

Exploring Multiple Visualization Perspectives with Aesthetic Computing

Paul Fishwick, *Senior Member, IEEE*

Abstract—The task of visualization, as it applies to computing, includes by default the notion of pluralism and perspectivism since there is an explicit attempt at *representing* one, often textual, interface in terms of a more graphical one. This desire for alternate perspectives is consistent with art theory and practice, and even though rigor and formalism generally mean different things to artists and computer scientists, there is room for collaboration and connection by applying artistic aesthetics to computing, while maintaining that which makes computing a viable, usable field. This new area is called *aesthetic computing*. Within this area, there is an attempt to balance qualitative with quantitative representational aspects of visual computing, recognizing that *aesthetics* creates a dimension that is consistent with supporting numerous visual perspectives. I introduce one aspect of aesthetic computing, with specific examples from our research and teaching to illustrate the potential and possibilities associated with alternate representations. We show that by linking with aesthetics, we surface some important philosophical and cultural questions regarding notation, which turn out to be at least as important as the algorithmic and procedural means of achieving customized model component representations.

Index Terms—modeling, aesthetics, customization, art, computing.

I. INTRODUCTION

THERE is an anonymous quote that goes something like this: *I would rather see one sunset through a thousand eyes than a thousand sunsets*. A slightly prosaic understanding of this suggests that we learn about a thing by experiencing multiple perspectives, with different visualizations, materials and ways of crafting. This approach is generally borne out when studying a piece of knowledge that we are trying to acquire. If I want to dive into the theory of groups in algebra, I am more likely to understand the material by reading different books on the subject, by doing several examples, and by asking different people from different backgrounds about their perspectives on groups. I would want to see an example of a group, and its effects, in multiple ways: textually and graphically.

The topic of *Aesthetic Computing* relies on multiple perspectives for different aspects of computing, from the mathematical foundations of computing (theory, discrete structures, program and data structures, database schemata, networks) to human-computer interaction and the topic of developing more diverse and customized [1], or personal, human interfaces. Aesthetic Computing is defined as the application of art theory and practice to computing. The underlying assumption is that by employing a diverse array of aesthetics and natural or artificial cultural artifacts, that we permit an exploration of different views for *representation* within computing. Another

assumption is that we are providing tools for increased usability, with some recent justification of this by Tractinsky [39], who suggests that aesthetics can have empirically definable positive results on usability and functionality. Before proceeding further, I should address what might be the first question for the Computer Scientist: namely, why this area is new given that we have always had aesthetics in both mathematics and computing.

To answer this requires some background on aesthetics, for the term has meant different things to different people depending on when it was used. In ancient Greece, aesthetics were synonymous with the principle of mimetics—that art was best when it resembled a target object. During the same time period, Plato invented the *forms*, which stood at the top of a hierarchy, representing the pinnacle of knowledge. Leaping forward to the 18th century, Kant reinvigorated a kind of Platonic view of aesthetics with the idea that *disinterest* and a cognitive reflection, and not so much visual appearance, were the key aspects of aesthetics. Today, the majority of aestheticians would agree that art employs a balance of the mind and matter, of cognition and the body. Returning to our topic, mathematics has often been treated as a creature of the mind, with the emphasis being less on the senses and more on cognitive activity. Thus, an elegant proof entails a minimal number of sentential connectives, but many mathematicians would not view notation in the same light; however, there is evidence that mathematicians create significant visualization, at least internally, when posing and solving problems [2].

With regard to computing, which can be seen as a significant outgrowth of mathematics, Knuth has written of the importance of the material aspects of aesthetics within his thrust area of *literate programming* [4]. Much of the reason for his creation of the \TeX typesetting language, resulting in the popular \LaTeX spinoff, seems to rest on his definition of aesthetics in style and presentation, which ventures beyond the purely mental into the *material* realm. For example, in the Metafont book [5], we find the following in the preface “Type design can be hazardous to your other interests. Once you get hooked, you will develop intense feelings about letterforms; the medium will intrude on the messages that you read.” Knuth was one of the first to make this connection within Computer Science; however this kind of thinking was also at the heart of Marshall McLuhan’s dictum “The medium is the message” [6]. McLuhan’s main thesis was that information was not only a matter of denotation, but that connotative effects of the material were equally as essential and influential for the task of communication.

Recently, there has been significant research in visualization for both software notation as well as execution [7], [8], [9],

[10], [11]. This work builds on top of the almost ubiquitous *flowchart* graphics so common in the 60s and 70s, but creates a substantial discipline out of it. However, the range of aesthetics and styles in representing software and execution results is rarely tied to art by allowing artistic practice, or an associated focus on customization and personalization, to play a major role. The recognition of domain-specific visual languages as a distinct area is more recent, with workshop proceedings serving as the major dissemination outlets. Esser and Janneck [12] provide a conceptual treatment based on a definition of domain-specific languages as being “tailored to a particular problem domain.” For Aesthetic Computing, there is a distinct connection with this approach, except that the word *problem* can be replaced by *cultural*. The more abstract the formalism, the greater the diversity possible in choosing cultural iconography to notate the formal elements. Aesthetics takes on the role of an extra dimension in the area of Information Visualization [13], with the possibility to represent graph structure, for instance, with markedly different objects and styles.

My purpose is to define aesthetic computing, show examples, and discuss the key issues that center on the technique. The paper is meant as an introduction to aesthetic computing, and its relation to visual computing, with other publications [1], [14] serving to define the algorithms and technical approaches required to achieve this goal. The notion is that problems in visual computing are not solely of a technical nature. My thesis is that we may, as a visual computing community, need to examine non-technical, aesthetic, cultural and philosophical assumptions about the way that we represent and notate software and other formal structures. Some of the new directions that we might take, suggested by aesthetic computing, are created by questioning assumptions. I’ll proceed by starting with a simple illustration of aesthetic computing, and then ask key questions identified through the classic questions *What?*, *Why?*, and *How?*. I then close the paper by summarizing the state of the art, and where we might venture in the future. The examples and explanations are oriented toward one part of aesthetic computing that interests our research lab, rather than trying to provide more general coverage. This part centers on representing formal elements in computing and mathematics, with a concentration on elements used in computer simulation and programming.

II. AESTHETIC COMPUTING EXAMPLE

In computing, the finite state machine (FSM) is ubiquitous, found in lexical scanners for language parsers and scripting languages, and in behavior encodings for artificial agents in interactive games. Let’s consider the following Moore machine M :

- $M = \langle Q, I, O, \delta, \lambda \rangle$
- $Q = \{S1, S2, S3\}$
- $\delta : Q \times I \rightarrow Q, \delta(Si, 0) = Si$ for $n \in \{1, 2, 3\}, \delta(S1, 1) = S2, \delta(S2, 1) = S3, \delta(S3, 1) = S2$
- $I = \{0, 1\}$
- $\lambda : Q \rightarrow O, \lambda(Sn) = Sn, n \in \{1, 2, 3\}$

The machine has 3 states ($S1$, $S2$, and $S3$) with an input value of 1 achieving a change in state. An input of 0 leaves the machine in the same state. The machine oscillates between

two states $S2$ and $S3$ after it gets a jump start from $S1$, and a subsequent stream of ones. This definition is compact and has a typographic presentation, which is amenable to most input and output devices. It has the distinct advantage of having been formulated centuries ago, first with wooden matrices, and then with metal, and more recently with computer typesetting hardware and software. An important aspect to visual computing is that we need not toss out one representation to make way for another. We recognize the power of the typographic notation, while at the same time recognizing that notation is largely at the mercy of whatever technologies exist to support it. Without the technology for making paper or parchment, one is limited to scratching marks on surfaces.

Now consider Figures 1(a) through 1(d), which illustrate four other views of this particular FSM. Each of these views can be considered graph visualizations, as M is a directed graph. The illustrations are not meant to be complete diagrams with all necessary information required to formally specify the static FSM structure, but rather snapshots of *interactive* interfaces suggesting alternate presentations. Thus, one should not expect these figures to carry the same static information content as the dynamic, interactive counterpart which cannot be inserted into a print medium. Transition inputs of M are assumed to be labeled with 1 with reflexive transitions triggered by a 0. Given these assumptions and this *class* of machine, consider Fig. 1(a), representing a 2D diagram of M . In a related logic-based diagrammatic application, Shin [15] demonstrates that iconography can be every bit as sound as complete as the typographic presentation. One infers that the reason why the digram is not as prevalent as type is due primarily to economy: as displays become cheaper and more plentiful, the reasons for using diagrams, and more generally visualizations, become less questionable and increasingly justifiable. Such a sentiment, or perhaps leap of faith, needs little encouragement within the computer visualization communities. Still, we gain greater recognition that that our modes of presentation are governed by the economy of materials and labor. This suggests that as these economies shift, that our notations need to evolve, and that we have the opportunity of exploring alternate evolutionary paths. Fig. 1(b) is made from 3D primitives, echoing the sort of research studies by Lieberman [16] (i.e., program execution) and Najork [17] (i.e., program structure). In particular, Fig. 1(b) is similar to Najork’s CUBE elements except that several primitives are used (i.e., sphere, cylinder, and cone). Spheres represent states, in Fig. 1(b), and cylinders and cones combine to form 3D arrows. Apart from being the first 3D programming language, CUBE also introduced new programming concepts (i.e., combined data flow/logic capability and static polymorphic typing) to the evolving area of visual programming. As with Shin’s work, Najork is concerned with issues of soundness and completeness within the visual framework, ensuring that formal attributes of textual languages are carried over into the visual domain—in this case one with 3D primitives.

Fig. 1(c) goes a step further than Fig. 1(b) by employing analogy and metaphor, allowing us to take advantage of our knowledge of fluids, pipes, and containers in representing the FSM operation. As with any metaphor, we must carefully

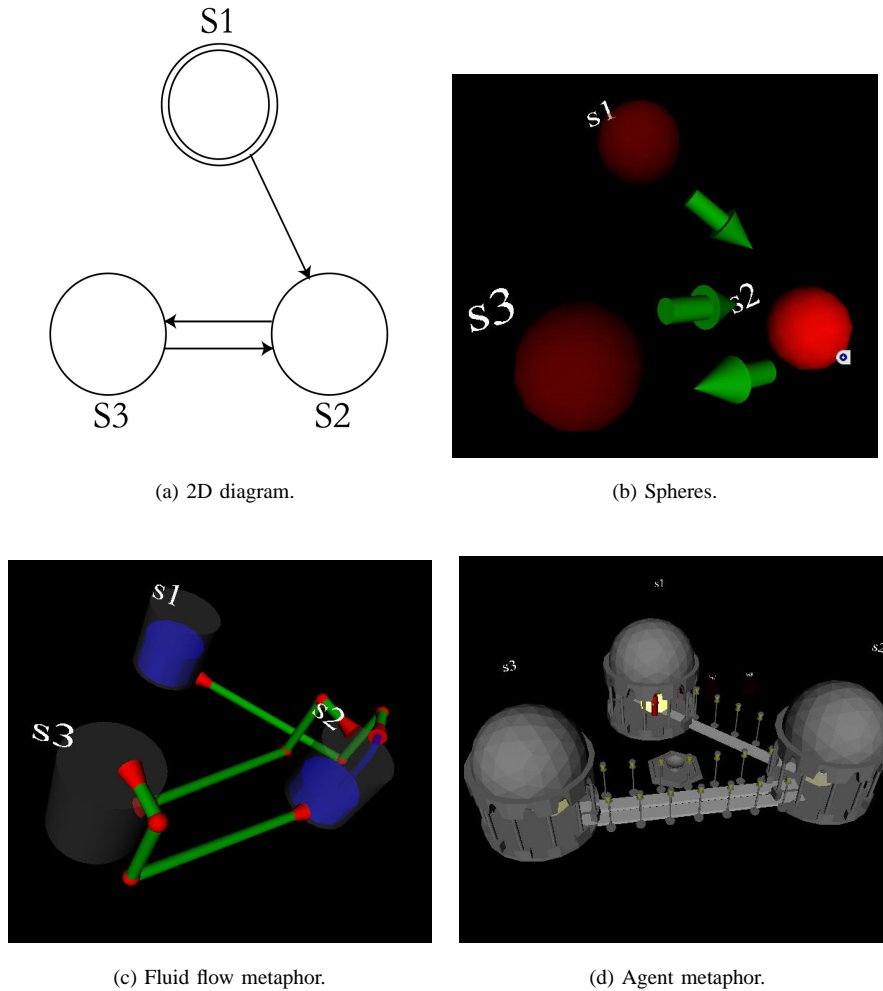


Fig. 1. Four different visual perspectives (i.e. presentations) of machine M .

qualify it, taking from it items that we wish to transfer while ignoring, or abstracting away, other items. For example, even though the FSM uses discrete inputs to drive state change, the VRML (Virtual Reality Modeling Language) animation associated with Fig. 1(c) shows a continuous fluid animation to capture this discrete change: the fluid nature of water is used to capture the idea of transportation, but we are to *ignore* speed or time needed for fluid transfer. Hesse [18] draws attention to this distinction, with definitions of positive, negative and neutral analogies. Positive analogies are attributes that carry through from source to target, like the directionality inherent with fluid flow. Negative analogies refer to attributes that we are to ignore when forming the connection. Even though, this could be seen as an inherent weakness of metaphors in general, the natural affinity humans have for analogy and metaphor allow us to work past these constraints. That is, the medicine of metaphor may have some negative side effects, but its overall benefits outweigh these effects, and are substantial [19], [20]. The ability to interpret Figs. 1(c) and (d) as discrete transitions is learned, and represents an arbitrary convention that can be associated with the visual encoding. As long as it is made clear that FSMs, for a particular group advocating this visual

style, are defined in a consistent manner with the semantics of discrete transition, then there is no serious issue. There are numerous cases of 2D arrows being used by different visual dynamic models [21] to define both continuous and discrete transitions, so even the meaning of a simple arrow is by no means obvious or natural.

Figure 1(d) is an agent-based metaphor, where gazebo-like structures represent states and a woman walks along a lamp-lit pathway to move from one state to another. Internal lamps within each gazebo light up when that state is active. This example brings out the idea that abstraction need not imply material or visual abstraction, since we can create human body models to do our bidding within formal structures, and that these non-minimal, visual structures are every bit as abstract as the equivalent typographical presentation. Abstraction is integral to the referential, not visual, attributes of an object. The abstraction is indicated by the one-to-many mapping associated with an element's symbolic reference regarding its role as a state or a transition. Some issues surrounding the use of these models are covered in Sec. IV.

Even though Figs. 1(b) through 1(d) are 3D, aesthetic computing as it pertains to representation is not solely focused

on 3D. Instead, it is focused on customization and the ability to easily explore a pluralistic set of forms. However, 3D seems to offer more opportunity in this regard than 2D, and so our models and methods tend to be oriented toward 3D models and animation.

III. What?

What is aesthetic computing? Some of the context for the area can be found in *multimodels* [21], which stress the idea of heterogeneity not only in formal model content, and coupling, but in a plethora of presentations. In 1999, our research group began a project called RUBE [22], to allow dynamic models (i.e., models specifying the temporal dynamics of systems) to be simulated over multiple abstraction levels, and with arbitrary representations. The motivation for the latter concept was derived in part by a new set of curricula we created called *Digital Arts and Sciences* (DAS) [23]. DAS bridges two colleges, Fine Arts and Engineering, with the idea of producing students who have a hybrid knowledge base allowing them to become “digital Leonardos,” but maintaining an academic, rather than vocational, orientation.

Some initial concepts in aesthetic computing were published [24], [25], and a Dagstuhl Seminar (cosponsored by *Leonardo* [26]) on the subject was convened in southwestern Germany in July 2002 [27], with colleagues Roger Malina and Christa Sommerer. We had attendees representing a broad array of mathematicians, computer scientists, designers, and artists. An edited book volume is in progress.

Aesthetic Computing, as an emerging area, is broader than the examples show in Sec. II. As the field is defined by the application of art theory and practice to computing, researchers span a gamut from those with an interest in the nature of aesthetics and semiotics, to interface and design specialists. The majority of attendees are concerned with aesthetics of the human-computer *interface*, some with an interest in synthesis, and others with a more analytic orientation. Our work, as evidenced by the simple example in Sec. I is one aspect of aesthetic computing, and is driven by an interest in representing mathematical formalisms found in computing: discrete structures, program, and data structures, and applied mathematics especially for dynamic systems typically employed for computer simulation.

IV. Why?

Why do aesthetic computing? Consider a *gedanken* experiment where one is able to conjure up any immersive holographic scene with advanced technological ease. Houses and landscapes, in such an environment, would be as easy to synthesize as circles and squares in today’s computing environments. Two questions arise: (1) what could we do with such technology in terms of representing formal computing structure and behavior, and (2) what issues arise as a result? A conservative attitude may be at the core of thinking that with this advanced technology, we would still use typography, draw circles, or mold spheres, but this seems unlikely when we consider technology trends from ancient to modern times. The art of representation is certainly constrained by technology, but

it is worthwhile considering the role of abstraction, since this is at the heart of several issues regarding aesthetic computing. Are we in violation of principles of economy and abstraction when artistically rendering formalisms? Is it always better to tend toward the minimal when doing computing?

Let us try to address these questions by first considering that abstraction is a form of *information reduction*. I can take an object for “tree” and imagine that the object refers to the set of all trees that exist in the world. Mathematical concepts of number, set theory, and algebra all leverage abstract thinking. However, this sort of abstraction is different than material reduction associated with using a circle instead of a sphere, or a sphere instead of a house. As long as the object in question can be visually identified as a reference for a target object, we are likely to employ whatever economical tools are available to us—which is why we scratched marks in clay [28] rather than synthesizing more substantial objects in a *Holodeck*¹ environment. However, the tendency for mathematicians and computer scientists to eschew substantial forms in lieu of small marks is strong, and has a long history. There may be a feeling that economy of thought requires economy of representation down to the very atoms and bits used to capture what an object looks or sounds like. Even the era of abstract art seems to suggest that we employ minimal presentational forms, until we remember that the use of abstraction is at least as strong within the surrealist genre [29], [30], and in that movement, objects are not depicted with simple geometric forms.

Abstraction may be one of the more salient aspects of aesthetic computing when critically analyzing it, but there are more facets of motivation and agenda items on the list of issues. Regarding motivations beyond the mental modeling exercise, one can always point to the age-old desire to better connect art with science. Science affects art, but in what ways does art affect science? For computing, new algorithms add to the painter’s palette or the sculptor’s hand tools, but aesthetic computing examines the converse situation where art affects computing. The issue arising from this thought is whether we can maintain qualities we consider important in computing as we mix artistic elements inside the computing barrel? After all, doesn’t art have its own motivations and goals, with some of these being in direct opposition-to or conflict-with computing? For answering the first part of this question, it is worthwhile noting that the Greeks referred to art, craft, and technology under the same name: *technē*. It is only within the past two hundred years that we have come to differentiate *fine art* from technological artifacts. It is possible, and achievable, to craft computing formalisms with an eye to aesthetics, without sacrificing rigor, completeness or soundness.

One of the aspects of artistic practice, which makes it attractive to computing, is its focus on the individual and the cultural. Personalization and customization, despite valid privacy issues associated with them, have become important attributes of constructivism [31] in education, and manufacturing [32] in the commercial sector. The idea of having many ways to represent something is usually viewed as a highly

¹The *Holodeck* is a fictional environment that first appeared in the popular science fiction show, *Star Trek: The Next Generation*, produced by Paramount Pictures.

positive concept, and this concept is quite consistent with the goals of art. Issues can arise based on who is doing the fine-tuning of the representation and what sorts of information they have at their disposal, some of which may be deemed too personal for some individuals. Like the issue of abstraction, the issue of *standardization* jumps to the forefront in discussions of customization. How can we standardize notation if everyone fosters a Tower of Babel? The first answer to this is that we need not eliminate one representation in favor of another. We can mix and blend notations together, or use one notation when in one cultural setting, and another notation for a different group, or for a different purpose, such as learning rather than communicating. At least within the literature on dynamic systems [21], there are a number of different visual modeling techniques, and yet we can all use the existing and powerful textual notation that has wide coverage and is well-understood by those having studied grade school mathematics.

V. How?

How does one *represent* formal structure with aesthetic computing? We can look at this from a theoretical perspective first, and then say something of our current software implementation. Regarding methodology, I have taught three classes in Aesthetic Computing at the University of Florida, in each of the last Spring semesters. The class is primarily a *special topics* class with some lecture, deliverables, student talks, and invited speakers [33]. The lectures focus on issues connected with the following:

- *representation*: a review of traditional notation for mathematics and computing, with examples drawn from algebra, discrete structures, programming, and dynamic systems. At this stage, students learn to begin with what they know, using the text-based notation that is familiar to them.
- *mapping*: a slew of techniques for mapping elements and systems to different representations, usually in target domains such as art, theatre, architecture, and machines. We start with creativity exercises and games, and then move onto computational approaches for analogy and metaphor.
- *crafting*: the actual making of the aesthetic computing target object—for example, a representation of a data flow network. This involves a discussion of different tools for 2D and 3D graphics, as well as sound editing. We discuss model juxtaposition techniques so that students can choose how to combine or blend a model with the system being modeled. Automated layout techniques, where appropriate, are presented.
- *assessment*: gauging the quality of the product, ensuring that it meets requirements for items such as completeness, consistency, scalability, usability, and aesthetics.

We have built a software system, called RUBE, which facilitates model-building in aesthetic computing and computer simulation. Recently, we have embarked on an approach that employs authoring tools, whereas in the past, we have relied upon text editors and ad hoc tools. For the new RUBE, we need to begin with an authoring method in either 2D or

3D. For 2D, we use SodiPodi [34], and for 3D, we use Blender [35]. Currently, the source formalisms for RUBE are those used in modeling dynamic systems: finite state automata, data flow networks, Petri nets, ordinary differential equations, and System Dynamics graphs. The approach is to allow the modeler to use these tools, while specifying what each graphical object means. The meaning is specified with an XML (eXtensible Markup Language) language called MXL (Multimodeling eXchange Language). MXL captures the topology of a system without specifying presentation, and is translated into a lower “assembly” layer DXL (Dynamics eXchange Language) elements, which are then translated into target code, currently Javascript, with Java and Python planned for future support.

VI. CONCLUSIONS

Even though we are constructing an initiative and environment conducive to pluralistic representation of formal mathematical and computing structures, it is natural to question whether anyone will use such a facility even if it is made efficient. Apart from our own inherent optimism, which drives our research, we see a positive set of trends that suggest that as a technologically-based society, we are moving in a direction consistent with the goals of aesthetic computing:

- Notation for mathematics and computing, which encompasses how we view program and data structures, is historically constrained by technology and economy. We move in a direction that is economical, avoiding complex, expensive approaches. The areas of visualization within computing, including those of information, software, and science, are still new, and made possible through early work in the 60s when computer graphics got its jump start. The formation, in the late 70s and 80s, of the graphical user interface was a major step in, not eliminating text, but augmenting it to achieve more humane connections to code. It is not too surprising to imagine that this movement will affect structures not only in computing, but also in mathematics.
- Ever since the early Greek state of confluence of art and science, in the spirit of *technē*, the two areas have split and separated. Efforts in mass production benefited society at large, but caused another wedge to increase the size of the split; however, it is ironic that mass customization builds directly on mass production, and so is making it economically feasible to personalize and customize objects. The *skinz* movement is one example of this in computing; however, at least within the area of dynamic systems, there is a half-century of developing new model structures, of permitting cultural groups to flourish within technical societies. It is possible to have standards and pluralism at the same time, permitting multiple metaphors to coexist.

An impediment to progress in representation is the inertia associated with our present biases and cultures. Mathematicians and computer scientists have been trained to think non-visually, and state of the art programming languages are still textual. UML [37] seems to be thriving, but it is used mainly as

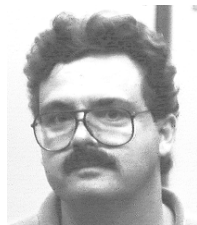
a requirements specification tool, not a language for creating code from diagrammatic forms. Until we get to that stage with UML or other contenders, visual computing paradigms will remain in their infancy. We have started one thread of research around using RUBE for programming, and hope to have results within the year. If the field of computing can be thought of as being slow to adopt visual programming languages, the field of mathematics proceeds at snail's pace. We have relied, successfully, on textual notation for many centuries but it is time to explore visual approaches, catalyzed by the work done in the visual computing community.

ACKNOWLEDGMENT

I would like to thank the National Science Foundation under grant EIA-0119532 and the Air Force Research Laboratory under grant F30602-01-1-05920119532, with special thanks to my current students working RUBE: Minh Park, Jinho Lee, Hyunju Shim, Kristian Damkjer, and John Hays, and Joella Walz for her independent study with myself and colleague Georg Essl. I would also like to thank the student population of the *Aesthetic Computing* classes for their valuable feedback and projects. I would also like to thank the anonymous referees for suggesting critical improvements to this paper.

REFERENCES

- [1] J. Hopkins and P. A. Fishwick, "Exploiting an Agent-Based Metaphor in Software Visualization using the rube System," *Journal of Visual Languages and Computing*, vol. 14, pp. 97–117, February 2003.
- [2] J. Hadamard, *The Psychology of Invention in the Mathematical Field*. Princeton University Press, 1945.
- [3] M. Emmer, Ed., *The Visual Mind: Art and Mathematics*. MIT Press, 1993.
- [4] D. E. Knuth, *Literate Programming*. CSLI Lecture Notes, Number 17, 1992.
- [5] —, *The METAFONT book*. Addison-Wesley Publishing Co., 1986.
- [6] M. McLuhan, *Understanding Media: the Extensions of Man*. MIT Press, 1994.
- [7] M. H. Brown, *Algorithm Animation*. MIT Press, 1987.
- [8] N. Shu, *Visual Programming*. New York: Van Nostrand Reinhold, 1988.
- [9] S.-K. Chang, Ed., *Visual Languages and Visual Programming*. New York: Plenum Press, 1990.
- [10] J. Stasko, J. Domingue, M. H. Brown, and B. A. Price, Eds., *Software Visualization: Programming as a Multimedia Experience*. MIT Press, 1998.
- [11] S. Diehl, Ed., *Software Visualization*. Springer Verlag, May 2001, INCS 2269.
- [12] R. Esser and J. W. Janneck, "A Framework for Defining Domain-Specific Visual Languages," in *Workshop on Domain Specific Visual Languages, ACM Conference on Object-Oriented Programming, Systems, Languages and Applications (OOPSLA-2001)*, October 2001.
- [13] S. K. Card, J. Mackinlay, and B. Shneiderman, Eds., *Readings in Information Visualization: Using Vision to Think*. Morgan Kaufmann, 1999.
- [14] T. Kim, J. Lee, and P. Fishwick, "A Two-Stage Modeling and Simulation Process for Web-Based Modeling and Simulation," *ACM Transactions on Modeling and Simulation*, vol. 12, no. 3, pp. 230–248, July 2002.
- [15] S.-J. Shin, *The Iconic Logic of Peirce's Graphs*. MIT Press, 2002.
- [16] H. Lieberman, "A Three-Dimensional Representation for Program Execution," in *Visual Programming Environments: Applications and Issues*, E. P. Glinert, Ed. IEEE Press, 1991.
- [17] M. Najork, "Programming in Three Dimensions," *Journal of Visual Languages and Computing*, vol. 7, no. 2, pp. 219–242, June 1996.
- [18] M. Hesse, *Models and Analogy in Science*. University of Notre Dame Press, 1966.
- [19] G. Lakoff and M. Johnson, *Metaphors We Live By*. University of Chicago Press, 1980.
- [20] G. Lakoff, *Women, Fire and Dangerous Things: what categories reveal about the mind*. University of Chicago Press, 1987.
- [21] P. A. Fishwick, *Simulation Model Design and Execution: Building Digital Worlds*. Prentice Hall, 1995.
- [22] J. F. Hopkins and P. A. Fishwick, "Synthetic Human Agents for Modeling and Simulation," *Proceedings of the IEEE*, vol. 89, pp. 131–147, February 2001.
- [23] (2003) Digital arts and sciences. [Online]. Available: <http://www.cise.ufl.edu/dept/das/das.html>
- [24] P. A. Fishwick, "Aesthetic Programming: Crafting Personalized Software," *Leonardo*, vol. 35, pp. 383–390, 2002.
- [25] —, "Aesthetic Computing: Making Artistic Mathematics and Software," *YLEM Journal: Artists Using Science & Technology*, vol. 10, pp. 6–11, September 2002, special Issue on Art and Programming.
- [26] R. Malina, "The Stone Age of the Digital Arts," *Leonardo*, vol. 35, no. 5, pp. 463–465, October 2002.
- [27] (2002) Aesthetic computing seminar. [Online]. Available: <http://www.dagstuhl.de/02291/>
- [28] D. Schmandt-Besserat, *How Writing Came About*. University of Texas Press, 1997.
- [29] W. Bohn, *The Rise of Surrealism*. State University of New York Press, 2002.
- [30] R. Arnheim, *Toward a Psychology of Art: Collected Essays*. University of California Press, 1966.
- [31] S. Papert, *Mind Storms: Children, Computers, and Powerful Ideas*. Basic Books, 1980.
- [32] B. J. Pine, *Mass Customization: The New Frontier in Business Competition*. Harvard Business School Press, 1993.
- [33] (2003) Aesthetic computing. [Online]. Available: <http://www.cise.ufl.edu/~fishwick/aescomputing>
- [34] (2003) Sodipodi. [Online]. Available: <http://sodipodi.sourceforge.net>
- [35] (2003) Blender. [Online]. Available: <http://www.blender.org>
- [36] G. Essl, "Physical Wave Propagation Modeling for Real-Time Synthesis of Natural Sounds," Ph.D. dissertation, Princeton University, November 2002.
- [37] J. Rumbaugh, I. Jacobson, and G. Booch, *The Unified Modeling Language Reference Manual*. Reading, MA: Addison-Wesley, 1999.
- [38] Frank lloyd wright. [Online]. Available: <http://www.cmgww.com/historic/flw/quotes.html>
- [39] N. Tractinsky, A. Shoval-Katz, and D. Ikar, "What is Beautiful is Usable," *Interacting with Computers*, vol. 13, pp. 127–145, 2000.



Paul Fishwick Paul A. Fishwick is Professor of Computer and Information Science and Engineering at the University of Florida. He received the PhD in Computer and Information Science from the University of Pennsylvania in 1986, and has six years of industrial and government production and research experience (Newport News Shipbuilding and NASA Langley Research Center). His research interests are in computer simulation modeling and analysis methods for complex systems. He is a Senior Member of the IEEE and a Fellow of the Society for Computer Simulation. He is also a member of the IEEE Society for Systems, Man and Cybernetics, ACM and AAAI. Dr. Fishwick founded the comp.simulation Internet news group (Simulation Digest) in 1987, which has served numerous subscribers. He has chaired several workshops and conferences in the area of computer simulation, including serving as General Chair of the 2000 Winter Simulation Conference. He was chairman of the IEEE Computer Society technical committee on simulation (TCSIM) for two years (1988-1990) and he is on the editorial boards of several journals including the ACM Transactions on Modeling and Computer Simulation, IEEE Transactions on Systems, Man and Cybernetics, The Transactions of the Society for Computer Simulation, International Journal of Computer Simulation, and the Journal of Systems Engineering. He has delivered 10 Keynote addresses at major conferences relating to simulation. Dr. Fishwick's WWW home page is <http://www.cise.ufl.edu/~fishwick> and his E-mail address is fishwick@cise.ufl.edu. He has published over 125 technical publications, written one textbook, co-edited two Springer Verlag volumes in simulation, and published seven book chapters.