ResearchGate

See discussions, stats, and author profiles for this publication at: https://www.researchgate.net/publication/2488204

Aesthetic Computing: Making Artistic Mathematics & Software

Article · January 2002

Source: CiteSeer

CITATIONS
8

READS
203

1 author:

Paul A. Fishwick
University of Texas at Dallas
345 PUBLICATIONS **5,387** CITATIONS

SEE PROFILE

Some of the authors of this publication are also working on these related projects:

The Profession of Modeling and Simulation View project

Second China Project View project

All content following this page was uploaded by Paul A. Fishwick on 04 November 2014.

The user has requested enhancement of the downloaded file.

Aesthetic Computing: Making Artistic Mathematics & Software

Paul Fishwick, University of Florida
http://www.cise.ufl.edu/~fishwick

Introduction

Computer art represents the use of the digital computer to create artistic products, by employing technology in the service of art. Its clearest societal presence is at the movie theatre, where computer graphics directly support the art of movie production. The bridge between computing and art also has other connotations. Creating software has often been termed an art, but by this term, most are referring to its craft-like qualities; it is the process that is likened to art, and not so much the product. However, Knuth has developed a significant agenda in his goals to elucidate a new sort of *software aesthetic*—a return to a form of software calligraphy [1]. That is, to create software is to participate in a kind of craft, an art form. However, most software has little in the way of its external appearances with art—there are few connections between the usual products of art found in art galleries or on the web and the typical programming language. Most programming is fairly minimal in representation. This minimalism is inherited from mathematics, which preceded it.

Before continuing on the subject of software, and its notation, we should recognize that the practice of computing is about several stages such as 1) designing a program, 2) providing input to a program, and finally 3) executing the program. Moreover, a program can be formalized as a mathematical expression such as a lambda function or production system, and so ultimately, the problem of representing programs regresses into one of mathematical notation. Each of the previous three program stages has the potential for artistic influence. The area of execution is likely the most successful in terms of artistic

influence, since early on, mathematics has demonstrated its facility for its graphical "output" with the Cartesian coordinate system, and many fascinating color projections as evidenced by the area of nonlinear dynamics and chaos theory. The Mandelbrot set is a good example, where if the equation generating the set is viewed as a program, the program's output can be visualized in many different ways, such as Fig. 1.

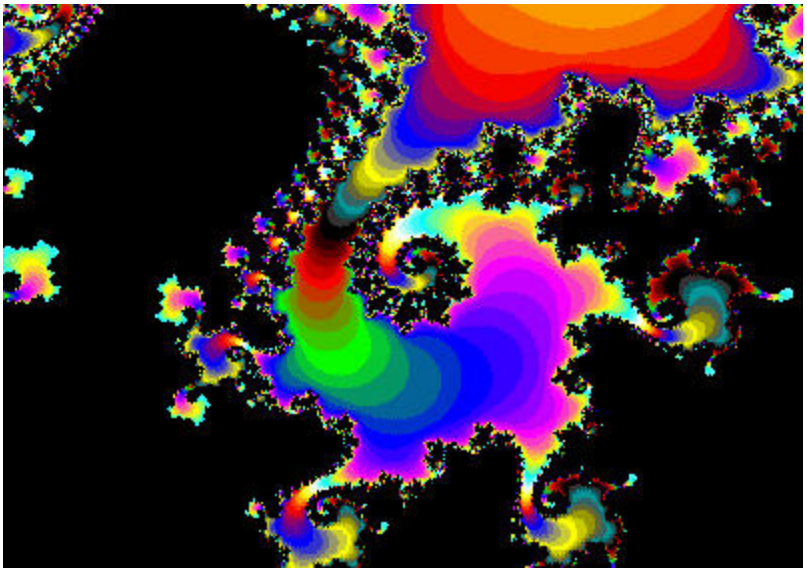


Figure 1: Mandelbrot set segment created using Thorsen's Java applet http://www.thorsen.priv.no/services/mandelbrot (X = -.1714806 and Y= -0.6508634)

Areas within scientific visualization, and more recently information visualization [2] and software visualization [3] all contribute to bringing art into the computer and into the software which executes in the black box. With regard to dynamic systems in general, with software being one type of system, visualization also has a significant presence [4].

Based on the past research within computer science and systems science and engineering, it seems that all is well, and that we are on the road to a stronger bridge which has historically separated art and science. Generally speaking, this is seems to be true; however, if we survey the landscape of aesthetics and styles afforded by art, we still find much of current day notations for both mathematics and software to be highly stylized and diagrammatic. For example, one may represent individual components and programs as primitives such as lines, circles, and

rectangles or as Platonic-like solids such as spheres, pipes and cubes. Can we go further in our representations, and can we personalize mathematics, and its computer science progeny to achieve a far richer aesthetic landscape?

Aesthetic Computing

Mathematical Beginnings

Aesthetic Computing is the study of artistic forms in the design of formal structures found in computing. The word "aesthetic" comes from the Greek *aisthEtikos*, which means "sense perception." Our use of the word *aesthetic* is centered on pluralism, eclecticism, and hybridism, which is to say that *aesthetic computing* refers to the potential use of a large *variety* of possible styles. Therefore, aesthetic computing is defined with this variegated potential in mind. We can reverse the words "computer and "art" and aesthetic computing becomes synonymous with "art computer" [5]. This area suggests the hypothesis that our notations for formal structures are largely dictated by the economics of labor; we notate in ways that are efficient given the current state of technology. At first, this may seem obvious, but it has dramatic consequences for envisaging new formal representations. Tokens, as seen in Figure 2, and frequently found in clay envelopes in Babylonia, suggest that language of which mathematics is one kind, gradually evolved from small tokens and figurines into flatter, stylized text [6]. These tokens were used primarily for accounting, and expressed both mathematical quantity as well as the quality of sign-based representation. This represented our first non-oral language. For example, the token at the top right represents one sheep. A concomitant

Figure 2: Clay tokens from Susa, Iran 3300BC Courtesy Musee du Louvre, Department des Antiquites Orientales, Paris.

stylization of Chinese characters also demonstrates this evolutionary tendency away from certain artistic products toward more stylized, economically-motivated forms. To some extent, we sacrifice art, or at least its vast potential in form and style, for the sake of technology. This technology has delivered many fruits of knowledge and engineering, and so we are generally optimistic and cognizant of these trends toward changes in representation. Without these changes, we would not have computers, nor would we have computer art. We must recognize, though, that as technology evolves and the economy of labor shifts, so must our representations. If I am able to create a virtual sculpture in the same amount of time that I can type **Y = X + 2**, this is bound to have magnificent consequences on the potential for remaking certain formal notations in new artistic ways, and with additional dimensions such as sound, narrative, and tactile sensation.

Let's start with a mathematical model and then progress to examples of dynamic models of systems and software. Consider the Mandelbrot set previously pictured in Fig. 1. From a systems view, we are viewing the output of the equations used to generate and define the set. But the equations have remained in a black box of sorts. Yes, we might take the equations and express them with typographically and calligraphically aesthetic forms, but we might also use our newly acquired levels of technology to remake and remediate the equations. Equations, like other mathematical and computer forms, are expressed in ways convenient to our limitations in model representation. The equation for the Mandelbrot set is $\mathbf{z = z^2 + c}$. It represents a difference equation since **z** on the left side is the new value of **z**, and **z** on the right side is the old value. Difference equations rely upon iteration for their solution, by taking the new **z** and putting it back into the old **z** to, again, calculate another, newer, **z**. Variables **z** and **c** are in the complex plane and so can be manipulated as if they were **(x,y)** pairs, with the proper transformation. We begin with a value for **c** and then start **z** at zero, while continuing to iterate: 1) $\mathbf{z = 0 + c = c}$; 2) $\mathbf{z = c^2+c}$ and so forth. A pixel is colored black if **z** *does not* tend toward infinity (i.e., it is part of the set), and another color if it does. Different rates of divergence result in a spectrum of colors. If we shine our light inside the "equation box,"

and use the pluralistic basis of aesthetic computing, we might instead remake the original equation using Figs 3 through 5.

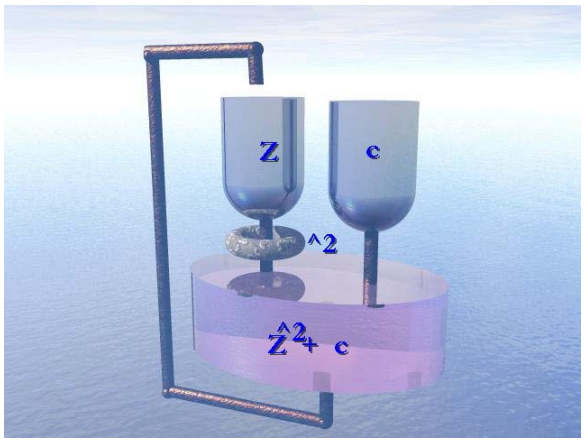


Figure 3: Mandelbrot Set machine

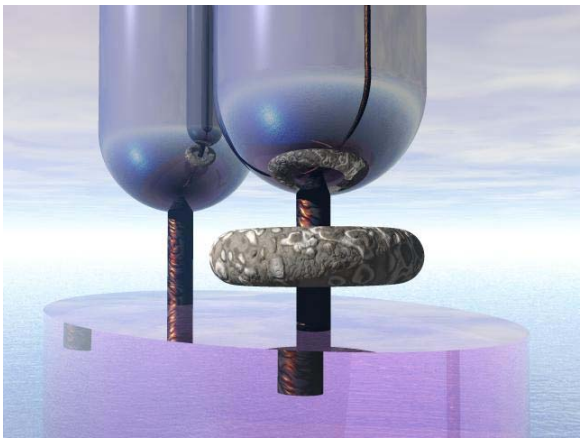


Figure 4: View # 2 of Fig. 3

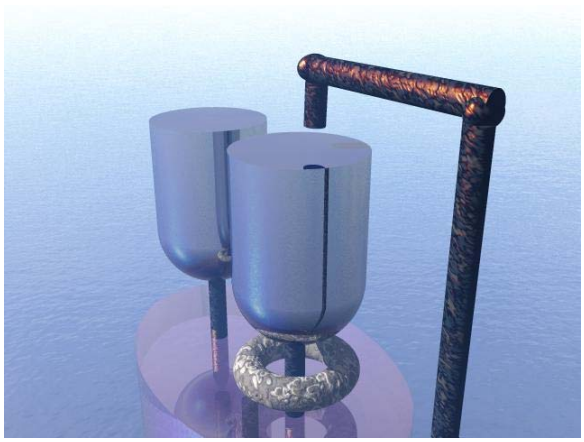


Figure 5: View #3 of Fig. 3

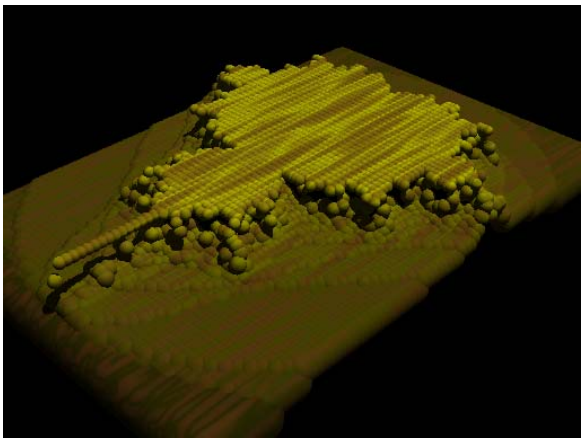


Figure 6: Resulting 3D Mandelbrot Set

Figure 3 shows a virtual "Mandelbrot machine" which has an isomorphic mapping to the original equation. Inside this figure, we have placed the equivalent variables, with the notation **^2** referring to the operation "to square." Using a processing plant metaphor, quantities are placed into containers and the sum total is collected in the elliptical tub, whose resultant is then pumped back up into the original container for **z**. Figures 4 and 5 show separate views of Figure 3. Figure 6 illustrates a monochromatic output from the machine.

New Focus and Goals

The goals of aesthetic computing build upon those of visualization as currently practiced in computer science, with a greater concentration on introducing *aesthetic variety* to traditional visualization projects. Cultural issues can predominate since, despite the evidence of an increasing rate of technology capability in creating virtual objects (through computer graphics) and physical objects (through rapid prototyping machines), computer scientists have been educated on more traditional fare, with textual notation having made its mark early in life. This is not to say that that computer science is averse to the idea of pluralistic aesthetics, but only that the field has grown upon a solid bedrock of mathematical notation. So, progressing toward greater freedom of expression does present a significant cultural challenge. The same goes for artists, on the other side of the coin, in that many artists are unfamiliar with formal model structures, even though they are well schooled in aesthetics. At the University of Florida, we have developed undergraduate and graduate degree programs in Digital Art and Science (DAS) and a Digital Worlds Institute [7] to help nurture students with a healthy blend of art and computer science.

Abstraction

Abstraction has been heralded as the hallmark of mathematics and computer science, but it is important to isolate abstraction for its fundamental properties, which appear to often be confused with the material which goes into making signs. The concept of abstraction can surface as a reason not to employ more extravagant forms. However, abstraction cannot completely come to the rescue because the essence of abstraction is not so much a reduction in material used to represent a model, but rather in the use of one object to capture a subset of attributes for another; the actual structure of an object, minimal or otherwise, is unrelated to the object's abstraction role in modeling. The letters **z** and **c** in Figure 3 help us to connect the equation to the corresponding 3D objects, but they become excess baggage when considered in the larger view of what it means to be an equation. As

with any language, meaning lies not in the object but in the relation among objects. Thus, the virtual machine in Fig. 3 is equivalent, under isomorphism, to the equation text, and we may think of the vessels in Fig. 3, designating variable names, as containing a fluid whose height, say, captures variable values. There is nothing about the very idea of "equation" which suggests that it must be represented by dried chemical inks and dead plant matter. A key difference in aesthetic computing, which differentiates it from prior efforts, is its focus on personalized, cultural representation rather than on standardized form. While standards may be created based on group aesthetics and convention, model design is construed to be a bottom-up activity that begins with a large diversity of styles and evolves not unlike its biological brethren. Usability, critical analysis as achieved through intermediary organizations (like a consumer's union for models), perceived needs, and market-driven demand will cull those designs and styles not meant to prosper in the large. The aesthetic computing field represents a call to a more diverse application of aesthetics for formal structure.

Virtual Analog

By selectively integrating artistic forms into mathematics and computing, in many ways we return to the past. The work in visualization is a return to a time when numerical information was visual and tactile, and a return to more calligraphically efficient, and expressive, forms of text realization. Knuth's TeX, for example, represents the return to gaining greater degrees of control over the text-based aesthetic as it was known and practiced before the introduction of typography and the printed book. This "boomerang effect" also shows itself in Fig 3 since prior to the electronic digital computer used today, we constructed analog computers whose components were far more visual and tactile then they are today. Fig. 3 is a type of *virtual analog* technology, which borrows from the past but is remade for the future in a more efficient form.

Computing

What does all this have to do with software and computing? By beginning with mathematical notation and structure, we are in a position to extend these stylistic expressions to software [8], whose formal realization is mathematical. Creating software is a natural extension of creating mathematics. However, many developments in computer science allow us to construct abstractions onto the mathematical expressions, and these serve as useful metaphors for aesthetic application. For example, the object-oriented paradigm in software engineering serves as a guideline for creating artistic objects, and the agent-based paradigm serves equally as well for creating artistic agents. Each software paradigm indicates a way in which mathematical elements may be viewed, heard, or touched.

rube

rube is a software project began two years ago in our research group. The primary purpose of rube is to facilitate aesthetic formal models. The package is currently tailor made for model builders who have a scene graph, containing geometry elements, and want to combine this with a formal specification of a dynamic model. These two files are termed the *scene* and *model* files. Both files are expressed in the new lingua franca of the web, the extensible markup language (XML). XML turns out to be an ideal vehicle for personalization since it is designed around the idea of styles. Formerly termed *style sheets,* there are now more powerful style translation mechanisms (XSLT). The scene file serves in the same capacity as the *skinz* of customizable desktop interfaces and applications. Figs. 3 through 5 are mathematical *skinz*.

The scene file is captured in the Extensible 3D (X3D) language, and the model file in the Multimodeling Exchange Language (MXL) being developed by our group. Both X3D and MXL have associated XML

Schemas and are integrated, with Javascript and Java, to create a combined, interactive 3D environment.

Figures 7 displays the formal structure of a Finite State Machine (FSM) containing 3 states and 3 directed transitions, all triggered by a binary input of one. When zero is received by the FSM, the FSM stays in its current state, with S1 being the start state. Figs. 8 through 10 display alternative aesthetics to the diagrammatic one in Fig. 7, but the underlying MXL files are equivalent to that of Fig. 7. Figure 8 displays primitive solids, not unlike fairly recent 3D programming language studies in the form of Najork's Cube language [9]. Figs 9 and 10 represent fluid flow and agent-based aesthetics.

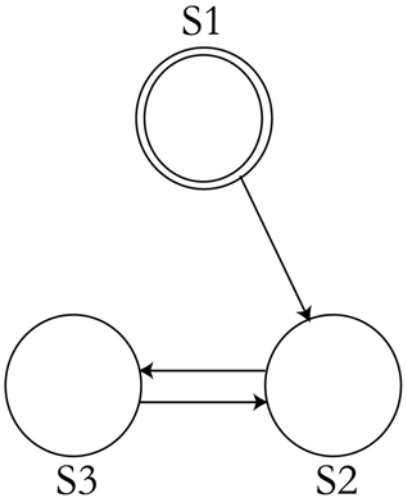


Figure 7: FSM in a 2D diagram

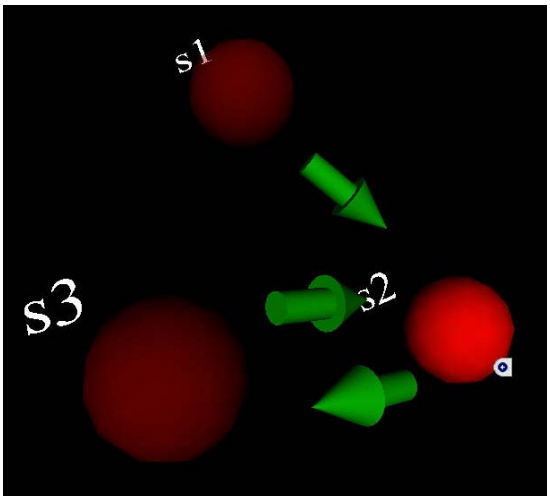


Figure 8: FSM using primitive solids

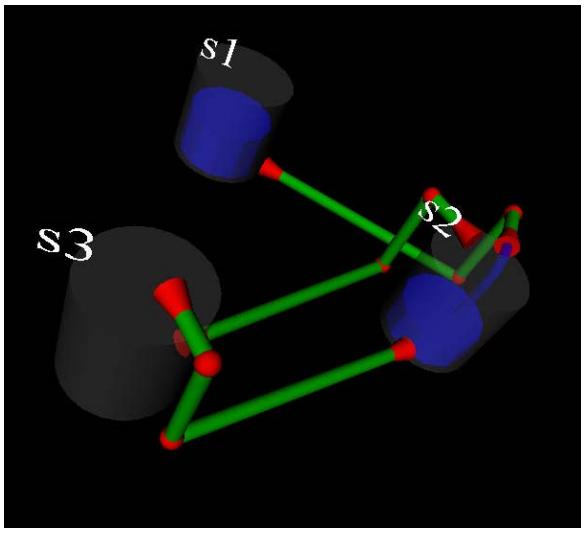


Figure 9: FSM using a fluid flow metaphor

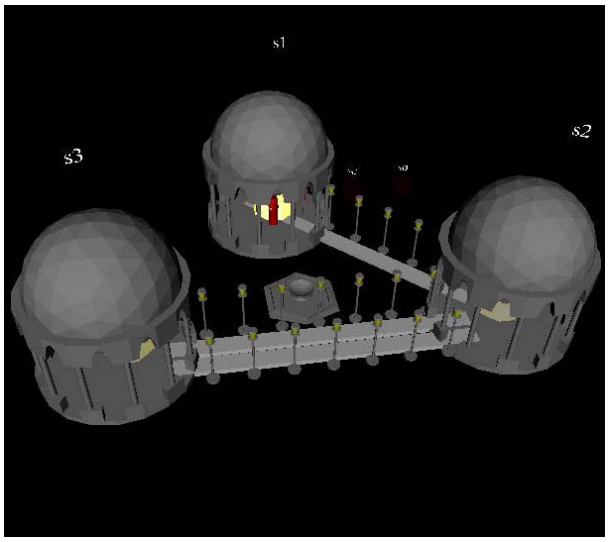
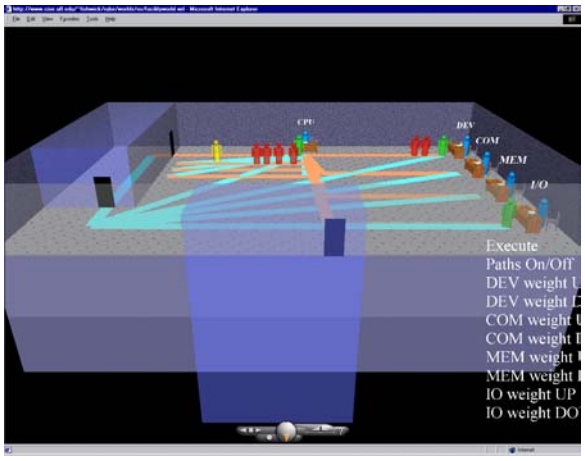
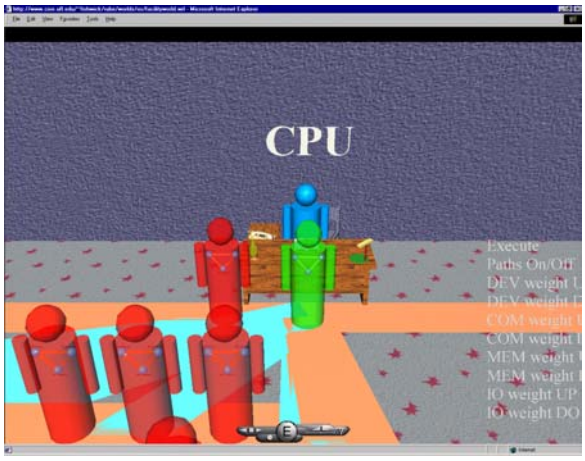


Figure 10: FSM using an agent metaphor

Figures 11(a) and 11(b) display a more complex program [10], a simple multi-tasking operating system, with programs being anthropomorphically presented as colored avatars, and OS resources being represented by people acting as servers behind desks. Different floor paths are defined to allow for avatars to queue behind each resource.



(a) Front View of complete floor



(b) Close-up view of tasks awaiting CPU allocation

Figure 11: Operating system architecture designed with an agent metaphor

There are two different approaches to research on model structures for computing and mathematics: *breadth-first* and *depth-first*, to use a tree search analogy. In the depth-first approach, one specific presentation—

an aesthetic—is chosen and studied in great detail for issues regarding its suitability for capturing all of the nuances of the formal semantics and robustness. This approach is akin to the process of science, where the goal is to seek out uniformity and to perform analysis. Our research is more along the lines of the breadth-first approach, where we do not single out one presentation style, but rather facilitate multiple styles, aesthetics, and personalized models via MXL. This approach is akin to engineering and art. This leads our work into other areas such as the ability of MXL to capture a broad range of styles. Clearly, there are likely to be many issues that plague 3D constructions in particular, most notably the layout problem (for semi-automated assistance in creating the scene) and the navigation problem (for being able to effectively peruse the model objects). Both the depth and breadth approaches are needed for aesthetic computing, if it is to progress.

Summary

Aesthetic computing opens the doors for a highly interdisciplinary engagement of artistic forms, styles, and aesthetics for the re-presentation of formalisms. The style-based approach to XML, as the basis of the future World Wide Web, and the trend toward personalization and mass customization in marketing and manufacturing indicate that this interdisciplinary work is needed. We have recently begun research on a number of fronts to build *rube*: 1) including the use of sound and narrative to increase the flexibility of modeling, 2) surveying and testing students and faculty to obtain data sets for analysis to judge which issues are paramount in the area, and 3) working with artists, new media experts, and visualization researchers to flesh out remaining issues. Our plan is for *rube* is to develop MXL to where it can be widely employed for scientific and engineering model building, while directly supporting individualized and group-oriented perspectives and aesthetics.

Acknowledgments

Many thanks to my present and recently graduated students Taewoo Kim, John Hopkins, and Linda Dance for their contributions in this area. I am also indebted to both the National Science Foundation under grant EIA-0119532 and the Air Force Research Laboratory grant F30602-01-1-0592 for their financial sponsorship, and to my co-PIs on these two grants, Jane Douglas (English), Timothy Davis (CISE), and Joachim Hammer (CISE).

References

[1] Knuth, Donald, *Literate Programming*, Technical Report STAN-CS-83-981, Stanford University, Dept. of Computer Science, 1983. See http://www.literateprogramming.com for details on the general topic.
[2] Card, Stuart, MacKinlay, Jock and Shneiderman, Ben, Eds, *Information Visualization: Using Vision to Think*, Morgan Kaufmann, 1999.
[3] Stasko, John, Price, Blaine and Brown, Marc, Eds, *Software Visualization: Programming as a Multimedia Experience*, MIT Press, 1998.
[4] Fishwick, Paul *Simulation Model Design and Execution: Building Digital Worlds*, Prentice Hall, 1995.
[5] Fishwick, Paul Aesthetic Computing home page http://www.cise.ufl.edu/~fishwick/cap6836
[6] Schmandt-Besserat, Denise. *How Writing Came About*, University of Texas Press, 1997. See also http://www.utexas.edu/cola/depts/lrc/numerals/dsb/dsb1.html
[7] Digital Art and Science (DAS) and Digital World Institute, http://www.digitalworlds.ufl.edu
[8] Fishwick, Paul *Aesthetic Programming*, to appear in Leonardo, MIT Press, 2002.
[9] Najork, Marc *Programming in Three Dimensions*, Journal of Visual Languages and Computing, 7(2): 219-242, June 1996.
[10] Hopkins, John and Fishwick, Paul *A Three-Dimensional Human Agent Metaphor for Modeling and Simulation,* Proceedings of the IEEE, 89(2): 131-147, February 2001.

Copyright © 2001 Paul Fishwick

View publication stats