Research),
School of Computing, University of Plymouth.
geoff@generative.net

Alex McLean, BSc.
State51, London.
alex@state51.co.uk

Adrian Ward, BSc.
Sidestream, London.
ade@sidestream.org

http://www.generative.net

if (

Abstract

Aesthetics, in general usage, lays an emphasis on subjective sense perception associated with the broad field of art and human creativity. Drawing particularly on Jonathan Rée's *I See a Voice: A Philosophical History* (1999), this paper suggests that it might be useful to revisit the troubled relationship between art and aesthetics for the purpose of discussing the value of generative code. Our argument is that, like poetry, the aesthetic value of code lies in its execution, not simply its written form. However, to appreciate generative code fully we need to 'sense' the code to fully grasp what it is we are experiencing and to build an understanding of the code's actions.

To separate the code and the resultant actions would simply limit the aesthetic experience, and ultimately limit the study of these forms - as a form of criticism - and what in this context might better be called a 'poetics' of generative code.

) {

Aesthetics

'The taste of the apple… lies in the contact of the fruit with the palate, not in the fruit itself; in a similar way… poetry lies in the meeting of poem and reader, not in the lines of symbols printed on the pages of a book. What is essential is the aesthetic act…' [1]

From the Greek ‘aisthesis’, aesthetics is broadly defined as pertaining to material things perceptible by the senses, and is more precisely defined by Baumgarten in *Aesthetica* (1750) defining beauty as ‘phenomenal perfection’ as perceived through the senses; with aesthetics ‘pertaining to the beautiful or to the theory of taste’ [2]. Thereafter in general usage, there remains an emphasis on subjective sense perception, but with particular reference to aesthetics and beauty generally associated with the broad field of art and human creativity. This applies despite Kant’s attempt to distinguish beauty as an exclusively sensuous phenomenon and aesthetics as a broader science of the conditions of sense perception [3]. For the purposes of our argument, we will retain this broader use of the term ‘aesthetics’, and add the proviso that there is an ideology to aesthetics that lies relatively hidden and difficult to perceive critically. This ideological aspect lies outside the scope of our paper but it is worth noting Slavoj Zizek’s evocative description of ideology - the ‘generative matrix’ [4] – that analogously expresses the generative code beneath the action. The suggestion, in keeping with this paper, would be that this requires a certain transparency to open it to criticism. We hope that revisiting the idea of the limits of aesthetic experience might serve to resolve some of the oppositions between theory and practice, and intellectual/physical division of labour involved in the production of generative art works. These issues are all too easily overlooked in an over-concentration on aesthetic outcomes that are all often reduced to subjective judgement and taste.

Limits

In discussions of aesthetics, the predominant philosophical legacy has been that any theory of art is predicated on the ‘specific characterisation of the senses’ [5]. It is now generally accepted that sense perception alone is simply not enough unless contextualised within the world of ideas [6]. Similarly, the world of multimedia is all too easily conflated with a multi-sensory experience (of combining still and moving image, sound, interaction and so on [7]) as if without a priori understanding of the integrated system (the body-machine) and its underlying code - that would include social and discursive frameworks.

Aesthetic theory has tended to collapse experience into what is perceived through the five senses, whilst privileging sight and hearing over touch and taste, leaving smell ‘at the bottom of the heap’ (Laporte’s *History of Shit* comes to mind) [8]. Subsequently there has been a recognition that this separation of sensual experience is inadequate and that a more systematic approach is called for that recognises the body as a whole as an integrated system. However, the legacy of the overall

(able-bodied) reductive approach is felt in the field of arts where the five senses are reflected in the classifications themselves. It was in Diderot's *Encyclopédie* in the 1750s, that the five 'beaux arts' were established in parallel to the senses, as: architecture, sculpture, painting, music and poetry. Where within such a schema would one place multimedia?

A more common-sensical approach might suggest multimedia in the role of binding together the other arts, and senses. It has long been recognised that there is some organising mechanism at work in what Aristotle called 'common sense'; somehow distributed amongst the other five senses - not a sixth sense as such, but more of an operating system perhaps. In philosophy, one approach to reconciling this dogma was to conclude that the sensory apparatus converged in the brain, and furthermore that mental 'ideas' combined the entirety of experience (Descartes thought this and therefore was, c. 1630). However, this approach, like much multimedia practice and theorising, stops short of providing satisfactory detail on the senses, intellectual or operational apparatuses. Nevertheless, it might be equally reductive to offer a synthesis of sense perception and the organising function in terms of the computer – emanating from the same legacy of an over-reliance on audio-visual codes. If this is where this line of argument seems to be heading, more background is required.

Rée in *I See A Voice* explains that Kant's 'Critical philosophy' managed to resolve some of the established divisions between a 'rationalist' approach (eg. Plato, Liebniz) that broadly argued for knowledge emanating from the intellect and therefore before sensory experience, and a 'empiricist' approach (eg. Aristotle, Locke) that argued for the senses producing knowledge, therefore making universal truth unreliable (and this is what mathematics and computer science is predicated on). Kant aimed to resolve this dilemma in the following manner: 'The intellect can sense nothing, the senses can think nothing; only through their union can knowledge arise' [9]. This does not suggest a relativist compromise but serves to stress that the intellect structures these processes. Or to put it more affirmatively, through Hegel: 'There was nothing in our senses, that had not been in our intellect all along' [10]. If we were to use this as an analogy for generative systems, it might similarly serve to stress the programming procedures that lie behind the raw code that in themselves can sense or think nothing.

Poetry

In the tradition of this line of thinking, Hegel elevated the 'art of sound' to the realm of the spiritual, and concluded that the 'art of speech' was 'total art' – 'the absolute and true art of the spirit' [11]. Despite later criticism against this 'Phonocentrism' as the legitimising voice and source of all meaning and authority (Derrida et al), the limits of traditional aesthetics are emphasised in the problem of defining poetry. Poetry throws sense-bound classificatory distinctions into question as it is both read and heard; or written and spoken/performed. Hegel suggests a way out of this paradox by employing dialectical thinking; as we do not hear speech by simply listening to it. He suggests that we need to represent speech to ourselves in written form in order to grasp what it essentially is. Thus poetry can neither be reduced to audible signs (the time of the ear) nor visible signs (the space of the eye) but is composed of language itself. This synthesis suggests that written and spoken forms work together to form a language that we appreciate as poetry. But does code work in the same way? Is the analogy productive?

Disappointingly, this appears not to be the case with 'Perl Poetry'. Take, for example the 'Best of Show' by Angie Winterbottom from *The Perl Poetry Contest*, and then compare to the original text supplied alongside:

```
if ((light eq dark) && (dark eq light)
   && ($blaze_of_night{moon} == black_hole)
   && ($ravens_wing{bright} == $tin{bright})){
 my $love = $you = $sin{darkness} + 1;
};
```

If light were dark and dark were light
The moon a black hole in the blaze of night
A raven's wing as bright as tin
Then you, my love, would be darker than sin.
[12]

All that has been demonstrated is an act of translation from an existing text, simply 'porting' existing poetry into perl. It produces poetry in a conventional sense, possibly expressing some clever word order and grammatical changes, but does little to articulate the language of perl in itself. When you execute perl poetry in this way, it simply repeats itself but does not acknowledge its execution. It is this operative function that is an essential of part of the experience of poetry.

Poetry at the point of its execution (reading and hearing), produces meaning in multitudinous ways, and can be performed with endless variations of stress, pronunciation, tempo and style. With this in mind, Surrealists and Dadaists used arbitrary patterns, rhythmical noise, and mere chance arrangements of words and sounds – particularly in brutist and simultaneous poems where texts in different languages were read at the same time, and in other automatic or generative experimentation. In this way, they rejected aesthetic conventions of perfection and order, harmony and beauty, and all bourgeois values and taste. From the Dada manifesto of 1918, Tristan Tzara said: 'I am against systems, the most acceptable system is on principle to have none...'. Famously, Tzara advised aspiring poets to cut a newspaper article into words and make a poem by shaking them out of a bag at random, revealing the hidden possibilities of language, and clearly undermining notions of creativity, genius and authority. He explained: 'in these phonetic poems we totally renounce the language that journalism has abused and corrupted' [13]. Thus, the idea of Poetry's universality as well as logic, reason, and aesthetics are brought simultaneously into question. Whereas the automatic text reduced the significance of the poet making the text a transcription or discovery rather than a production or invention, we are keen to stress more purposeful arrangements of code by the programmer.

```
# Extract from walk1/start.pl

my $walk1_beat=0;
my $foo;
sub on_clock {
  return if($foo++ % 4);
  my $beat = $walk1_beat + 1;

  if (($beat-1)%4 eq 0) {
    playnote(7,47+$pitches[$bassctr]-(int($beat/4)*12)) # on-beat
  }
  if (($beat-1)%3 eq 0) {
    playnote(7,35+$pitches[$bassctr]-(int($beat/6)*12)) # syncopate!
  }

  for (0..$#pitches) {
    if (abs($beats[$_]) eq $beat) {
      playnote($_+1,59+$pitches[$_]);
    }
  }

  $bassctr=($bassctr+1)%$#pitches;

  if (rand(50)<25) { $beats[rand(@beats)]++ }
  else { $beats[rand(@beats)]-- }

  if (rand(50)<25) { $pitches[rand(@pitches)]+=$pitches[rand(@pitches)] }
  else { $pitches[rand(@pitches)]-=$pitches[rand(@pitches)] }

  for (0..$#beats)   {   $beats[$_]=wraparound(  $beats[$_],16) }
  for (0..$#pitches) { $pitches[$_]=wraparound($pitches[$_],12) }

  $walk1_beat = ++$walk1_beat % 16;

}
```

Rather than chance arrangements, attention to detail is paramount when it is encountered in written form and in terms of its execution. For instance, significant portions of the code are ‘conditions’ which dictate when the subsequent indented parts are to be executed. In terms of form, any indenting and other visual patterning is a technique to visualise the flow of logic – whereas the same code could be expressed in any shape or arrangement and would run the same output. Some conditions are evaluated inside other conditions to create infinitely complex responses - the indenting programming technique visualises the boolean logic that forms the major core of the code. The language is used in a highly controlled manner and with subtle nuances.

For instance:

```
$walk1_beat = ++$walk1_beat % 16;
```

One might add parenthesis to make this clearer, or not.

```
$walk1_beat++;
if ($walk1_beat eq 16) { $walk1_beat=0 }
```

This executes much the same output as before but through a different operation, and requires specialised knowledge of perl to realise that ‘eq’ is a string comparison operator and not a numeric one. The ‘eq’ and ‘==’ equivalence is a subtle play of language.

Crucial to generative media is that data is actually changed as the code runs. In the example, the ‘++’ and ‘−−‘ symbols are used to increment and decrement numbers - this, in association with the modulo mathematics operator '%' reveals how the numbers are constantly changing. Although these numbers could be calculated by hand and plotted onto something like a musical score, the power of code allows this to happen in ‘real-time’, and the effects are largely unknown until execution. The code could run forever, and it would always be producing new arrangements.

Evidently, code works like poetry in that it plays with structures of language itself, as well as our corresponding perceptions. In this sense, all poetry might be seen to be generative in that it is always in the process of becoming. Even for the Surrealist Paul Valéry, a poem ‘entails a continuous linkage between the voice that is, the voice that impends, and the voice that is to come’ [14]. It is generative in the sense that itunfolds in real-time.

```
# Extract from nuane/start.pl

sub on_clock {
  return if ($foo++ % 4);
  return if (++$beats < $aTime);
  $beats = 0;
  $client->ctrl_send('note', "$aNote, 1, 0") if $aNote;
  $aNote=47+$notes[$ptr];
  $aTime=$times[$ptr];
  $ptr=($ptr+1)%8;
  $client->ctrl_send('note', "$aNote, 1, " . (80 + rand(40)));
}
```

Commands can be executed in a variety of ways. The first two lines of the '`on_clock`' subroutine are '`return`' statements, which prevent the rest of the code from executing if the supplied condition becomes true.

```
return if (++$beats < $aTime);
```

is functionally similar to

```
if (!(++$beat < $aTime)) {
 # …
}
```

In this example, an 'alternative' word order has been chosen. An obvious parallel to poetry can be made in that word order can help to express what is most important in a particular statement - the condition or the action.

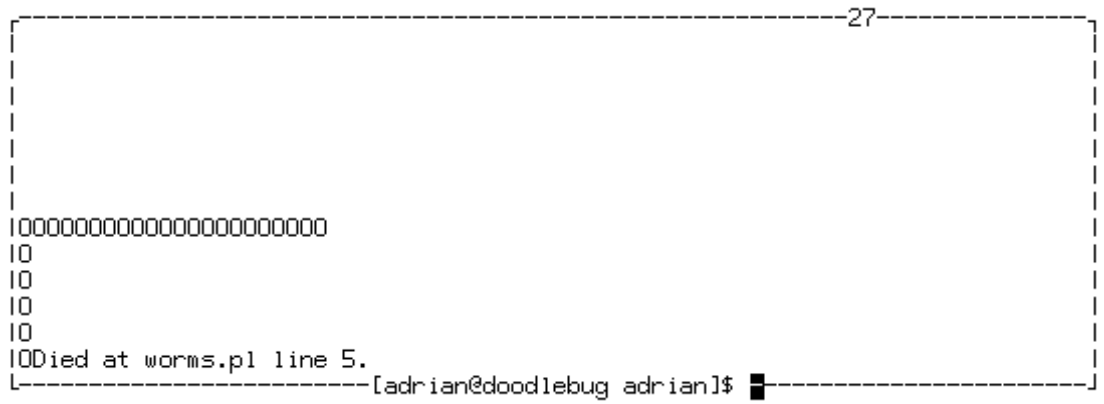
By analogy, generative code has poetic qualities, as it does not operate in a single moment in time and space but as a series of consecutive 'actions' that are repeatable, the outcome of which might be imagined in different contexts. Code is a notation of an internal structure that the computer is executing, expressing ideas, logic, and decisions that operate as an extension of the author's intentions. The written form is merely a computer-readable notation of logic, and is a representation of this process. Yet the written code isn't what the computer really executes, since there are many levels of interpreting and compiling and linking taking place. Code is only really understandable with the context of its overall structure – this is what makes it like a language (be it source code or machine code, or even raw bytes). It may be hard to understand someone else's code but the computer is, after all, multi-lingual. In this sense, understanding someone else's code is very much like listening to poetry in a foreign language - the appreciation goes beyond a mere understanding of the syntax or form of the language used, and as such translation is infamously problematic. Form and function should not be falsely separated.

Poetics

Code itself is clearly not poetry as such, but retains some of its rhythm and metrical form. Code is intricately crafted, and expressed in multitudinous and idiosyncratic ways. Like poetry, the aesthetic value of code lies in its execution, not simply its written form. To appreciate it fully we need to 'see' the code to fully grasp what it is we are experiencing and to build an understanding of the code's actions [15].

```
#!/usr/bin/perl
use Curses;keypad initscr;nodelay 1;box qw{| -};($l,$d,$k,@f)=(1..3,[10,10]);&
n;while(){refresh;@f=([$f[0][0]+$d%2-($d==1)*2,$f[0][1]+$d%2-1+($d==2)*2],@f);
select$f,$f,$f,.06;($c=getch)+1and$d=4-($c%2?2:0)-($c<260);addch@{pop@f},' 'if
@f>$l;$l+=$_=inch@{$f[0]};if(!/ /){/\d/||die;addstr 0, 60,$l;&n}addch@{$f[0]},
'O'}sub n{while(){@v=(rand 24,rand 80);inch(@v)eq' '&&last}addch@v,''.rand 10}
```

This code is extremely dense and difficult to interpret. Certain keywords emerge, but more importantly the code is neatly justified into five lines of equal length. Conditional structures still exist here (see the appearance of { and }) but they are arranged and condensed for visual impact. To appreciate the code fully, you either have to deconstruct the code, as well as use it (or play it as it is a game) [16].

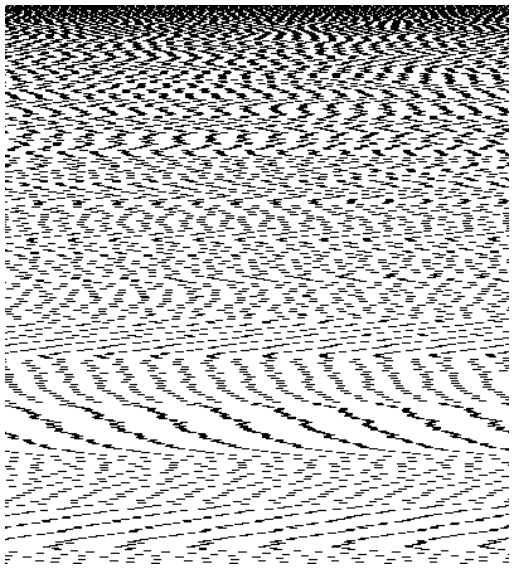


The obfuscation is a side effect to its density as the main intention was to reduce the code to the smallest number of characters possible. In overall terms, it attempts to combine form and function.

This is decidedly not to say that the code should be privileged (as implied by Adorno's comments on music being a by-product of the score) but that the code and the execution of the code need to be experienced in parallel. This is both necessary and impossible for generative or autonomous systems. Any sense of code's autonomy is subject to its place within its operational structure. In this way, code reflects human activity and human activity is coded within social and discursive frameworks - thus authorship is characterised in terms of (social) responsibility to the operating system and language structures [17]. Clearly generative media operates in this way too and appears to encapsulate the paradox of autonomy. Generative art needs to acknowledge the conditions of its own making - its *poesis* (from the Greek *poiesis*, poetic art or creativity from *poiein* - to make). This needs to be made transparent in the spirit of open process, and open source.

```
#!/usr/bin/

$power = 8;
sub fission {
  fork or $child = 1;
  --$power if $child;
  if ($child) {
    exit unless --$power
}
  return $child;
}
while (not &fission) {
  print 0;
  bomb:
  while (&fission) {
    print 1
  }
}
goto 'bomb';
```



[18]

In this example, the program splits in two with every iteration. The code is relatively lengthy as the basic instruction could be reduced to one short line of code:

```
fork while 1;
```

The instruction is simply to ‘split this process in two for ever’ - thus, after the first iteration you get two processes, after the second you get four, then eight, and so on indefinitely. However, the output of the first example is significant in that it is a visualisation of the execution of the process in a more complex performative manner. On a technical level, the computer is under such a high load that it fails to comply to its instructions - after a while the fork calls fail to split the process in two, and. the ordering in which the task scheduler does things becomes less-ordered the harder it is pushed. In this way, the output is a visualisation of the computer’s performance during the program's execution. The output would look very different on different computers, thus providing a ‘watermark’ of the processor and operating system. The code and the resultant actions are intricately linked in poetic dialogue.

To separate the code and the resultant actions would simply limit the aesthetic experience, and ultimately the study of these forms - as a form of criticism - and what in this context might be better called ‘poetics’. Generativecode encapsulates these issues:
‘Its output would be… that is to say [like] poetry correctly defined; Language so well chosen and aptly arranged that, even when expressing tedious or distasteful subjects, it would remain vivid and lively and “pleasing to the ear”.’ [19]

We propose that the production of generative code should be undertaken with similar critical reflection and panache.

}

References

All perl scripts are written by Alex McLean and Adrian Ward of Slub, http://www.slub.org/

[1] Jorge Luis Borges, Foreword to *Obra Poética*, quoted in, Juhani Pallasmaa, *The Eyes of the Skin*, Polemics, London: Academy, 1996, p. 6.

[2] T. F. Hoad, *The Concise Oxford Dictionary of English Etymology*, Oxford: Oxford University Press, 1986, p. 7.

[3] Raymond Williams, *Keywords: a vocabulary of culture and society*, London: Fontana, 1988, p. 31.

[4] Slavoj Zizek, ed., *Mapping Ideology*, London: Verso, 1997.

[5] Georg W. Hegel, *Introductory Lectures on Aesthetics* (1823) trans. B. Bosanquet, London: Penguin, 1993.

[6] For more on the limits of aesthetics, see Andrew Benjamin & Peter Osborne. eds., *Thinking Art: Beyond Traditional Aesthetics*, London: ICA 1991.

[7] One suitably named attempt to try to engage with digital systems beyond mere design issues is Sean Cubitt's *Digital Aesthetics*, London: Sage 1998.

[8] Dominique Laporte, *History of Shit*, London: MIT Press, 2000.

[9] Kant, from 'The History of Pure Reason' in *Critique of Pure Reason* (1781, 1787), quoted in Jonathan Rée, *I See a Voice: a Philosophical History*, London: Flamingo 1999, p. 330.

[10] Hegel, from *Logic* (Encyclopedia), quoted in Rée, *Ibid.,* p. 342. This is not to say that the senses are not crucially important as they structure our interpretations through space and time (what Kant distinguished as 'outer' and 'inner' experience – as spatial and temporal accordingly). Rée proceeds to chart the history of this by pointing to the importance of Husserl's Phenomenology. For more on this, see Rée, 'The Five Senses and the History of Philosophy' in, *Ibid*., pp. 329-345.

[11] Rée, *Ibid*., p. 356.

[12] Kevin Meltzer, 'The Perl Poetry Contest', in *The Perl Journal*, Volume 4, Issue 4, 2000,

http://www.itknowledge.com/tpj/contest-poetry.html

Meltzer explains: 'This short entry, by Angie Winterbottom, was the most interesting. Her style was fresh and unique, and her use of visual representations in the text are clever. Consider the following excerpt:

```
($blaze_of_night{moon} == black_hole)
```

"The moon, a black hole in the blaze of night."

Marvellous! Angie tells us that this entry is from Jim Steinman's song *The Invocation,* on the Pandora's Box album *Original Sin.*'

[13] Tristan Tzara, 'Dada Manifesto' (1918), in Charles Harrison & Paul Wood, *Art in Theory: 1900-1990: an anthology of changing ideas*, Oxford: Blackwell 1998, pp. 249-253.

[14] Rée, quoting Valéry*, Ibid*., p. 361.

[15] The potential for embracing this could be expressed in software development like *MPEG-4 Structured Audio* that specifies sound not as sampled data, but as a computer program that generates audio when run. Computer scientists call this approach 'Kolmogorov' encoding. It combines a powerful language for computing audio (SAOL, pronounced 'sail') and a musical score language (SASL, pronounced 'sassil') with legacy support for the MIDI format. MP4-SA also defines an efficient encoding of these elements into a binary file format suitable for transmission and storage.

John Lazzaro and John Wawrzynek*, MPEG-4 Structured Audio: Developer Tools*, http://www.cs.berkeley.edu/~lazzaro/sa/

[16] It is a full-featured, highly addictive classic arcade game called *Worm*, now being re-discovered by owners of Nokia phones as *Snake*.

[17] For more on this issue, see our previous paper 'The Authorship of Generative Art', GA 1999, http://www.generative.net/

[18] Cover graphic and source from Pimmon's *invalidObject Series* ('while') http://www.fallt.com/invalidObject/while/index.html

[19] Rée, *Op cit.,* p.349, paraphrasing Baumgarten's *Reflections on Poetry*.