Constructing Experience: Art and Digital Literacy

Dr Bill Hart
Lecturer Electronic Media, Tasmanian School of Art, University of Tasmania

It is difficult to think of an aspect of contemporary art production that hasn't been subsumed, augmented or influenced by digital tools and processes. In creative arts education the necessity of attaining a level of digital literacy, alongside visual, verbal and written literacies for creative arts graduates is generally accepted. Yet common definitions of the aspects of digital literacy that describe the variety of strategies that artists adopt in engaging with computer technology do not yet exist.

Before proposing taxonomy to describe strategies of engagement with digital technology, some existing terminologies and their ambiguous contemporary and historical meanings are reviewed to highlight the changing attitudes towards and definitions of digital literacy in the creative arts. Finally it is argued that there are some critical aspects of creative digital literacy that are not well addressed in current educational programs.

A plethora of words have been used to describe digitally augmented creative practices: Software Art, Programmer Art, Digital Art, Computer Art, Computational Art, Processor Art, Game Art, Interactive Art and New Media.

It is interesting to reflect on how the attitudes towards the use of computers by artists has changed over the last two decades; from being a niche specialisation to being widely embedded in contemporary art practice. The use of these terms has also changed over time, occasionally in contradictory ways. For instance philosopher Dominic Lopes, distinguishes computer art from digital art. Lopes argues that digital art is not actually a distinct art form:

> 'Since computers handle information in a common digital code, usually binary code, they are all-purpose representation devices. We use them to make, manipulate, transmit, and display text, music, sound, and images, whether alone or combined in multimedia. Many scholars explore the varied and far reaching implications of this for the established arts. Yet digital stories are still stories, digital images remain images, and digital music is a kind of music. ' (Lopes, 2009, p1.)

Lopez goes onto argue that Computer Art is a new art form that arises out of the concept of computation, but that it is characterised by being interactive.

> ‘An item is a work of computer art just in case (1) it is art, (2) it is run on a computer, (3) it is interactive, and (4) it is interactive because it is run on a computer.’ (Lopes, 2009, p2.)

Moreover the term ‘Computer Art’ is loaded, more usually it has referred to artefacts generated by computer programs in the period between 1960 and 1990. Computer Art generally doesn’t have a good name; it was associated with late Modernism and is considered an overly formalist practice that ‘failed’. It was only when art became ‘digital’ in the 1990s that it became more acceptable to the mainstream.

What Lopes does make clear is that, in considering art and the computer, distinctions can be made between using this technology as an extension or replacement of existing technology, and explorations into a genuinely new territory. However, Lopes’ analysis that interactivity is what characterises this difference, is unconvincing. Consider Harold Cohen’s 30 plus year project in developing his autonomous drawing program AARON.

Cohen a successful UK painter turned in the late 1960s to programming as a way to explore the representational processes of drawing. (McCorduck, 1990.) For over thirty years until the early 2000s Cohen continued to develop the capabilities and sophistication of his software. Lopes discusses the output from Cohen’s software and categorises it as an ‘appreciative art form’ produced by computer software, but no different in the way that it is appreciated than human-produced drawing or painting. I think that Lopes misses the point, the pictures that AARON produces aren’t the artwork, they are a manifestation of the artwork. The real art object is the program AARON and its potential to produce pictures that encapsulates Cohen’s thirty years of reflection and perception on how children (and adults) draw shapes and render colour. AARON produces pictures that are all different but similar, variations on a theme. The broader significance of AARON is that it highlights that originality or creativity is not about representational skill, but the ability to generate difference in kind.

The ontological status of software as art is an interesting topic and more than one commentator has claimed that it appears to be of a similar order and bears a close relations to the *conceptual art* movement of the 1960’s. (Cramer 2002., Reas 2004.) However the purpose of this paper is practical; to consider how to foster creativity with technology, and that this is best approached by characterising the types of creative engagement we have with the computer rather than the theoretical and historical context.

A taxonomy of engagement by artists with computers

Several broad categories can be made based upon the modes through which the artist engages with software in the development of art works:

1. *Artist as software user:* where the artist uses a software package to create the art work, for example a digital image made using Adobe Photoshop. The computer and associated software is a tool for the production of the work; in many cases the software used was developed to simulate a mechanical, chemical or electronic analogue process such as photography, painting, video. The digital process is apparently transparent; there is no inherent digital aesthetic, but there are signatures that the commercial software leaves upon the work. This is the common conception of *digital art*, and began initially with Quantel paintbox systems in the 1980s and gained widespread acceptance with the advent of the affordable personal colour computer. Digital art was allied to many post-modern strategies such as appropriation and bricolage. (Taylor, 2004, p202.)
2. *Artist critically engaged with software culture:* This movement arose during the 1990s in response to the opaqueness of software as the tools for cultural production, and their control by a small number of multinational corporations. This genre, manifested as *net.art* in the 1990s and later more generally as *software art*, attempted to subvert and expose the underlying cultural assumptions of software, raising a level of awareness that the apparent transparency is a lie, that along with the power of process and flexibility software provides, there is restriction and control. Sometimes whimsical, often containing strong elements of political or cultural criticism, these artists would hack existing software systems or applications or develop their own processes but, in general, the reference would be to an existing software system. (Cramer, 2002, p10.) Artists in this genre include Graham Harwood and Mongrel, Dirk Paesmans and Joan Heemskerk from jodi.org.
3. *Artist as programmer collaborator:* Is where artists collaborate with, or direct, programmers; the artist is acting as producer, specifying and communicating an idea to the programmer who would try and interpret it. Commonly based in installation or interactive art, the programmer either creates or customises an existing software environment, but is usually not given creative credit for the work. Artists working in this mode include Geoffrey Shaw, Bill Seaman, Char Davies. This was the dominant mode during the interactivity era from the late 1980s to the late 1990s, particularly amongst older artists, and usually required institutional support. Much of what has been referred to as new media falls into this category.
4. *Artist as programmer:* This category is the most complex. It has several distinct phases; the earliest phase came from technologists producing images with software in the 1960s, and was called computer art. This was followed by the artist/programmer in the 1970s. The work was (of necessity) embedded in algorithmic process, aligned with Modernism through its focus on outcomes rather than process, and criticised for its mechanical coldness and inhumanity.[i] After a period of unpopularity during the 1980s and 90s, the

> artist/programmer has been rehabilitated in recent years under the banner of software art in recent years (somewhat confusing as this term is also used to describe category 2, *Artist critically engaged with software culture*). These artist/programmers often work within a specialised environment such as the open-source environments *PureData (PD)* and *Processing* or the commercial *MAX/MSP* or *Adobe Flash.* Artists working in this mode include Casey Reas and Jared Tarbell. This branch of contemporary Software Art claims a lineage from the Conceptual art movement of the 1960s and 70s rather than from the historically unpopular Computer Art. (Reas 2004.)

These categories of this taxonomy are not absolute; an individual artist in developing an artwork may incorporate several of these modes of engagement with software.

With the almost complete digitisation of print, photography, film, video and sound, the approach of *artist as software user* has become the dominant production mode of contemporary image culture, and qualifies 'digital literacy' in creative arts education.

Given the dominance of these tools, it is culturally important that *Artist critically engaged with software culture* continues their critique, although this critique is almost exclusively political and rarely addresses formal or aesthetic issues. Discussions of Computer Art during the 1980's emphasise the 'dehumanising' impact of the computer on artistic expression and that it impeded free thought and creativity. Interestingly such criticisms are only occasionally levelled at the sophisticated software systems used in contemporary digital art production. (Lopes 2010, pp28-34., Taylor 2004, pp176-184.)

Theorist Lev Manovich concludes in an analysis of the revolution of 'digital media' that

> *'There is no such thing as "digital media." There is only software – as applied to media data (or "content".)*
>
> *To rephrase: for users who can only interact with media content through application software, "digital media" does not have any unique properties by itself. What used to be "properties of a medium" are now operations and affordances defined by software. If you want to escape our prison "prison- house" of software – or at least better understand what media is today – stop downloading Apps created by others. Instead, learn to program – and teach it to your students.'* (Manovich, 2011.)

While in general I would concur with the sentiments of Manovich, he ascribes a purely media centric model to the engagement between artists and computers. The artist creates or captures media, and software is the tool that acts upon that media, to craft the artwork. That the software used contains implicit political, formal and aesthetic bias is a rationale for the artist to take control

of aspects of this process by writing some of their own software. The outcome, the art object, is still media encoded digitally.

However, I would argue that the software itself can be an art object, and that it can be a unique art form without the dependence on the characteristic of interactivity that Lopes postulates. Harold Cohen's AARON is an example, its not that it reduced drawing pictures to a mechanical process, but that it encapsulates an insight into some processes of drawing. Such software, like conceptual art it is probably more aptly described as a meta-art, or to quote Sol LeWitt; 'the idea is a machine that makes the art.' (LeWitt, 1967.)

LeWitt's assertion has been used to justify the connection between the artist programmer and conceptual art; further analogies are made between software as composition, and its execution on particular hardware as a performance. (Bunt, 2011.)

The artist programmer and particularly the artist/system developer of contemporary software art does operate in a substantially different environment to their predecessor the Computer Artist. The contemporary programmer constructs software within an ecosystem of software objects nested upon levels of abstraction away from the physical hardware. Computers are no longer isolated environments, but interconnected and extended by layers of network services. In comparison the artist programmer of the 1960s had to construct everything from scratch, and while contemporary software and generative art can still be constrained by algorithmic formalism, the richness and complexity of the contemporary open source software ecology that the artist can draw upon mean that it does not have to be.

Socially and economically the potential value of artists who are capable of a deep engagement with software as an art form are substantial. They bring to the create arts new tools of expression, and thus to broader society necessary insights in a significant period of technological change and its consequent social impact.

The obstacles to a creative arts education that facilitates an engagement with software as an art form exist at multiple levels. Career counselling and elective structures in secondary schools often force students to made a choice between a 'creative' stream, and a 'science' or technical stream, regardless of their aptitude in interdisciplinary studies. Undergraduate fine art degrees have become streamlined and cost effective and often don't have the resources to offer programming experiences. Other than the rare individuals who choose to study both a fine arts and a computing degree, the closest undergraduate experience for those that want to engage with software as a creative medium of expression is one of the many computer gaming majors that have emerged. Unfortunately few of these degrees engage with the wider cultural context that an awareness of the history and practice of experimental art and technology could provide.

One way forward to foster a deeper educational engagement between art and technology would be for the development of software art extension programs offered to secondary students. In these programs future arts students could explore a creative engagement with digital technology that goes beyond learning to use software packages and outside the science and technology stream, and in so doing develop an understanding of the value of interdisciplinary educational choices.

References

Bunt, Brogan. *LeWitt* 2011 [cited 28 July 2011]. Available from http://www.broganbunt.net/?p=325.

Cramer, Florian, and Ulrike Gabriel. "Software Art." (2001), http://cramer.plaintext.cc:70/essays/software_art_and_writing/software_art_and_writing.pdf.

Cramer, Florian. "Concepts, Notations, Software, Art." In *Run_Me 1.2*. Moscow, 2002.

LeWitt, Sol. "Paragraphs on Conceptual Art." *Art Forum* 5, no. 10 (1967).

Lopes, Dominic. *From the Author's Perspective: A Philosophy of Computer Art,* American Society for Aesthetics Newsletter 29, no. 1 (Spring 2009): 1–3.

Lopes, Dominic. *A Philosophy of Computer Art*. London ; New York: Routledge, 2010.

Manovich, Lev. *There is Only Software* 2011]. Available from http://www.manovich.net/DOCS/Manovich.there_is_only_software.pdf.

McCorduck, Pamela. *Aaron's Code: Meta-Art, Artificial Intelligence and the Work of Harold Cohen*. New York: W H Freeman and Company, 1990.

Reas, Casey. *{Software} Structures*. Whitney Museum 2004 [cited May 24, 2008]. Available from http://artport.whitney.org/commissions/softwarestructures/text.html.

Taylor, Grant D. "The Machine That Made Science Art : The Troubled History of Computer Art 1963-1989." Doctor of Philosophy, University of Western Australia, 2004.