Programmer as Artist

Essay for Creative Technologies and Art Practices

Code: IS71018A

Phil Jones

Introduction

When I arrived on the course at Goldsmiths I brought with me an idea. I imagined a spectrum of "artistic software" at one end of which are generic tools: packages like Photoshop, flexible enough to allow an artist to produce almost any image, but inert, heavy, doing nothing until the user starts pushing with the mouse. At the other end of the spectrum are algorithmic art pieces where the programmer defines a system with a set of rules, and the viewer's role is passive : to watch the rules unfold and appreciate the results.

I was, I declared, interested in neither extreme of the spectrum, but in works somewhere in the middle: "sweet spots" where both program and "participant" are expected to be active. Where the participant is "thrown"[1] into the world of the software. We are, I thought, *dynamical systems*, at home in a changing world. And software which is too passive is unnatural and uncomfortable, while software that simply unfolds according to its own rules feels hermetically opaque. If the principle of its evolution is part of what makes a work interesting, then it must be perceptible in some sense. If the algorithm is not legible then at least it can be *tangible* and manipulable.

Hence I wanted to explore the space of pieces which recruit the participant as an active co-creator of the form of the work[2].

I set out to create sketches with this quality, but too-rigid assumptions are soon challenged. When describing ideas for one recent work-in-progress : "Machine Gardens", I found it hard to explain what I was aiming at except "it's a program which helps artists create mechanical systems" which elicited the response "well then, you need to find some real artists to test it with to see if it works for them".

While that could be dismissed as a simple misunderstanding, it had a profound effect on my thinking. My struggle to describe the work, and the response, opened many questions about the role of computers as tools in art, software in art, and what kind of artist the programmer can or might aspire to be.

Does making a program to "help artists" imply that one is not an artist oneself? Must programs be either "tools" (means to an end) or "art works" (ends in themselves)? Can, for example, Kai Krause[3], be an artist?

Another point that has become clearer to me : my spectrum of software outlined above is a natural way for a *programmer* to look at things. Anyone making software must consider how its users will interact with it. And almost every software developer cares that the user's experience is good, which means finding the right balance between competing requirements such as ease of use, flexibility and completeness.

By framing interactive art as the "problem" of finding the right compromise I was thinking like a developer. But that may not be how the visual artist coming to digitality thinks about interaction. A common response when I explain the above is to be asked what is wrong with being at one or other end of the spectrum?[4]

Now artists solve "plastic problems" of finding the right balances of form and colour and material all the time. What, perhaps, made my question strange was that I saw "interaction" itself as a kind

---

1 "Thrown" in the Heideggerian sense which I take from Winograd and Flores. (Winograd, T., and F. Flores. 1986. Understanding Computers and Cognition. Norwood, NJ: Ablex.)

2 I am not saying here that the normal consumption of an art piece is merely passive. Of course there is active engagement : interpretation, looking from different angles, co-creation of meaning, selectivity of when and where the work is viewed, communal experience triggered by the work etc. I do, however, want to say that the active co-creation of *form* in the art I'm talking about does differ in some significant way from all of these.

3 A maker of Photoshop plugins (see http://en.wikipedia.org/wiki/Kai_Krause, retrieved 12 July, 2011)

4 Of course there's nothing wrong with being at either end of the spectrum if that's where you want to be.

of material to be explicitly shaped and worked with. And in this I started to see an answer to the question of what kind of artist a programmer might aspire to be.

The programmer is already, by inclination and training, a system builder. The programmer's material is “system dynamics” and her techniques are ways to describe and manipulate those dynamics.

The rest of this essay tries to throw more light on an *artistic* application of those facts by exploring analogies between programming and other arts. It is divided into two sections :

1) *The Programmer's Material* elaborates further on the kind of materials that the programmer perceives and works with.
2) *System Builders* invokes Clare Bishop's discussions of relational and collaborative arts and considers in the context of programmer art.

The Programmer's Material

What is the Programmer's Material? I'd argue that material is whatever is malleable or *manipulable* by the artist. For the painter, it is paint. For the potter clay. For the programmer it is *code.*

Analogies between code and artistic media are common. Ward Cunningham, the inventor of wiki, makes an explicit simile driven by malleability[5] when discussing his “refactoring”[6] practice :

> *I like the notion of working the program, like an artist works a lump of clay. An artist wants to make a sculpture, but before she makes the sculpture, she just massages the clay. She starts towards making the sculpture, and sees what the clay wants to do. And the more she handles the clay, the more the clay tends to do what she wants ... A development team works on a piece of code over several months. Initially, they make a piece of code, and it's a little stiff. It's small, but it's still stiff. Then they move the code, and it gets a little easier to move.*

To call code a “material” creates some difficulty because many words that we'd like to use to describe or explain become contentious. Isn't material meant to have “physicality”? What are the “plastic” or “formal” properties of code?

Semiotician Kumiko Tanaka-Ishii[7] elaborates. “Computer language” is a system of signs, much like human language. He uses painting and sculpture to illustrate the different varieties of sign : Ito Jakuchu's birds[8] are literal representations, Magritte's bird-shaped hole[9] represents a bird without showing it, Brancusi's Bird in Space[10] abstracts out some general qualities of birds. Tanaka-Ishii finds all three kinds of representation occurring in a single line of code[11]:

```
int x = 32;
```

Here “32” is a literal number; the variable[12] “x” is a symbol that can represent numbers even though it isn't a literal picture of one. And “int”, short for integer or “whole number”, is a *type* i.e. an *abstraction* which represents what is common to all whole numbers.

This is the material that the programmer needs to understand and manipulate : complexes of symbols of different kinds including those which represent generalisations about classes of things. In fact, it should be noted that *inventing abstractions* is at the heart of programmer practice and, as Malcolm McCullough points out, “the history of programming may be understood as largely a matter of increased abstraction”[13] (This throws up an intriguing parallel with the history of painting.

5 Interviewed on Artima : http://www.artima.com/intv/clay3.html, retrieved July 13, 2011
6 Reworking code without changing its behaviour. See Fowler, Martin, 1999, Refactoring : Improving the Design of Existing Code, Addison-Wesley Professional
7 Tanaka-Ishii, Kumiko, 2010, The Semiotics of Programming, Cambridge
8 See, for example, http://www.stolaf.edu/courses/2004sem2/Art/260/rocklin/essay.htm (retrieved July 13, 2011)
9 La Grande Famille, see http://www.serjacopo.com/MagrittePicta/Paint_001.html (retrieved July 13, 2011)
10 See http://en.wikipedia.org/wiki/Bird_in_Space (retrieved July 13, 2011)
11 This example is from The Semiotics of Programming, chapter 6 : The Statement x := x + 1
12 A kind of “box in memory” into which numbers and other data can be put.
13 Abstracting Craft, page 97

Like Mondrian evolving from reporting the tangled branches of trees to ever cleaner lines and colour patches[14], the programmer evolves from individual variables containing each item of data, through classes and general-purpose collections, to higher-order functions and monads[15].)

In his book, "The Craftsman"[16], philosopher Richard Sennett talks about "material consciousness" that the crafter[17] develops. This consciousness comes in three guises : a *metamorphosis* of the material itself, attempts at *marking* the material with some stamp to say "I made this", and an *anthropomorphism* attributing human character to the work.

In code, metamorphosis includes the continual invention of new abstractions in programming languages. Marking by commenting is common. And, despite attempts by some people to exterminate it[18], anthropomorphism is alive and well.

From the growing craftsmanship of the programmer come the new abstractions.

Malcolm McCullough, summarizing the history of symbolic notation[19], draws attention to "generative structure" :

> "[S]oftware design is a matter of defining an appropriate structure for serving a task or problem … Often the best way to do this is through manipulation. Generative structure is the beginnings of a medium largely because it invites manipulation … a structure has a feel based on internal laws and self-regulation, which we experience by working on it – through transformation. This is a fundamental idea behind any understanding of the computer as medium."

He emphasizes structure, and its ally *grammar*, in the generation of new things, including art works. But having done so, he then then reintroduces the key virtue of craft as a balance :

> "Although it is important to respect generative structures we must also acknowledge that their power will be realised through the complementary role of personal sensibility."

Here, for a moment, is a flash of the tradition of craft as humanist expression, the ethics of John Ruskin and William Morris, and rejection of the machine[20]. McCullough, though, is seeking a rapprochement. His book, "Abstracting Craft", starts with the premise that the graphical user interface and the trope of *direct manipulation* of data has reintroduced hand / eye co-ordination in the use of computers and this is therefore an opportunity for the return of craft values of skill and workmanship.

His next few sentences could be a manifesto :

> Theories of design computing demand attention to the skilful nature of accomplished symbolic manipulation. Explorations of generative structure obtain power from hand, eye and tools. They arise from personal knowledge, practice and commitment of the sort found in traditional handicrafts, now applied to symbolic systems.

Sennett shares the sentiment, preferring the Enlightenment to the Romantic view of craftsmanship, "when working with machines rather than fighting was the radical, emancipatory challenge."[21]

---

14 There's a good animated film of this at http://denverartsygal.blogspot.com/2009/04/piet-mondrian-journey-through-modern.html (retrieved July 13, 2011)

15 See http://en.wikipedia.org/wiki/Monad_(functional_programming) (retrieved July 13, 2011)

16 Sennet, Richard, 2008, The Craftsman, Penguin Books. Interestingly Sennett uses Linux programmers as one of his examples of dedicated crafter.

17 I'll normally use the term "crafter" because of the unfortunate gender bias of "craftsman".

18 For example, Djikstra, Edsgar. W., The Fruits of Misunderstanding : http://www.cs.utexas.edu/users/EWD/transcriptions/EWD08xx/EWD854.html (retrieved July 13, 2011)

19 McCullough, Malcolm, 1996, Abstracting Craft : the practiced digital hand, MIT Press. This history of symbols is chapter 4.

20 I have a lot of interest in that side of the story too. But no space to explore it here.

21 William Morris also was not as Luddite as many of his followers. In "Art and its Producers", 1881, he writes "*I do not [believe] we should aim at abolishing all machinery; I would do some things with machinery which are now done by hand, and other things by hand which are now done by machinery; in short, we would be the masters of our machines and not their slaves, as we are now.*" quoted http://pubs.socialistreviewindex.org.uk/sr196/nineham.htm, retrieved June 15, 2011

System Builders

So we have moved from a particular problem : the programmer as artist, to consider the programmer's material : code. Acknowledging that it has a variety of kinds of sign, including abstract representations, we noted that the metamorphosis that was part of Sennett's "material consciousness" included the invention of new abstractions leading to the "generative structures" that McCullough mentions. Generative structures are the material of much algorithmic art but McCullough calls for this to be tempered by human sensibilities and skills. Not just for the sake of individual artworks, but also perhaps for the humanist impulse that runs through the history of craft defined in opposition to industrial manufacture.[22]

McCullough, then, is another seeker of a sweet-spot on a spectrum between algorithmic unfolding and human agency.

We will now take a slight digression. Software is animated first by the machine and secondly by its users. Furthermore good software is typically *developed* incrementally, that is, developers accept feedback from the users to steer each new release. The programmer takes on a new responsibility of managing this dynamic as the software becomes a social object in people's lives. Phil Agre says that computers become a "trading zone" between different people and different disciplines.[23] And the developer becomes orchestrator of their discourse.

This should remind us of some modern "systems thinking" in contemporary art. Clare Bishop offers us a useful overview of the character of Relational Aesthetics[24].

> "The implication is that this work inverses the goals of Greenbergian modernism. Rather than a discrete, portable, autonomous work of art that transcends its context, relational art is entirely beholden to the contingencies of its environment and audience.
>
> Moreover, this audience is envisaged as a community: rather than a one-to-one relationship betweens work of art and viewer, relational art sets up situations in which viewers are not just addressed as a collective, social entity, but are actually given the wherewithal to create a community, however temporary or utopian this may be. "[25]

Relational artists actively recruit the viewers as participants in co-creation of the work. Rirkrit Tiravanija offers gallery visitors the use of a mockup of his apartment for social activities such as communal eating and partying and the work encompasses these activities. There are still constraints : participants can not sell the apartment and move on. Nor, I guess, extensively redecorate. So here is another in-between spot on a formal spectrum between participant freedom and artist determination.)

Bishop then proceeds to critique relational aesthetic analysis as prioritising the *form* of engagement without enquiring deeply into its *content*. What kind of social activity is engendered? Who, specifically, is invited to participate?

> "For Bourriaud, the structure is the subject matter—and in this he is far more formalist than he acknowledges. "

While Bourriaud is reluctant to include overtly technological art in his relational aesthetics, this discussion does transfer across to the domain of interactive computer art. Personally I welcome the attention relational aestheticians pay to hedonistic concerns such as whether an engendered social situation is "comfortable" or "liveable" as similar concerns of comfort obtain for any computer interaction. Comfort is, perhaps, analogous to visual beauty, something that the artist may be wary of, but which in the long tradition of art (and even more in craft) remains an important consideration.

---

22 For more on this history, see The Craftsman, Chapter 3 : Machines

23 Agre, Phil, 2004, Internet Research : for and against, in Mia Consalvo, Nancy Baym, Jeremy Hunsinger, Klaus Bruhn Jensen, John Logie, Monica Murero, and Leslie Regan Shade, eds, *Internet Research Annual*, Volume 1: *Selected Papers from the Association of Internet Researchers Conferences 2000-2002*, New York: Peter Lang, 2004. Retrieved http://polaris.gseis.ucla.edu/pagre/research.html July 13, 2011

24 Bourriaud, Nicolas, 2002, Relational Aesthetics, Les Presse Du Reel

25 Bishop, Clare. 2004, Antagonism and Relational Aesthetics, October

Nevertheless Bishop's complaint seems a valid one if, as she points out, an artist as politically interesting as Santiago Sierra who recruits collaborators through the market and plays with exploitative relations is effectively ignored. A too narrow focus on both generative and interactive dynamics risks becoming obsessed with structure at the cost of other considerations.

Interestingly, in another piece[26], Bishop focuses on the "ethical turn" in art criticism that considers more politically engaged contemporary artists working with communities and groups. Here, it seems, critics have lost sight of *aesthetic* considerations in judging whether an artist is exploiting or otherwise abusing a collaborative community. Of most interest is the implication that there might be a trade-off between ethical correctness and artistic merit. An artist who gives up too much control of a work, to honour the moral imperative of letting the community speak, is also losing the capacity to make interesting art.

The same worry may be carried across into the realm of software art. And might, indeed, be behind the concerns we have noted at the beginning of this essay. Perhaps the problem with accepting "tools" as art-works is that their creator is seen as having already given up too much authorial voice to be considered interesting. And perhaps the reluctance of many digital artists to explore far from the algorithmic end of the spectrum is motivated by fear of this loss.

Similarly, what makes the author of a Photoshop plugin not an artist (or at least, not an interesting one) may be that she intends no reference to anything outside the pixels on the screen. Hers is a mere formal exercise. Fortunately painting has shown that the move via abstraction to the non-figurative does not automatically imply a loss of external meaning and the same is true for the programmer-artist working with abstraction.

Conclusion

I have tried to cover a lot of ground quickly and as result can give only fragmentary suggestions. I could not, in any case, argue that the only role for the programmer-artist is either to engage with the tradition of craft or to explore the zones between complete user freedom and artist-determined algorithmic unfolding. All artists must follow their own way.

But for me, these ideas fit together. I hope McCullough persuades that there is an opportunity for a digital / abstracted craft: one which makes powerful symbolic representations and abstract structures available for manipulation by the individual crafter. A reading of Sennett will reveal that "craft" is a complex of ideas and ideals (including political, ethical and aesthetic). Craft is not merely utilitarian knowhow or decorative tradition (though it is those too).

Bishop warns us that both intoxicating formal innovation or committed ethical projects can nevertheless founder aesthetically. Interesting works will allow artist's authorial voice to be heard over both generative structure and social orchestration.

26 Bishop, Clare, 2006, The Social Turn, Collaboration and its Discontents, ArtForum

Bibliography

Agre, Phil, 2004, Internet Research : for and against, in Mia Consalvo, et al, eds, Internet Research Annual, Volume 1, Peter Lang, 2004.

Bishop, Clare. 2004, Antagonism and Relational Aesthetics, October

Bishop, Clare, 2006, The Social Turn, Collaboration and its Discontents, ArtForum

Bourriaud, Nicolas, 2002, Relational Aesthetics, Les Presse Du Reel

Fowler, Martin, 1999, Refactoring : Improving the Design of Existing Code, Addison-Wesley Professional

McCullough, Malcolm, 1996, Abstracting Craft : the practiced digital hand, MIT Press

Sennet, Richard, 2008, The Craftsman, Penguin Books

Tanaka-Ishii, Kumiko, 2010, The Semiotics of Programming, Cambridge

Winograd, T., and F. Flores. 1986. Understanding Computers and Cognition. Norwood, NJ: Ablex.