# Towards a Supportive Technological Environment for Digital Art

Greg Turner and Ernest Edmonds
*Creativity and Cognition Studios, UTS, Sydney*
*greg@gregturner.org and ernest@ernestedmonds.com*

## Abstract

*This paper presents the case for extending programming languages to support digital artists engaged in technologically-innovative work. The anticipated result is an "environment for building environments", which will need to satisfy certain technological requirements according to the areas in which digital artists most need creative support. A review of these areas is undertaken, and a proposal is made to capture the specific areas in which digital artists most need technological support.*

## 1. Context

### 1.1. Artists are Technology Innovators

Artists, as the quintessential creative workers, give form to their imaginations. The physical world of artefacts is very different to the conceptual world of the imagination, and artists often find themselves pushing technology forward, creating new artefacts either as part of, or in order to construct, their art. These new artefacts present new ways of using and thinking about other things. In other words, artists are excellent catalysts for invention.

Specifically, digital artists are responsible for many of the most exciting advances in human-computer interaction today, precisely because they are not exclusively technologists, who "are often taken by surprise to find that their world can be looked at in unfamiliar terms" (L. Candy & E. A. Edmonds, 2002, p. 32). The anthropologist Lucy Suchman cites Heidi Tikka's 2002 piece *Mother, Child* as an example of where the most promising developments in interaction are coming from (Suchman, 2002). Stephen Wilson states, "We are at an interesting place in history, in which it is sometimes difficult to distinguish between techno-scientific research and art – a sign that broader integrated views of art and research are developing" (Wilson, 2002, p. 4). In this sense, digital art is most interesting to technologists, including the HCI community, when it is pushing technology forward.

### 1.2. Artists, like Most People, Struggle with Computers

The main barrier in attempting to achieve such an advancement of technology is the lack of understanding, control, and consequent perceived power over current technology. This sense of powerlessness over technology and the onrush of progress is concomitant with the dystopian future vision often advocated by the traditional arts and humanities, of a world governed by technology; the abandonment of the body and thus our human qualities. For example, Hubert Dreyfus expresses his fears about today's Internet, arguing that it "diminishes one's sense of reality and of the meaning in one's life." (Dreyfus, 2001, p. 102). In contrast, technologists remain more optimistic; in control of their destiny. Alan Kay said in 1971, "Don't worry about what anybody else is going to do… The best way to predict the future is to invent it" (*www.smalltalk.org: Alan Kay*, 2003). However, thirty years later, he is more neutral, claiming that "the computer revolution hasn't happened yet" (Steinberg, 2003), that computers have helped scientists and engineers think in entirely new ways, but have had much less impact on the thinking in other fields, blaming the lack of new creative environments in these fields.

Artists have also written books and articles on programming for other artists: John Maeda describes his personal perceptions of the relationship between computation and digital form and introduces the DBN programming language as a result (Maeda, 1999); Golan Levin et al. present artwork, case studies and code by four artists known for their experiments in computational design (Levin, Ward, lia, & meta, 2001). However, these and others are accounts of artists who are also computer scientists, which is to say that they have also had to overcome the current problems with learning to control technology.

Instead, our goal should be to improve the *technology* in order to give artists, and others not primarily concerned with technology, control over it. That means not just the power to use technology, but the power to explore, and hence *create* technology, and to create *with* technology.

Digital artists need to be provided with a creativity support system, an environment in which to create art. In the technologically-uninteresting case, one or more off-the-shelf packages are used, unmodified, as the environment. In more interesting situations, where technological innovation is taking place, the environment is constructed, by a technologist or technologists, to fit the individual needs of the artist (L. Candy & E. A. Edmonds, 2002, pp. 2-35). (It is worth noting that, often, the artist and technologist is the same person, but for endeavours beyond the technical skills of the artist, the environment is constructed in conjunction with a second party.) Such interdisciplinary collaborations have been classified into three levels: the technologist-as-assistant model, the partnership-with-artist-control model, and the full-partnership model (Mamykina, Candy, & Edmonds, 2002); roughly according to the passivity of the technologist's participation. The technologist's role in each of these models is to construct *and participate in* an environment which turns the expressive ideas of the artist into something executable by a computer, and in turn, to make the processes of the computer intelligible to the artist (see Figure 1). The expressivity of the computer comes from both the technologist(s) and the environment they provide. Hence, the *environment*, not the artwork, is the technological creation, so it is the creation of the *environment* which needs to be technologically supported, possibly with a so-called "environment for building environments" (L. Candy & E. Edmonds, 2002).

**Figure 1. Technologist (right) constructing and participating in the artist's (left) creative environment.**

## 2. The Route

The first step towards providing the necessary support for environment creation is to ask: Where are we now? What are artists doing with digital technology now? Where are they requiring new digital technology? The converse question: Where are they struggling with existing digital technology?

### 2.1. Historical Programming

To illustrate why we need to ask these questions, it is useful briefly to consider the development of computer languages to date. There are currently about five generations of languages (Campbell, 2003), starting with the zeroth generation, machine code and assembly language, which are single instructions for the computer's processor. From working with these single instructions, programmers found they were often working with mathematical expressions, so the first generation of languages created (FORTRAN I, Algol, c1954-8) made these common expressions easier to work with. The second generation (COBOL, Lisp, c1958-62) added higher-level constructs such as functions and subroutines. The third generation (Basic, Pascal, c1962-1970) concentrated on ease of programming, and interpretive languages. It is no coincidence that the first digital art appeared at around this time.

In short, each successive generation of programming language has arisen from the need to simplify the computer for the programmer. This is a goal shared with the HCI community, reinterpreted to encourage creativity, exploration and power, rather than to "ease" the completion of a task. (C. Bradley Dilger presents a well-considered analysis of the concept of "ease" and describes the paradoxical notion that a technology which supposedly eases a particular task can actually increase the complexity of, and skills required for, the completion of that task (Dilger, 2003).)

To simplify programming, and computing in general, difficult operations are made more straightforward, and common operations are sped up. Overall, intelligibility is improved and the power of the programmer is increased. Often, and lamentably, the "hood" is closed on the lower-level technologies, never to be reopened, inhibiting even interested users from looking at or changing exactly what is going on, and forcing them to reinvent old technologies or hack and fudge around the existing ones.

So-called "high-level" languages are appropriately named when compared to assembly language, but we need to keep going. This power should now be spread to other creative fields, and not continue to be the sole preserve of the programmer, since programmers are no longer the sole innovators of computer technology. John Maeda wrote "There is no greater need for visual design than rethinking and redesigning programming itself" (Maeda, 2000, p. 406) – and we should expand this

statement to include all forms of digital creativity. The implication is the eventual abandonment of programming languages as we know them altogether, working instead towards generalised creativity support systems.

## 2.2. Building upon What We Know

In finding answers to the questions asked at the beginning of this section, we are identifying the aspects of digital art development which are currently unnecessarily difficult, and the aspects that are tedious and prosaic. These are areas to make quicker and easier in the new generation of creative environments. We would also be able to see where the boundaries of existing practice lie – where the edges are being pushed.

Currently, it is difficult to find answers to these questions. In general, most digital artists do not publicly discuss in-depth the technological processes they underwent in order to create their work – the finished piece is often seen as the only aspect of importance to the outside world. Since the *environment* for creating the piece (rather than the piece itself) is of paramount importance to the identification of technological requirements, the artistic requirements of such environments must be identified before they can be satisfied. To achieve this, it will be useful to examine the general requirements for environments found so far, and to augment these with a new database of specific technological experiences and requests of artists. Information from analysis of the database will allow the specification of which tools and techniques to provide to aid the efficient building of new, customised environments. Such a database would include information about:

**Which existing technologies were used:** This will allow the mapping of current digital art practice according to technological areas, to indicate trends in and relationships between technology uses in the arts.

**What custom technologies were created:** Where common answers are given, this will serve to highlight were existing technology is repeatedly failing artists. Where less common answers are given, this is a guide to the new technological areas which are the foci of artistic endeavours and in which artists require most customisation.

**Which hardware and software products were used:** As well as assisting in the above analyses, individual projects can use this data as a problem-avoiding/solution-finding device, where the project exhibits similarities to works in the database.

**How the artist achieves control of the technology:** The path of the high-level concept to technological manipulation and the nature of any collaboration which took place, either with the technology directly or indirectly through human collaborators. Analysis of the results in this category will help to identify the popularity of techniques that artists employ to impart their expressivity to a computer.

**The technological barriers encountered:** What technological issues or constraints particularly hindered the development of work. Of particular interest are HCI issues that can be addressed by technological or design improvements.

This database may be populated in a number of ways, each with its own advantages. The database should allow for all of these levels of information, as each type can contribute to trend-finding and analysis:

**1.** At the most basic level, the technological features externally apparent in digital artwork can be inferred from the description provided by the artist or others, for example, the use of animated displays or web interaction. Such information could be gained merely from a technologist's review of existing artwork, and some examples are given in section 4. This approach is limiting because it provides no insight into the developmental process, or even the nature of the environment used. However, in a publicly-accessible database, such information does act as a lure for artists to check over their entry and make additions!

**2.** A questionnaire completed by the artist would provide more details, such as the exact technologies used (for example, the development environment and manufacturers of equipment used), the nature of any collaboration, and some idea of the difficulties encountered. This would be useful because of the ability to disseminate the questionnaire and to obtain results of statistical significance, but is limiting in the types of response sought.

**3.** The database could also contain the results of interviews with artists, where open-ended questions could be asked and clarifications sought, and to which the artist could add additional observations.

**4.** Information could also be used from observational studies of artists and technologists during collaborations, as exemplified in the "COSTART" projects of the Creativity & Cognition Studios (L. Candy & E. A. Edmonds, 2002, ; Edmonds & Candy, 2002). These studies contain detailed descriptions of the events which took place during each residency, from the perspectives of artist, technologist, and an observer, to control for subjectivity, and are a rich source of empirical data.

In order to demonstrate the technological usefulness of this approach, the next two sections contain a review of the technological implications of support for a) the generalised features of creativity support systems and b) the specific technological features of digital art which have been identified so far.

# 3. Identifiable Generalised Creativity Support Features

The results of the proposed database will build upon the work already done in the area of creativity support tools, in which some generalised features of good

creativity support systems have been identified. The proposed new work will combine these general guidelines with specific technological requirements in order to specify and create actual tools for building environments.

In earlier research, empirical studies were used to identify some examples of aspects of creative exploration: Breaking with convention, immersion in the activity, holistic view, parallel channels of exploration (Edmonds & Candy, 2002). Ben Shneiderman lists eight specific operations that should "help more people be more creative more of the time": Searching (for knowledge and inspiration), Visualising, Consulting, Thinking, Exploring, Composing, Reviewing, Disseminating (Shneiderman, 2002). In the digital art domain some of these tasks would be highly interrelated (for instance, visualisation, exploration and composition). Michael Terry and Elizabeth D. Mynatt highlight the need for support of Schön's theory of reflection-in-action: near-term experimentation (previews of the results of actions), longer-term variations, and evaluation of actions, each demonstrated in the "Side Views" application (Terry & Mynatt, 2002).

These general requirements map well onto desired technological features of creative environments. Some examples of these follow:

## 3.1. Visualisation

Obviously, all creative environments are dynamic, interactive systems – even if the finished piece is non-dynamic and/or non-interactive. Technologically, this would mean the standard provision of, for example, a double-buffered graphical display, on more than one screen, which is capable of showing the artwork and environmental controls and, if relevant, multi-channel, multi-tracked audio.

The artist should be able to visualise aspects of the development in different ways. For instance, a section of code could be viewed as text, a diagram, a plain- or pseudo-English explanation (with the aid of metadata to describe the semantics of the code) or as an interactive process in which the artist can try inputs and witness the outputs. The same applies for debugging. For example, adding a "watch" to a variable which is a y-coordinate should cause a visual indication of the coordinate to be overlaid on the space to which it applies.

The work of Manfred Mohr, for example, is centrally about handling data that can only be visualised with the aid of a computer, because of its multi-dimensional complexity (Mohr, 2002), see Figure 2.

## 3.2. Exploration and Thinking

Generally, as with any interactive system, the principles of good HCI design (most importantly, user testing) should be followed. Specifically in this context, the results of changes made to the environment should be immediately apparent where possible, either during execution of the artwork or in preview. Code-modification during runtime is a feature of some interpreted programming languages such as Max (Edmonds et al., 2003), and so-called "late-binding" virtual machines such as Squeak Smalltalk (Ingalls, Kaehler, Maloney, Wallace, & Kay, 1997). Manual-override variable modification during runtime should be supported.

Yasunao Tone's work with Creativity and Cognition Studios used Max/MSP as the basis of a very open exploration in which, quite deliberately, nobody new what the outcome might be. In this case exploration into the unknown was at the core of the approach to the art (Edmonds et al., 2003).

## 3.3. Reviewing

As is generally understood (Lieberman & Fry, 1997), the ability to undo/redo and keep/recover/move previous versions of the program/artwork is useful, as would be the ability to artificially slow down or step through a process in order to understand its behaviour more clearly.

## 3.4. Holistic View

The ability to "fold up" complex processes or structures into simple entities allows the artist to take a holistic view. This is a feature of many visual programming languages (Edmonds et al., 2003) and document-centric creativity support tools (for example, the "Document Map" feature of MS Word). Additionally, the functionality (particularly that provided by the technologist) should be expressible in a way that is intelligible to the artist.

## 3.5. Immersion in the Activity

This is not only a physical environmental requirement, but it is also important not to burden the artist with the minutiae of operating the interface or dealing with peripheral tasks. For example, syntax errors should be fixed automatically or semi-automatically, if possible. For example, the pedantic requirement for semicolons at the ends of lines of code should be removed or reduced to the level of automation to avoid annoying interruptions to the user's process. As mentioned in 3.2, the delay of the code-compile-execute cycle should be minimised.

## 3.6. Breaking with Convention

The creative environment should allow the breaking of its own convention – its own modification and extension, which, when taken to the extreme, implies that the environment should be created in (a subset of) itself, and should allow its own manipulation. Such recursion is often seen in Art-Technology collaborations like the one depicted in Figure 1, where the technologist

**Table 1. Depth of interactivity (input)**

| *Type of Input* | *Additional Technological implications* |
|---|---|
| ***None*** *(except potentially the passage of time).* *e.g. Pictures; film; animation; recorded music* | *Few, bearing in mind that the artwork may need to be stopped or restarted.* |
| ***Simple input:*** *the user or physical environment crosses thresholds or makes selections, usually continuous along one dimension only.* *e.g. Switches; Hyperlinks; Interactive DVD; Simple Motion Sensing; specifically* cubeLife *(Everitt, Turner, 2002)* | *The actions of the user, and possibly the actions of the program, can be represented in a flow-chart-style diagram, and be debugged relatively easily. Some discrete input must be derived, using simple signal analysis, from continuous input (for example, sensing a heartbeat).* |
| ***Complex input:*** *the user or physical environment provides input that can take a large number of values, permutations, or dimensions, allowing for greater expressivity.* *e.g. Text; Speech; Gesture; Video; Continuous Biosensors; specifically* Iamascope *(Fels & Mase, 1999)* | *There is a need for signal processing and calibration tools in the environment. For example, fuzzy logic processing. Debugging is more complex – manually-overriding the input should be supported.* |

**Table 2. Depth of interactivity (output)**

| *Type of Output* | *Additional Technological implications* |
|---|---|
| *None* | *None. Although conceptual artists may disagree, we presume that all art has an output of some sort.* |
| ***Simple output:*** *The art outputs only one result, or one of a discrete set of results (the intricacy of these results is irrelevant for this case).* *e.g. Static art; interactive DVD; linearly-animated art; most "predictable behaviour" art (see Table 3).* | *The provision of tools for pre-calculating, collecting and enumerating these states, if there are a large number of them, would be useful. The operation of the artwork might also be described with flow-charts, or state-event modelling.* |
| ***Complex output:*** *The art outputs any of an extremely large set of states.* *e.g. Virtual environments; generative art* | *Mathematical modelling, AI and, signal processing tools are all potentially useful.* |

places himself in the creative environment which he has constructed for the artist. In the software world, this is less common, although several compilers and tools for programming are written in the very languages they support.

The environment should also be *general*. In other words, every problem that is solvable by computer should have some solution in the environment (Campbell, 2003).

A number of examples of such behaviour have been seen to be partly supported by interpreted visual programming methods (Edmonds et al., 2003). The immediacy of feedback and the holistic view provided seem to be significant factors.

# 4. Identifiable Creativity Support Features for Digital Art

The authors examined digital artworks created in previous studies (L. Candy & E. A. Edmonds, 2002) and published descriptions of works, e.g. the review by Stephen Wilson (Wilson, 2002). By considering the *external* attributes of these works, it was possible to begin to map out the specific technological requirements of the environments which gave rise to them, which augment the list in the previous section. These external attributes fall into two general categories – the way digital artworks are presented, and their apparent behaviour. (The purpose here is not to "classify" art, nor to present an exhaustive review, but to show how considering various aspects and dimensions of art can help describe future technological requirements.)

## 4.1. Presentation

Digital art has brought with it artworks which communicate, or are perceived, in two directions. Communication takes place on a number of levels, from the excitation of photons or electrons (the level at which hardware and software operates) to the 'obvious', concrete object (the level at which both artists and technologists can communicate), to the 'subtle' language of the work (the level at which artists conceptualise).

The technological *communications* requirements of *completed artworks* are straightforward to identify by observing them. To support the most common paradigm, **onscreen drawing** of 2D output and **mouse-and-keyboard input,** would be the trivial case. Common extensions to simple visual media are more complex forms such as **video and 3D**. Of these, 3D output is particularly difficult to satisfy convincingly, because it often imposes a "look" on the image as a result of the use of conventional 3D rendering techniques. Ideally, artists would be able to modify the rendering engine to customise the depiction of 3D objects.

**Table 3. Complexity of change**

| *Level of Complexity* | *Additional Technological implications* |
|---|---|
| ***Predictable:*** *the artist, and familiarised audience, is able to predict what should happen in response to certain events. e.g. Interactive DVD* | *Bug-finding is relatively easy. Predictable art normally has discrete input and output, or no input.* |
| ***Exploratory:*** *the art produces interesting responses to stimuli, which may be entirely deterministic and comprehensible, but the affordance of exploration, or new comprehension by the artist or audience is imperative. e.g. Manfred Mohr (Mohr, 2002), (see Figure 2); Iamascope (Fels & Mase, 1999)* | *Bug-fixing is more complex, as many permutations of complex inputs and outputs all need to be evaluated. Aids to testing and process-control displays are paramount.* |
| ***Emergent-Algorithmic:*** *The computer makes unpredictable changes as a result of internal calculations or random decisions. e.g. Complex Systems; Artificial Life; specifically* Olivine Trees *(Miranda, 2001)* | *Emergent art calls for extensive experimentation and debugging before it is exhibited. Additionally, emergent art is the result of complex systems, so tools to inspect, override, simulate or replay the complex interactions would be useful. The science of complex systems is in its infancy so such tools are not widely available.* |
| ***Emergent-Environmental:*** *The computer makes changes to its behaviour as a result of unpredictable factors outside of its control, namely humans and natural entities (i.e. the weather) e.g. AI and Expert Systems* | |

**Saved and printed output** should also be provided; a difficulty arises if there is a need for high-resolution output. This means that the information for a vector-based description of the onscreen image should either be maintained in real-time or calculable off-line, perhaps by a high-resolution recreation of the actions which formed the display.

**Audio output**, via multi-channel, multi-track waveform or MIDI events should be supported. However, real-time digital audio processing (as with digital video processing) is both an extremely complex and extremely wide-ranging technology, and is likely to require the ability to communicate with third-party software for anything more than simple (or created-from-scratch) applications.

Almost all specialised hardware devices communicate with the **serial, or USB protocol**. The environment should ease the interpretation of the messages which such devices use to communicate, for example by aiding the mapping of high-level commands to sequences of serial data (and vice-versa) and providing visualisation tools to examine these sequences at a higher level (for example, video).

A growing number of artworks (and hardware devices) exhibit some form of **communication over a network**. Given the limitations of the Internet Protocol (IP), these artworks most often communicate in UDP (change-centric data; not all data must arrive at its destination) and TCP (event-centric data; all data is important). Again, aiding the mapping of these messages to higher-level constructs is useful. In addition, some artworks may require distribution (one-to-many, or many-to-many) or storage of messages via a central server, rather than just communication with one other computer (one-to-one). Tools for communicating with a computer *not* under the control of an artist (for example, to get the weather) would also be beneficial.

Last but not least, the communication which a computer has with **files** is important for storing information ready to be used at a later time. The three most ubiquitous types are text files, binary files (difficult to work with for non-technologists) and SQL databases (again, an understanding of SQL is currently necessary). Fast previews and overviews of such files, again where possible, would aid their design and creation by the artist.

## 4.2. Behaviour

Stroud Cornock and Ernest Edmonds classified art according to behaviour in 1973 (Cornock & Edmonds, 1973). The categories given were Static, Dynamic-Passive, Dynamic-Interactive and Dynamic-Interactive (varying). In the light of the present examination of the digital art climate, we can reconfigure these categories in order to elicit further technological requirements. Three axes of behaviour become apparent: depth of interactivity for both input and output, and complexity of change. A single artwork may exhibit one or more characteristics in each axis.

The **depth of the interactivity** is independent for the input (see Table 1) and output (see Table 2) media for any given artwork, and consists of three levels. Each level, for both input and output, has certain technological implications for a supportive environment, beyond those which may arise from other requirements.

The **complexity of change** in behaviour of an artwork is an informal ranking of how predictable the artwork is (see Table 3). This is a subjective classification, as it depends on the level at which the artwork is analysed (on a state-event level, every operation of a computer is inherently predictable. Conversely, it is often easier to execute the instruction than to predict its result). The classification here

illustrates the types of exploration and debugging activities at the human level.

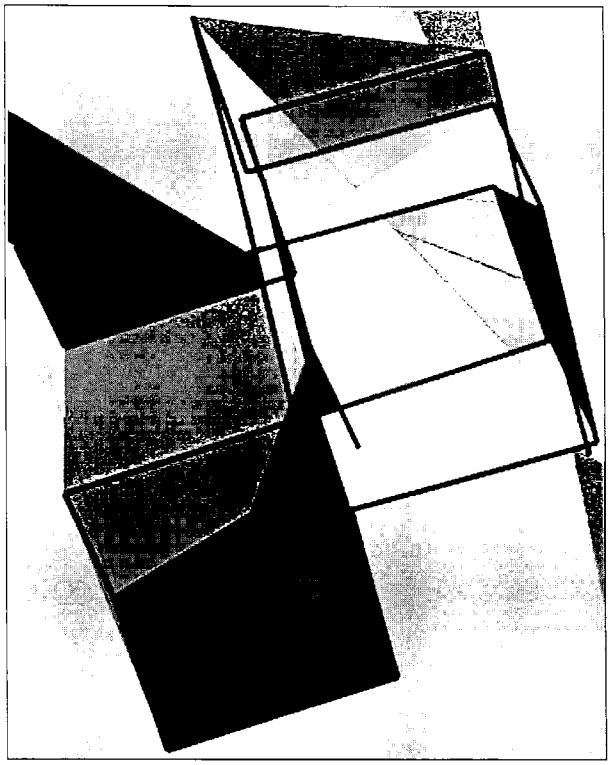

**Figure 2. P-701/B, Manfred Mohr, enduraChrome/canvas, 1999 (from http://research.it.uts.edu.au/creative/ccrs/gallery/mmohr/mmohr.htm)**

## 5. Conclusion

The externally-apparent requirements of digital art and the general requirements for creativity support map closely onto technological requirements for a supportive environment for digital art. Digital artworks have further, distinctive and unique technological requirements not obvious from an external observer's perspective: types of behaviour and calculation not directly related to output (and simple input); the technologies used by the environment; and other, new-technology aspects. The proposed database will assist greatly in identifying these further requirements, and thus can be used to actually construct an "environment for building environments".

(As a further consideration, technical quality should not be ignored in meeting the requirements for such an environment. If recognised, open standards and protocols are used for depiction, communication and interaction within the environment, the flexibility, extensibility, and hence longevity of the artworks will benefit.)

Existing programming languages are all examples of creative environments – someone with knowledge, skills and perseverance can create an environment which does exactly what is required. The problem is that programming languages are hardly any good for anyone without these skills. As Alan Kay remarked, "I thought we would be way beyond where we are now... The irony is that today it looks pretty good. The result of our [Xerox PARC] work is techniques for doing software in an interesting and more powerful way. That was back in the seventies. People today aren't doing a lot of work to move programming to its next phase" (Steinberg, 2003). The issue for the HCI community in particular is to build exploratory, rather than task-oriented environments, since it is exploration which most encourages creativity.

It is time to broaden the focus of creating new digital technology from new computer tools towards new creativity support systems, and from the programmer to the generalised digital creator. In the time to come, the computer should be moulded by the ideas of anyone who wishes to create within its possibilities, much as wood or clay has for millennia. It will become a creative medium in the truest sense. Digital artists are currently the closest fit to the ubiquitous digital creators of the future, and it is their work which is responsible for much of the innovation in interaction technology today.

## 6. Acknowledgements

The authors would like to thank the OZCHI reviewers, Dave Everitt, Keir Smith and Alastair Weakley for their valuable perspectives and insights on the issues presented in this paper.

## 7. References

Campbell, J. (2003). *Lessons in Object-Oriented Programming*, from http://www.cs.qub.ac.uk/~J.Campbell/myweb/oop/oophtml/node4.html

Candy, L., & Edmonds, E. (2002). Interaction in Art and Technology. *Crossings: eJournal of Art and Technology (http://crossings.tcd.ie/), 2*(1).

Candy, L., & Edmonds, E. A. (2002). *Explorations in Art and Technology*: Springer.

Cornock, S., & Edmonds, E. (1973). The Creative Process Where The Artist Is Amplified Or Superseded By The Computer. *Leonardo*(6), 11-16.

Dilger, C. B. (2003). *Ease in Composition Studies*. PhD Dissertation, University of Florida, Florida.

Dreyfus, H. L. (2001). *On The Internet: Thinking in Action*. New York: Routledge.

Edmonds, E., & Candy, L. (2002). Creativity, Art Practice and Knowledge. *Communications of the ACM, 45*(10), 91-95.

Edmonds, E., Candy, L., Fell, M., Knott, R., Pauletto, S., & Weakley, A. (2003). *Developing Interactive Art Using Visual Programming*. Paper presented at the Proceedings of HCI International 2003, Crete.

Everitt, D. (Artist), G. Turner (Collaborator). (2002). *cubeLife: An Internet-Resident Artwork* [Interactive Art]; http://www.cubelife.org.

Fels, S., & Mase, K. (1999). Iamascope: A Graphical Musical Instrument. *Computers and Graphics, 2*(23), 277-286.

Ingalls, D., Kachler, T., Maloney, J., Wallace, S., & Kay, A. (1997). *Back to the Future: The Story of Squeak, A Practical Smalltalk Written in Itself.* Paper presented at the OOPSLA'97 Conference, Atlanta, Georgia.

Levin, G., Ward, A., lia, & meta. (2001). *4x4 Generative Design: Beyond Photoshop*: Friends Of Ed.

Lieberman, H., & Fry, C. (1997). ZStep 95, A Reversible, Animated Source Code Stepper. In J. Stasko, J. Domingue, M. Brown & B. Price (Eds.), *Software Visualization: Programming as a Multimedia Experience*. Cambridge, MA,: MIT Press.

Maeda, J. (1999). *Design By Numbers*: MIT Press.

Maeda, J. (2000). *Maeda@Media*: Thames & Hudson.

Mamykina, L., Candy, L., & Edmonds, E. (2002). Collaborative Creativity. *Communications of the ACM, 45*(10), 96-99.

Miranda, E. R. (2001). On the Origins and Evolution of Music in Virtual Worlds. In D. W. Corne & P. J. Bentley (Eds.), *Creative Evolutionary Systems*: Morgan Kaufmann.

Mohr, M. (2002). Generative Art. In L. Candy & E. A. Edmonds (Eds.), *Explorations in Art and Technology*: Springer.

Shneiderman, B. (2002). Creativity Support Tools: Establishing a framework of activities for creative work. *Communications of the ACM, 45*(10), 116-120.

Steinberg, D. H. (2003). *Daddy, Are We There Yet? A Discussion with Alan Kay*, from http://www.openp2p.com/pub/a/p2p/2003/04/03/alan_kay.html

Suchman, L. (2002). Replicants and Irreductions: Affective encounters at the interface. *European Association for the Study of Science and Technology (EASST)*.

Terry, M., & Mynatt, E. D. (2002). *Recognizing Creative Needs in User Interface Design.* Paper presented at the Creativity and Cognition Conference 2002, Loughborough, UK.

Wilson, S. (2002). *Information Arts: Intersections of Art, Science and Technology*. Cambridge, Massachusetts: The MIT Press.

*www.smalltalk.org: Alan Kay*. (2003). from http://www.smalltalk.org/alankay.html

*In Viller & Wyeth (Eds.) Proceedings of OzCHI2003: New directions in interaction, information environments, media and technology. 26-28 November 2003, Brisbane, Australia: Information Environments Program, University of Queensland.* 

# Towards a Supportive Technological Environment for Digital Art

Greg Turner and Ernest Edmonds
*Creativity and Cognition Studios, UTS, Sydney*
*greg@gregturner.org and ernest@ernestedmonds.com*

## Abstract

*This paper presents the case for extending programming languages to support digital artists engaged in technologically-innovative work. The anticipated result is an "environment for building environments", which will need to satisfy certain technological requirements according to the areas in which digital artists most need creative support. A review of these areas is undertaken, and a proposal is made to capture the specific areas in which digital artists most need technological support.*

## 1. Context

### 1.1. Artists are Technology Innovators

Artists, as the quintessential creative workers, give form to their imaginations. The physical world of artefacts is very different to the conceptual world of the imagination, and artists often find themselves pushing technology forward, creating new artefacts either as part of, or in order to construct, their art. These new artefacts present new ways of using and thinking about other things. In other words, artists are excellent catalysts for invention.

Specifically, digital artists are responsible for many of the most exciting advances in human-computer interaction today, precisely because they are not exclusively technologists, who "are often taken by surprise to find that their world can be looked at in unfamiliar terms" (L. Candy & E. A. Edmonds, 2002, p. 32). The anthropologist Lucy Suchman cites Heidi Tikka's 2002 piece *Mother, Child* as an example of where the most promising developments in interaction are coming from (Suchman, 2002). Stephen Wilson states, "We are at an interesting place in history, in which it is sometimes difficult to distinguish between techno-scientific research and art – a sign that broader integrated views of art and research are developing" (Wilson, 2002, p. 4). In this sense, digital art is most interesting to technologists, including the HCI community, when it is pushing technology forward.

### 1.2. Artists, like Most People, Struggle with Computers

The main barrier in attempting to achieve such an advancement of technology is the lack of understanding, control, and consequent perceived power over current technology. This sense of powerlessness over technology and the onrush of progress is concomitant with the dystopian future vision often advocated by the traditional arts and humanities, of a world governed by technology; the abandonment of the body and thus our human qualities. For example, Hubert Dreyfus expresses his fears about today's Internet, arguing that it "diminishes one's sense of reality and of the meaning in one's life." (Dreyfus, 2001, p. 102). In contrast, technologists remain more optimistic; in control of their destiny. Alan Kay said in 1971, "Don't worry about what anybody else is going to do… The best way to predict the future is to invent it" (*www.smalltalk.org: Alan Kay*, 2003). However, thirty years later, he is more neutral, claiming that "the computer revolution hasn't happened yet" (Steinberg, 2003), that computers have helped scientists and engineers think in entirely new ways, but have had much less impact on the thinking in other fields, blaming the lack of new creative environments in these fields.

Artists have also written books and articles on programming for other artists: John Maeda describes his personal perceptions of the relationship between computation and digital form and introduces the DBN programming language as a result (Maeda, 1999); Golan Levin et al. present artwork, case studies and code by four artists known for their experiments in computational design (Levin, Ward, lia, & meta, 2001). However, these and others are accounts of artists who are also computer scientists, which is to say that they have also had to overcome the current problems with learning to control technology.

Instead, our goal should be to improve the *technology* in order to give artists, and others not primarily concerned with technology, control over it. That means not just the power to use technology, but the power to explore, and hence *create* technology, and to create *with* technology.

Digital artists need to be provided with a creativity support system, an environment in which to create art. In the technologically-uninteresting case, one or more off-the-shelf packages are used, unmodified, as the environment. In more interesting situations, where technological innovation is taking place, the environment is constructed, by a technologist or technologists, to fit the individual needs of the artist (L. Candy & E. A. Edmonds, 2002, pp. 2-35). (It is worth noting that, often, the artist and technologist is the same person, but for endeavours beyond the technical skills of the artist, the environment is constructed in conjunction with a second party.) Such interdisciplinary collaborations have been classified into three levels: the technologist-as-assistant model, the partnership-with-artist-control model, and the full-partnership model (Mamykina, Candy, & Edmonds, 2002); roughly according to the passivity of the technologist's participation. The technologist's role in each of these models is to construct *and participate in* an environment which turns the expressive ideas of the artist into something executable by a computer, and in turn, to make the processes of the computer intelligible to the artist (see Figure 1). The expressivity of the computer comes from both the technologist(s) and the environment they provide. Hence, the *environment*, not the artwork, is the technological creation, so it is the creation of the *environment* which needs to be technologically supported, possibly with a so-called "environment for building environments" (L. Candy & E. Edmonds, 2002).

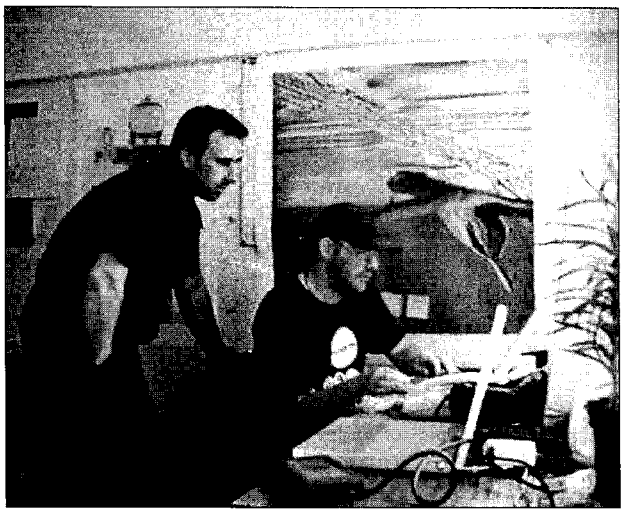

**Figure 1. Technologist (right) constructing and participating in the artist's (left) creative environment.**

## 2. The Route

The first step towards providing the necessary support for environment creation is to ask: Where are we now? What are artists doing with digital technology now? Where are they requiring new digital technology? The converse question: Where are they struggling with existing digital technology?

### 2.1. Historical Programming

To illustrate why we need to ask these questions, it is useful briefly to consider the development of computer languages to date. There are currently about five generations of languages (Campbell, 2003), starting with the zeroth generation, machine code and assembly language, which are single instructions for the computer's processor. From working with these single instructions, programmers found they were often working with mathematical expressions, so the first generation of languages created (FORTRAN I, Algol, c1954-8) made these common expressions easier to work with. The second generation (COBOL, Lisp, c1958-62) added higher-level constructs such as functions and subroutines. The third generation (Basic, Pascal, c1962-1970) concentrated on ease of programming, and interpretive languages. It is no coincidence that the first digital art appeared at around this time.

In short, each successive generation of programming language has arisen from the need to simplify the computer for the programmer. This is a goal shared with the HCI community, reinterpreted to encourage creativity, exploration and power, rather than to "ease" the completion of a task. (C. Bradley Dilger presents a well-considered analysis of the concept of "ease" and describes the paradoxical notion that a technology which supposedly eases a particular task can actually increase the complexity of, and skills required for, the completion of that task (Dilger, 2003).)

To simplify programming, and computing in general, difficult operations are made more straightforward, and common operations are sped up. Overall, intelligibility is improved and the power of the programmer is increased. Often, and lamentably, the "hood" is closed on the lower-level technologies, never to be reopened, inhibiting even interested users from looking at or changing exactly what is going on, and forcing them to reinvent old technologies or hack and fudge around the existing ones.

So-called "high-level" languages are appropriately named when compared to assembly language, but we need to keep going. This power should now be spread to other creative fields, and not continue to be the sole preserve of the programmer, since programmers are no longer the sole innovators of computer technology. John Maeda wrote "There is no greater need for visual design than rethinking and redesigning programming itself" (Maeda, 2000, p. 406) – and we should expand this

statement to include all forms of digital creativity. The implication is the eventual abandonment of programming languages as we know them altogether, working instead towards generalised creativity support systems.

## 2.2. Building upon What We Know

In finding answers to the questions asked at the beginning of this section, we are identifying the aspects of digital art development which are currently unnecessarily difficult, and the aspects that are tedious and prosaic. These are areas to make quicker and easier in the new generation of creative environments. We would also be able to see where the boundaries of existing practice lie – where the edges are being pushed.

Currently, it is difficult to find answers to these questions. In general, most digital artists do not publicly discuss in-depth the technological processes they underwent in order to create their work – the finished piece is often seen as the only aspect of importance to the outside world. Since the *environment* for creating the piece (rather than the piece itself) is of paramount importance to the identification of technological requirements, the artistic requirements of such environments must be identified before they can be satisfied. To achieve this, it will be useful to examine the general requirements for environments found so far, and to augment these with a new database of specific technological experiences and requests of artists. Information from analysis of the database will allow the specification of which tools and techniques to provide to aid the efficient building of new, customised environments. Such a database would include information about:

**Which existing technologies were used:** This will allow the mapping of current digital art practice according to technological areas, to indicate trends in and relationships between technology uses in the arts.

**What custom technologies were created:** Where common answers are given, this will serve to highlight were existing technology is repeatedly failing artists. Where less common answers are given, this is a guide to the new technological areas which are the foci of artistic endeavours and in which artists require most customisation.

**Which hardware and software products were used:** As well as assisting in the above analyses, individual projects can use this data as a problem-avoiding/solution-finding device, where the project exhibits similarities to works in the database.

**How the artist achieves control of the technology:** The path of the high-level concept to technological manipulation and the nature of any collaboration which took place, either with the technology directly or indirectly through human collaborators. Analysis of the results in this category will help to identify the popularity of techniques that artists employ to impart their expressivity to a computer.

**The technological barriers encountered:** What technological issues or constraints particularly hindered the development of work. Of particular interest are HCI issues that can be addressed by technological or design improvements.

This database may be populated in a number of ways, each with its own advantages. The database should allow for all of these levels of information, as each type can contribute to trend-finding and analysis:

**1.** At the most basic level, the technological features externally apparent in digital artwork can be inferred from the description provided by the artist or others, for example, the use of animated displays or web interaction. Such information could be gained merely from a technologist's review of existing artwork, and some examples are given in section 4. This approach is limiting because it provides no insight into the developmental process, or even the nature of the environment used. However, in a publicly-accessible database, such information does act as a lure for artists to check over their entry and make additions!

**2.** A questionnaire completed by the artist would provide more details, such as the exact technologies used (for example, the development environment and manufacturers of equipment used), the nature of any collaboration, and some idea of the difficulties encountered. This would be useful because of the ability to disseminate the questionnaire and to obtain results of statistical significance, but is limiting in the types of response sought.

**3.** The database could also contain the results of interviews with artists, where open-ended questions could be asked and clarifications sought, and to which the artist could add additional observations.

**4.** Information could also be used from observational studies of artists and technologists during collaborations, as exemplified in the "COSTART" projects of the Creativity & Cognition Studios (L. Candy & E. A. Edmonds, 2002, ; Edmonds & Candy, 2002). These studies contain detailed descriptions of the events which took place during each residency, from the perspectives of artist, technologist, and an observer, to control for subjectivity, and are a rich source of empirical data.

In order to demonstrate the technological usefulness of this approach, the next two sections contain a review of the technological implications of support for a) the generalised features of creativity support systems and b) the specific technological features of digital art which have been identified so far.

# 3. Identifiable Generalised Creativity Support Features

The results of the proposed database will build upon the work already done in the area of creativity support tools, in which some generalised features of good

creativity support systems have been identified. The proposed new work will combine these general guidelines with specific technological requirements in order to specify and create actual tools for building environments.

In earlier research, empirical studies were used to identify some examples of aspects of creative exploration: Breaking with convention, immersion in the activity, holistic view, parallel channels of exploration (Edmonds & Candy, 2002). Ben Shneiderman lists eight specific operations that should "help more people be more creative more of the time": Searching (for knowledge and inspiration), Visualising, Consulting, Thinking, Exploring, Composing, Reviewing, Disseminating (Shneiderman, 2002). In the digital art domain some of these tasks would be highly interrelated (for instance, visualisation, exploration and composition). Michael Terry and Elizabeth D. Mynatt highlight the need for support of Schön's theory of reflection-in-action: near-term experimentation (previews of the results of actions), longer-term variations, and evaluation of actions, each demonstrated in the "Side Views" application (Terry & Mynatt, 2002).

These general requirements map well onto desired technological features of creative environments. Some examples of these follow:

## 3.1. Visualisation

Obviously, all creative environments are dynamic, interactive systems – even if the finished piece is non-dynamic and/or non-interactive. Technologically, this would mean the standard provision of, for example, a double-buffered graphical display, on more than one screen, which is capable of showing the artwork and environmental controls and, if relevant, multi-channel, multi-tracked audio.

The artist should be able to visualise aspects of the development in different ways. For instance, a section of code could be viewed as text, a diagram, a plain- or pseudo-English explanation (with the aid of metadata to describe the semantics of the code) or as an interactive process in which the artist can try inputs and witness the outputs. The same applies for debugging. For example, adding a "watch" to a variable which is a y-coordinate should cause a visual indication of the coordinate to be overlaid on the space to which it applies.

The work of Manfred Mohr, for example, is centrally about handling data that can only be visualised with the aid of a computer, because of its multi-dimensional complexity (Mohr, 2002), see Figure 2.

## 3.2. Exploration and Thinking

Generally, as with any interactive system, the principles of good HCI design (most importantly, user testing) should be followed. Specifically in this context, the results of changes made to the environment should be immediately apparent where possible, either during execution of the artwork or in preview. Code-modification during runtime is a feature of some interpreted programming languages such as Max (Edmonds et al., 2003), and so-called "late-binding" virtual machines such as Squeak Smalltalk (Ingalls, Kaehler, Maloney, Wallace, & Kay, 1997). Manual-override variable modification during runtime should be supported.

Yasunao Tone's work with Creativity and Cognition Studios used Max/MSP as the basis of a very open exploration in which, quite deliberately, nobody new what the outcome might be. In this case exploration into the unknown was at the core of the approach to the art (Edmonds et al., 2003).

## 3.3. Reviewing

As is generally understood (Lieberman & Fry, 1997), the ability to undo/redo and keep/recover/move previous versions of the program/artwork is useful, as would be the ability to artificially slow down or step through a process in order to understand its behaviour more clearly.

## 3.4. Holistic View

The ability to "fold up" complex processes or structures into simple entities allows the artist to take a holistic view. This is a feature of many visual programming languages (Edmonds et al., 2003) and document-centric creativity support tools (for example, the "Document Map" feature of MS Word). Additionally, the functionality (particularly that provided by the technologist) should be expressible in a way that is intelligible to the artist.

## 3.5. Immersion in the Activity

This is not only a physical environmental requirement, but it is also important not to burden the artist with the minutiae of operating the interface or dealing with peripheral tasks. For example, syntax errors should be fixed automatically or semi-automatically, if possible. For example, the pedantic requirement for semicolons at the ends of lines of code should be removed or reduced to the level of automation to avoid annoying interruptions to the user's process. As mentioned in 3.2, the delay of the code-compile-execute cycle should be minimised.

## 3.6. Breaking with Convention

The creative environment should allow the breaking of its own convention – its own modification and extension, which, when taken to the extreme, implies that the environment should be created in (a subset of) itself, and should allow its own manipulation. Such recursion is often seen in Art-Technology collaborations like the one depicted in Figure 1, where the technologist

**Table 1. Depth of interactivity (input)**

| *Type of Input* | *Additional Technological implications* |
|---|---|
| ***None*** *(except potentially the passage of time).* *e.g. Pictures; film; animation; recorded music* | *Few, bearing in mind that the artwork may need to be stopped or restarted.* |
| ***Simple input:*** *the user or physical environment crosses thresholds or makes selections, usually continuous along one dimension only.* *e.g. Switches; Hyperlinks; Interactive DVD; Simple Motion Sensing; specifically* cubeLife *(Everitt, Turner, 2002)* | *The actions of the user, and possibly the actions of the program, can be represented in a flow-chart-style diagram, and be debugged relatively easily. Some discrete input must be derived, using simple signal analysis, from continuous input (for example, sensing a heartbeat).* |
| ***Complex input:*** *the user or physical environment provides input that can take a large number of values, permutations, or dimensions, allowing for greater expressivity.* *e.g. Text; Speech; Gesture; Video; Continuous Biosensors; specifically* lamascope *(Fels & Mase, 1999)* | *There is a need for signal processing and calibration tools in the environment. For example, fuzzy logic processing. Debugging is more complex – manually-overriding the input should be supported.* |

**Table 2. Depth of interactivity (output)**

| *Type of Output* | *Additional Technological implications* |
|---|---|
| ***None*** | *None. Although conceptual artists may disagree, we presume that all art has an output of some sort.* |
| ***Simple output:*** *The art outputs only one result, or one of a discrete set of results (the intricacy of these results is irrelevant for this case).* *e.g. Static art; interactive DVD; linearly-animated art; most "predictable behaviour" art (see Table 3).* | *The provision of tools for pre-calculating, collecting and enumerating these states, if there are a large number of them, would be useful. The operation of the artwork might also be described with flow-charts, or state-event modelling.* |
| ***Complex output:*** *The art outputs any of an extremely large set of states.* *e.g. Virtual environments; generative art* | *Mathematical modelling, AI and, signal processing tools are all potentially useful.* |

places himself in the creative environment which he has constructed for the artist. In the software world, this is less common, although several compilers and tools for programming are written in the very languages they support.

The environment should also be *general*. In other words, every problem that is solvable by computer should have some solution in the environment (Campbell, 2003).

A number of examples of such behaviour have been seen to be partly supported by interpreted visual programming methods (Edmonds et al., 2003). The immediacy of feedback and the holistic view provided seem to be significant factors.

# 4. Identifiable Creativity Support Features for Digital Art

The authors examined digital artworks created in previous studies (L. Candy & E. A. Edmonds, 2002) and published descriptions of works, e.g. the review by Stephen Wilson (Wilson, 2002). By considering the *external* attributes of these works, it was possible to begin to map out the specific technological requirements of the environments which gave rise to them, which augment the list in the previous section. These external attributes fall into two general categories – the way digital artworks are presented, and their apparent behaviour. (The purpose here is not to "classify" art, nor to present an exhaustive review, but to show how considering various aspects and dimensions of art can help describe future technological requirements.)

## 4.1. Presentation

Digital art has brought with it artworks which communicate, or are perceived, in two directions. Communication takes place on a number of levels, from the excitation of photons or electrons (the level at which hardware and software operates) to the 'obvious', concrete object (the level at which both artists and technologists can communicate), to the 'subtle' language of the work (the level at which artists conceptualise).

The technological *communications* requirements of *completed artworks* are straightforward to identify by observing them. To support the most common paradigm, **onscreen drawing** of 2D output and **mouse-and-keyboard input,** would be the trivial case. Common extensions to simple visual media are more complex forms such as **video and 3D**. Of these, 3D output is particularly difficult to satisfy convincingly, because it often imposes a "look" on the image as a result of the use of conventional 3D rendering techniques. Ideally, artists would be able to modify the rendering engine to customise the depiction of 3D objects.

**Table 3. Complexity of change**

| *Level of Complexity* | *Additional Technological implications* |
|---|---|
| ***Predictable:*** *the artist, and familiarised audience, is able to predict what should happen in response to certain events. e.g. Interactive DVD* | *Bug-finding is relatively easy. Predictable art normally has discrete input and output, or no input.* |
| ***Exploratory:*** *the art produces interesting responses to stimuli, which may be entirely deterministic and comprehensible, but the affordance of exploration, or new comprehension by the artist or audience is imperative. e.g. Manfred Mohr (Mohr, 2002), (see Figure 2); Iamascope (Fels & Mase, 1999)* | *Bug-fixing is more complex, as many permutations of complex inputs and outputs all need to be evaluated. Aids to testing and process-control displays are paramount.* |
| ***Emergent-Algorithmic:*** *The computer makes unpredictable changes as a result of internal calculations or random decisions. e.g. Complex Systems; Artificial Life; specifically* Olivine Trees *(Miranda, 2001)* | *Emergent art calls for extensive experimentation and debugging before it is exhibited. Additionally, emergent art is the result of complex systems, so tools to inspect, override, simulate or replay the complex interactions would be useful. The science of complex systems is in its infancy so such tools are not widely available.* |
| ***Emergent-Environmental:*** *The computer makes changes to its behaviour as a result of unpredictable factors outside of its control, namely humans and natural entities (i.e. the weather) e.g. AI and Expert Systems* | |

**Saved and printed output** should also be provided; a difficulty arises if there is a need for high-resolution output. This means that the information for a vector-based description of the onscreen image should either be maintained in real-time or calculable off-line, perhaps by a high-resolution recreation of the actions which formed the display.

**Audio output**, via multi-channel, multi-track waveform or MIDI events should be supported. However, real-time digital audio processing (as with digital video processing) is both an extremely complex and extremely wide-ranging technology, and is likely to require the ability to communicate with third-party software for anything more than simple (or created-from-scratch) applications.

Almost all specialised hardware devices communicate with the **serial, or USB protocol**. The environment should ease the interpretation of the messages which such devices use to communicate, for example by aiding the mapping of high-level commands to sequences of serial data (and vice-versa) and providing visualisation tools to examine these sequences at a higher level (for example, video).

A growing number of artworks (and hardware devices) exhibit some form of **communication over a network**. Given the limitations of the Internet Protocol (IP), these artworks most often communicate in UDP (change-centric data; not all data must arrive at its destination) and TCP (event-centric data; all data is important). Again, aiding the mapping of these messages to higher-level constructs is useful. In addition, some artworks may require distribution (one-to-many, or many-to-many) or storage of messages via a central server, rather than just communication with one other computer (one-to-one). Tools for communicating with a computer *not* under the control of an artist (for example, to get the weather) would also be beneficial.

Last but not least, the communication which a computer has with **files** is important for storing information ready to be used at a later time. The three most ubiquitous types are text files, binary files (difficult to work with for non-technologists) and SQL databases (again, an understanding of SQL is currently necessary). Fast previews and overviews of such files, again where possible, would aid their design and creation by the artist.

## 4.2. Behaviour

Stroud Cornock and Ernest Edmonds classified art according to behaviour in 1973 (Cornock & Edmonds, 1973). The categories given were Static, Dynamic-Passive, Dynamic-Interactive and Dynamic-Interactive (varying). In the light of the present examination of the digital art climate, we can reconfigure these categories in order to elicit further technological requirements. Three axes of behaviour become apparent: depth of interactivity for both input and output, and complexity of change. A single artwork may exhibit one or more characteristics in each axis.

The **depth of the interactivity** is independent for the input (see Table 1) and output (see Table 2) media for any given artwork, and consists of three levels. Each level, for both input and output, has certain technological implications for a supportive environment, beyond those which may arise from other requirements.

The **complexity of change** in behaviour of an artwork is an informal ranking of how predictable the artwork is (see Table 3). This is a subjective classification, as it depends on the level at which the artwork is analysed (on a state-event level, every operation of a computer is inherently predictable. Conversely, it is often easier to execute the instruction than to predict its result). The classification here

illustrates the types of exploration and debugging activities at the human level.

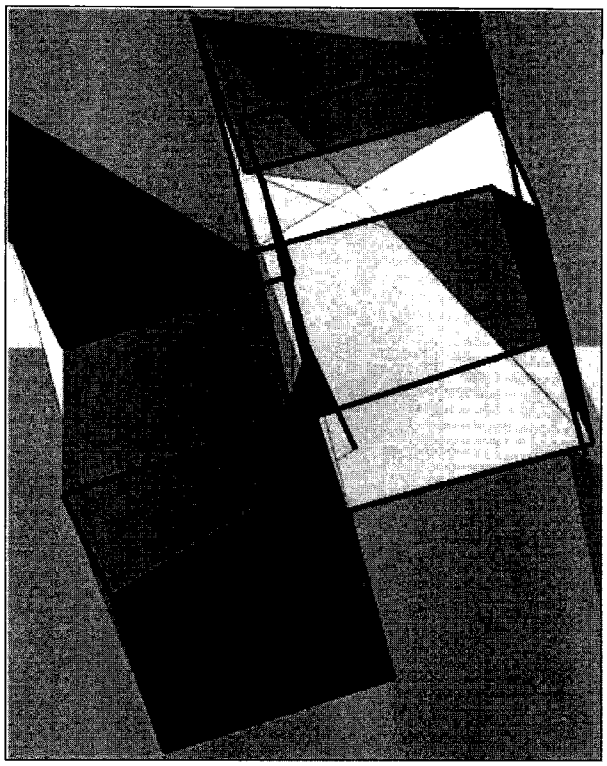

**Figure 2. P-701/B, Manfred Mohr, enduraChrome/canvas, 1999 (from http://research.it.uts.edu.au/creative/ccrs/gallery/mmohr/mmohr.htm)**

## 5. Conclusion

The externally-apparent requirements of digital art and the general requirements for creativity support map closely onto technological requirements for a supportive environment for digital art. Digital artworks have further, distinctive and unique technological requirements not obvious from an external observer's perspective: types of behaviour and calculation not directly related to output (and simple input); the technologies used by the environment; and other, new-technology aspects. The proposed database will assist greatly in identifying these further requirements, and thus can be used to actually construct an "environment for building environments".

(As a further consideration, technical quality should not be ignored in meeting the requirements for such an environment. If recognised, open standards and protocols are used for depiction, communication and interaction within the environment, the flexibility, extensibility, and hence longevity of the artworks will benefit.)

Existing programming languages are all examples of creative environments – someone with knowledge, skills and perseverance can create an environment which does exactly what is required. The problem is that programming languages are hardly any good for anyone without these skills. As Alan Kay remarked, "I thought we would be way beyond where we are now... The irony is that today it looks pretty good. The result of our [Xerox PARC] work is techniques for doing software in an interesting and more powerful way. That was back in the seventies. People today aren't doing a lot of work to move programming to its next phase" (Steinberg, 2003). The issue for the HCI community in particular is to build exploratory, rather than task-oriented environments, since it is exploration which most encourages creativity.

It is time to broaden the focus of creating new digital technology from new computer tools towards new creativity support systems, and from the programmer to the generalised digital creator. In the time to come, the computer should be moulded by the ideas of anyone who wishes to create within its possibilities, much as wood or clay has for millennia. It will become a creative medium in the truest sense. Digital artists are currently the closest fit to the ubiquitous digital creators of the future, and it is their work which is responsible for much of the innovation in interaction technology today.

## 6. Acknowledgements

The authors would like to thank the OZCHI reviewers, Dave Everitt, Keir Smith and Alastair Weakley for their valuable perspectives and insights on the issues presented in this paper.

## 7. References

Campbell, J. (2003). *Lessons in Object-Oriented Programming*, from http://www.cs.qub.ac.uk/~J.Campbell/myweb/oop/oophtml/node4.html

Candy, L., & Edmonds, E. (2002). Interaction in Art and Technology. *Crossings: eJournal of Art and Technology (http://crossings.tcd.ie/), 2*(1).

Candy, L., & Edmonds, E. A. (2002). *Explorations in Art and Technology*: Springer.

Cornock, S., & Edmonds, E. (1973). The Creative Process Where The Artist Is Amplified Or Superseded By The Computer. *Leonardo*(6), 11-16.

Dilger, C. B. (2003). *Ease in Composition Studies.* PhD Dissertation, University of Florida, Florida.

Dreyfus, H. L. (2001). *On The Internet: Thinking in Action.* New York: Routledge.

Edmonds, E., & Candy, L. (2002). Creativity, Art Practice and Knowledge. *Communications of the ACM, 45*(10), 91-95.

Edmonds, E., Candy, L., Fell, M., Knott, R., Pauletto, S., & Weakley, A. (2003). *Developing Interactive Art Using Visual Programming.* Paper presented at the Proceedings of HCI International 2003, Crete.

Everitt, D. (Artist), G. Turner (Collaborator). (2002). *cubeLife: An Internet-Resident Artwork* [Interactive Art]; http://www.cubelife.org.

Fels, S., & Mase, K. (1999). Iamascope: A Graphical Musical Instrument. *Computers and Graphics, 2*(23), 277-286.

Ingalls, D., Kaehler, T., Maloney, J., Wallace, S., & Kay, A. (1997). *Back to the Future: The Story of Squeak, A Practical Smalltalk Written in Itself.* Paper presented at the OOPSLA'97 Conference, Atlanta, Georgia.

Levin, G., Ward, A., lia, & meta. (2001). *4x4 Generative Design: Beyond Photoshop*: Friends Of Ed.

Lieberman, H., & Fry, C. (1997). ZStep 95, A Reversible, Animated Source Code Stepper. In J. Stasko, J. Domingue, M. Brown & B. Price (Eds.), *Software Visualization: Programming as a Multimedia Experience*. Cambridge, MA,: MIT Press.

Maeda, J. (1999). *Design By Numbers*: MIT Press.

Maeda, J. (2000). *Maeda@Media*: Thames & Hudson.

Mamykina, L., Candy, L., & Edmonds, E. (2002). Collaborative Creativity. *Communications of the ACM, 45*(10), 96-99.

Miranda, E. R. (2001). On the Origins and Evolution of Music in Virtual Worlds. In D. W. Corne & P. J. Bentley (Eds.), *Creative Evolutionary Systems*: Morgan Kaufmann.

Mohr, M. (2002). Generative Art. In L. Candy & E. A. Edmonds (Eds.), *Explorations in Art and Technology*: Springer.

Shneiderman, B. (2002). Creativity Support Tools: Establishing a framework of activities for creative work. *Communications of the ACM, 45*(10), 116-120.

Steinberg, D. H. (2003). *Daddy, Are We There Yet? A Discussion with Alan Kay*, from http://www.openp2p.com/pub/a/p2p/2003/04/03/alan_kay.html

Suchman, L. (2002). Replicants and Irreductions: Affective encounters at the interface. *European Association for the Study of Science and Technology (EASST)*.

Terry, M., & Mynatt, E. D. (2002). *Recognizing Greative Needs in User Interface Design.* Paper presented at the Creativity and Cognition Conference 2002, Loughborough, UK.

Wilson, S. (2002). *Information Arts: Intersections of Art, Science and Technology*. Cambridge, Massachusetts: The MIT Press.

*www.smalltalk.org: Alan Kay*. (2003). from http://www.smalltalk.org/alankay.html