The ACSE Multimedia Science Learning Environment

John F. Pane **Philip L. Miller**

Carnegie Mellon University
School of Computer Science
5000 Forbes Avenue
Pittsburgh, PA 15213 USA
E-mail: pane+icce@cs.cmu.edu

Abstract

Advanced Computing for Science Education (ACSE) is a multimedia science learning environment. Lessons, called ACSE Volumes, serve as electronic laboratory units in a developmental biology course. Each ACSE Volume is focused on a single scientific research area. The centerpiece of each Volume is a simulation of a scientific process. In exposing the scientific essence of the simulation to students, ACSE promotes the use of a programming language to convey scientific information. Snippets of executable simulation code are interspersed with other media to form a literate lesson. The simulation is open to student modification, and experimentation is encouraged. The learner is invited to extend the central theory or pose alternative hypotheses, and then test them in the simulation. ACSE investigates whether students with one semester of computer programming experience can participate in science and communicate their results through ACSE Volumes. Early empirical results and future directions are discussed.

1 Introduction

The Advanced Computing for Science Education project is a collaboration among Carnegie Mellon University's Department of Biological Sciences, Center for Light Microscope Imaging and Biotechnology, and School of Computer Science.[1] ACSE is producing a science learning environment and associated lessons. The environment is called ACSE and the lessons are known as ACSE Volumes. The Volumes are multi-media hyperdocuments that can be read like a book or journal. These Volumes exploit all of the modes of presentation available on the Macintosh computer. They are created in collaboration with a scientist, and each one provides a focused, in-depth journey to the frontier of knowledge in the scientist's current research area.

ACSE promotes a style of learning and scientific inquiry that is not available in current computer-based learning environments for science. The feature that distinguishes ACSE from other efforts is that students are able to go beyond what has been built into a Volume, even beyond what the builders of the Volume anticipated during its creation. This is because the centerpiece of each Volume is a simulation of a scientific process, and this simulation is open to modification by the student.

ACSE is built as an extension of the MacGnome Object Pascal Genie.[2] It allows the Volume author to bring scientifically important parts of the simulation to the attention of the learner, while hiding the parts that are irrelevant to the science. This invites the learner to extend the central theory or to pose alternative hypotheses.

In the simulation, science is presented in snippets of program code. One of the core goals of ACSE is to promote the use of a programming language to convey structured scientific information. The obvious advantage is that this kind of scientific discourse is *executable*. ACSE investigates whether student scientists with one semester of computer programming experience can do research and publish their results in this way.[3] Does a learning environment that features programming in this way improve student learning? Does it enhance student interest in the sciences by bringing them to the exciting forefronts of current knowledge?

The following sections describe the context of this work, show the need for ACSE, and discuss a prototype ACSE Volume. This is followed by a description of the elements of an ACSE Volume and the underlying programming environment, early evaluation of the prototype ACSE environment, and discussion of future directions.

---

1. The ACSE Project is work in progress supported by National Science Foundation Grant Number MDR-9150211.

2. The MacGnome programming environments, described below, were supported in part by National Science Foundation Grant Number MDR-8652015.

3. Ideally, this work would lead to the design of real laboratory experiments to confirm the simulation results.

2. Context

ACSE is an assembly of many of the best features of existing computer-based learning systems. The need for multimedia visualizations, videos, and animations of complex data and processes is easy to see. The idea of providing hypertext links among the elements of a lesson is common, and the bundling of tools so that students can bring together statistical analysis, graphing packages, document production software, and collaborative support is not new. Furthermore the use of simulation in learning is widely appreciated. All these are components of ACSE.

ACSE embraces the constructionist theory of learning, which views programming as an accessible and powerful form of knowledge called reflexive knowledge. Reflexive knowledge is capable of enhancing learning in other domains. Researchers such as [Papert, 1989] and [DiSessa, 1989] have explored the use of programming to teach science and math. ACSE extends this work to Pascal, a general purpose, widely-known programming language.

ACSE Volumes address topics in advanced science, and emphasize the learning of the science rather than the programming. ACSE provides the tools that allow graceful transitions from passive learning to parametric control of the simulation, and then to structural changes through programming. The programming is supported with state-of-the-art novice programming technology.

3 The Need For ACSE Volumes in Biology

There is a need for scientific visualization. Regardless whether data is obtained from simulation, various kinds of microscopes, or biomedical equipment like MRI, there is a trend toward huge amounts of data being available to the scientist. The need for good tools to navigate and view this data, as well as to present it to students and peers, is crucial.

There is a key role to be played by animation in teaching developmental biology. For example, embryogenesis is a highly dynamic process. It is intimately related to complex changes in spatial patterns of gene expression and the shapes of organs and tissues over time. The author of the most widely-used text of undergraduate developmental biology has noted that “developmental biologists get used to seeing nature in four dimensions.” Not all students are equally adept at four dimensional visualization, however, and any method that would improve their ability to understand complex morphological processes would enhance their appreciation of this discipline.

The revolution in molecular biology that has taken place over the past two decades has provided new insights into the mechanisms of fundamental developmental processes. These are perhaps the most exciting aspects of modern developmental biology, yet in order for students to understand this work in its proper context it is essential that they first be exposed to a large knowledge base concerning the fundamental morphology of the systems being discussed (i.e., the developmental anatomy of embryos of different organisms). To respond to this need, many universities devote entire courses to teaching developmental anatomy alone. Such courses consist of lectures employing slides or transparencies, almost always coupled with a laboratory session in which students study slides of serially-sectioned embryos with the aid of a text – usually an atlas of diagrams or photographs of similar histological sections. Other courses attempt to combine an introduction to developmental anatomy with a treatment of the underlying mechanisms of such morphological changes. This is the case, for example, at Carnegie Mellon. To combine both elements requires that a large amount of lecture time be devoted to a description of the morphology of embryonic development.

Animation is a much more efficient means of conveying this information to students. We integrate animations of key developmental processes with lecture material that focuses on the cellular and molecular mechanisms that underlie those processes. For example, the process of gastrulation, a phase of early development in which embryonic cells undergo complex coordinated movements, is very difficult to convey effectively using conventional methods. Moreover, there are interesting and potentially significant comparisons to be made between different species that require an examination of gastrulation in the embryos of several organisms. We are experimenting with conveying this information in sessions independent of lectures, in which the students interact with animations of gastrulation. This frees the lecturer to discuss more recent research concerning the molecular mechanisms of such cell movements. Other opportunities exist for effective use of animation, such as in the display of chemotactic signaling systems, temporal and spatial patterns of gene expression in developing systems, and current models of pattern formation based on gradients of diffusible morphogens.

One educational objective for ACSE is to enable students to participate in the business of thinking scientifically about cutting-edge science. Using programming as a language for scientific discourse offers the hope of facilitating student participation. By capturing contemporary theory in code, students can not only understand the theory, they can alter it and see the impacts of their ideas by executing the modified theory.

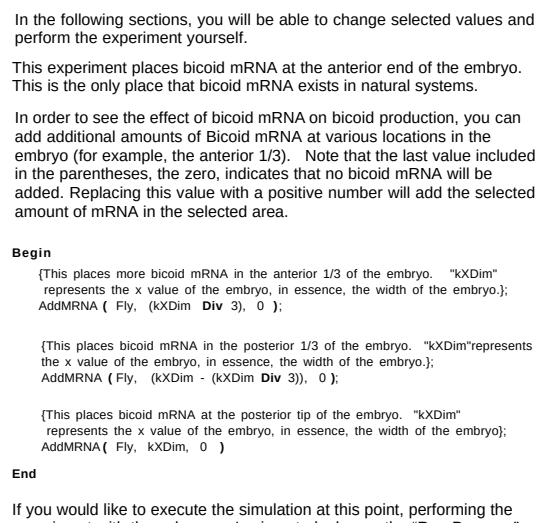
4 A Volume: Patterning in the Drosophila

An interesting aspect of development is patterning. Patterning is the process of cells changing from undifferentiated to highly specialized. The cells of the early embryo are difficult to distinguish from one another. Yet, as embryogenesis proceeds, cells in different regions of the embryo progressively acquire specialized properties. The

properties are based on different patterns of gene expression. Moreover, regions of gene expression form a precise pattern in the embryo; i.e., cells in the head region of the embryo begin to express a particular set of genes while those in the abdomen region express a different set. How then is the embryo subdivided in distinct domains of gene expression?

One explanation for this phenomenon is that expression of genes is regulated by concentrations of chemicals. It may not be obvious that complex differentiation patterns can emerge from simple chemical gradients. Drosophila, the common fruit fly, is a well-studied system that is used in this Volume to illustrate patterning based on a chemical gradient. The output of the Volume's simulation of this process is juxtaposed with micrographs of an actual embryo. In this way the student is able to observe correspondences and divergences between the simulation and the natural system. The point of this Volume is to teach students what is known and to let them explore the predominant theory.

It is known that a gradient of Bicoid mRNA exists in the embryo. The concentration is high at the anterior and low at the posterior. This is shown in graphs produced by the Volume's simulation. The student explores how the Bicoid mRNA gradient is formed, and the effect of the gradient. In the Volume, text, pictures, movies are interspersed with sections of live, editable code (see Figure 1).



In the following sections, you will be able to change selected values and perform the experiment yourself.

This experiment places bicoid mRNA at the anterior end of the embryo. This is the only place that bicoid mRNA exists in natural systems.

In order to see the effect of bicoid mRNA on bicoid production, you can add additional amounts of Bicoid mRNA at various locations in the embryo (for example, the anterior 1/3). Note that the last value included in the parentheses, the zero, indicates that no bicoid mRNA will be added. Replacing this value with a positive number will add the selected amount of mRNA in the selected area.

```
Begin
    {This places more bicoid mRNA in the anterior 1/3 of the embryo.  "kXDim"
     represents the x value of the embryo, in essence, the width of the embryo.};
    AddMRNA ( Fly,  (kXDim  Div  3),  0 );

    {This places bicoid mRNA in the posterior 1/3 of the embryo.  "kXDim"represents
    the x value of the embryo, in essence, the width of the embryo.};
    AddMRNA ( Fly,  (kXDim - (kXDim  Div  3)),  0 );

    {This places bicoid mRNA at the posterior tip of the embryo.  "kXDim"
     represents the x value of the embryo, in essence, the width of the embryo};
    AddMRNA ( Fly,  kXDim,  0  )
End
```

If you would like to execute the simulation at this point, performing the experiment with the values you've inserted, choose the "Run Program" command from the Run menu.

Figure 1: ACSE Volume Text with Live Simulation Code

This particular Volume has a number of different experiments, actually parametric changes to the simulation, that a student is directed to perform. The experiments successively add more and more of the detail that exists in the real developing embryo. After exploring these pre-set configurations, the student is encouraged to modify or even replace critical pieces of the simulation.

5 The Elements of an ACSE Volume

Visualization and Animation Tools

In general, ACSE takes advantage of the Macintosh Computer as a multimedia platform. ACSE has high quality graphics, and includes production-quality animations. Microscope data is combined and rendered so that the student sees rotating 3-D images rather than a series of 2-D slices. Quicktime movies of time-lapse video are included in the Volume. The simulations even produce 3-D graphic output and data graphs.

Abstraction

Software engineering, an important sub-field of computer science, has given the world the term *abstraction*. Computer scientists speak in terms of procedural abstraction. Procedural abstraction prescribes dividing problems up into constituents and then reasoning in terms of the names of those constituents. Computer scientists also speak of data abstraction. Data abstraction is the bundling of data objects together with the operations on those objects in such a way that they can be used reliably without knowledge of how they are implemented. Software systems are designed, written, maintained, and extended as a group of isolated subsystems. The virtue is that, when making an extension, the programmer need not keep an entire system in mind; just the one piece that is being worked on and its interface to the rest of the world.

Abstraction is of fundamental importance to ACSE. The MacGnome programming environments provide support for reducing background information, things already learned, and things to be learned later, into tidy black boxes. The MacGnome environment enables ACSE to separate the essential science, the thing being taught, from the mathematical and computational underpinnings. Students work directly with the science while dealing with everything else as a well-behaved abstraction.

There is a body of literature on the importance of abstracting away the mathematically or computationally necessary. R. Beare, of Warwick University, United Kingdom, articulates the need for abstraction: "As it is usually taught, physics traditionally involves a lot of quite difficult mathematics. This is a pity because mathematical difficulties can often obscure the underlying scientific ideas, which may be quite simple. Numerical methods can frequently deal with the mathematics in a way which is easier to understand, and so enable the student to concentrate on the physics" [Beare, 1986]. Fazarinc has built structurally accurate graphical simulations for delivering important ideas in physics. His intended use is for lectures, not for student problem solving as in ACSE, but his experiences are relevant. He found that about 2% of his software contains the

science while the rest is support [Fazarinc, 1988]. The objective in the ACSE environments is to cleanly split science away from support. With the science encapsulated in small, approachable chunks, students can manipulate problems both structurally and parametrically. They can ignore the overhead and the distraction of the support code.

Role of Programming Languages in Science Education

ACSE asserts and tests the proposition that computer programming is a form of communication that can be used to good effect in science education. Programming languages augment natural languages, graphics and mathematics as vehicles for scientific discourse. The extent to which this become widely adopted depends on the degree to which students and teachers are comfortable with programming. We plan to evaluate the effectiveness of ACSE in distinct populations. ACSE incorporates recent advances in graphics, animation, and software engineering into teaching environments. An important purpose of the ACSE project is to build quality teaching environments in which the role of programming can be fairly evaluated.

ACSE treads on dangerous ground here. The word *programming* conjures up images that can intimidate individuals who are not experienced with or enthusiastic about computer science. Statements like, "as a teacher of science I don't want my students spending their time programming," have been repeated to us often enough to convince us that we have a stigma to overcome. There are two misconceptions in this kind of thinking. One is that there are two well-defined groups of people: those who can program and those who can't. We believe that everyone can program in a system that facilitates it. Programming in ACSE is not fundamentally different than other forms of scientific writing.

ACSE tests whether programming can improve learning in a particular field of science. The programming aspects of ACSE are available to and will empower those who know how to program. For those who don't, ACSE provides other useful ways to learn, by virtue of its multimedia format. Early evidence shows that the non-programmer student learns with ACSE. But what if a teacher who doesn't know anything about programming is using ACSE? He or she will be able to take existing volumes and use them to teach. Furthermore, this teacher will be able to deal with and assist students who choose to use the programming aspects of ACSE. By extending the system with graphical programming and direct manipulation tools, our goal is to make the programming accessible to these ACSE clients – so accessible that they will begin programming without even knowing that is what they are doing. The kind of thinking required to write or modify an ACSE program is exactly the kind of thinking that is required to conduct a scientific experiment.

Science expressed in Pascal has an unusual sentence structure, and is interspersed with unusual syntax when compared with natural language version. ACSE hopes to unify this duality by presenting the Pascal in an alternate syntax that is more like natural language. Underneath will be the Pascal code in all of it's gory detail, but the student will decide whether it is necessary or desirable to peel back the layers to expose this.

Discovery Learning: Wet Labs and ACSE Labs

ACSE Volumes are employed as supplements to formal lectures, much as "wet labs" have traditionally been used to complement lecture material. ACSE does not replace a traditional laboratory experience – the two are complimentary. ACSE Volumes are used: to familiarize the students with unusually complex developmental processes that require extensive four-dimensional visualization; to expose students to important experimental systems and regimes that cannot practically be employed in an undergraduate laboratory, because of a need for sophisticated equipment, extensive training, excessive experiment durations, or dangerous reagents (such as radioactive probes); and to allow students to make direct changes to the underlying scientific model realized as a program, and to see the ramifications of those changes through the execution of the program.

6 The MacGnome Programming Environment

ACSE is bold to ask science students and teachers to use programming as a way to talk about science and to simulate experiments. In order for this to be a success, the programming effort must be supported as much as possible. The ACSE environment is built on the Object Pascal Genie. The Object Pascal Genie is a novice programming environment that represents state-of-the-art technology in this field. It was created by the MacGnome Project of Carnegie Mellon University's School of Computer Science. The MacGnome Project built, evaluated, and disseminated a family of novice programming environments, known as the Genie programming environments [Miller, 1989]. A MacGnome programming environment is an integrated system built around a structure editor [Chandhok, 1985]. It utilizes multiple views of the program to facilitate design [Garlan, 1987]. MacGnome environments lead in the automatic generation and display of data visualizations [Myers, 1988] and continue to break ground in the design, implementation, and use of compiler technology such as incremental static semantic analysis [Vorthmann, 1990].

MacGnome programming environments are built for the novice user and for the teacher of the novice. Features such as syntax directed editing and a fully integrated runtime system make it easy for the beginner to understand and write programs right away [Goldenson, 1988a, 1988b, 1989a, 1989b]. Incremental semantic checking gives the user

immediate feedback about errors and hints about how to correct them. Tools such as the Browser, Design View [Roberts, 1988], the Outline View, the runtime system with tracing, and the Call Stack's visual debugger provide excellent support for the teacher to clarify tricky concepts.

MacGnome encourages and supports the student in thinking about programs as data objects and operations on those objects, as do mature computer professionals. MacGnome environments make it easy to get started in programming and MacGnome starts the student with a view of programming that is consistent with modern wisdom of computer science and software engineering.

One well-received part of the Macgnome environment is the Call Stack (Figure 2). Any data that is on the call stack can be seen visually. The visual debugger clarifies data and data representation. It is a boon to understanding recursion, arrays, enumerated types, records, and linked structures.

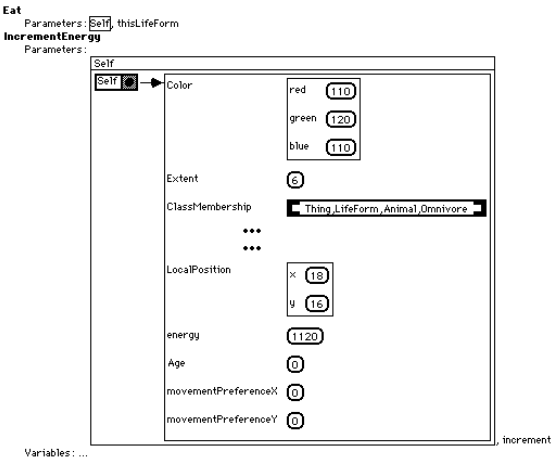


Figure 2: The Call Stack View

Another important view of programs provided by the MacGnome environments is a call graph representation known as the Design View (Figure 3). This view, like all program views, operates directly on the underlying program representation. Any changes made there are immediately reflected in every view. In addition to calls, the Design view shows variable declarations and formal parameters.

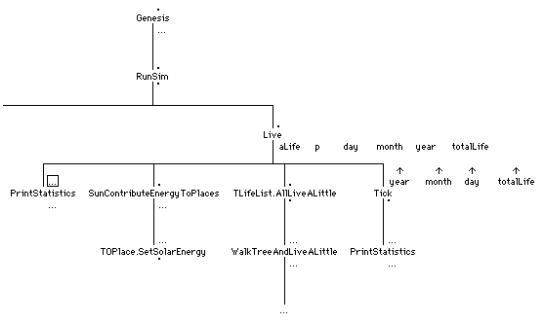


Figure 3: The Design View

7 Anecdotal Evaluation

A prototype of the ACSE environment has been in use for one year. It has been used in a Carnegie Mellon developmental biology class. Ordinarily the class is taken by undergraduate juniors and seniors. In collaboration with the biology professor, two volumes have been developed on topics in this curriculum. The volumes combined time-lapse videos, still electron micrographs, light microscopy, fluorescence light microscopy, animations, and a computer simulation with introductory text on the biological processes.

Evaluation has been informal. Questionnaire evaluation shows student and teacher evaluations to be positive. In particular, the simulation was viewed to be very helpful for visualizing three-dimensional processes. The greatest complaints were that the environment is slow and that navigation is difficult. These issues are addressed in the design for our next-generation system. One clear negative is that students have not made fundamental changes to the simulation. We believe this is due, in part, to the fact that students are given only about two hours with a Volume. However, there are important changes that can be made to make computational intellectual inquiry more inviting and easier to accomplish.

In addition to university-level implementations, ACSE has been introduced to secondary-level teachers at a Carnegie Mellon summer teaching institute. Teachers were given a brief presentation on the ACSE Project and given a hands-on opportunity to work with the Volumes individually. Evaluations of the session were positive. Teachers presented the ACSE Project group with important suggestions for future Volumes and asked to include ACSE in their own work for the coming school year. One local school, Pittsburgh's Taylor-Allderdice High School, has installed the ACSE environment into the classroom, but results are not known at this time.

8 Future Directions

There are two areas of improvement in the design for the new Volume interface. They are improved navigation through information and better support of the ACSE student programmer, in the form of tools for direct manipulation and graphical programming. In addition we plan to make greater efforts in the Volume to include carefully crafted, focused exercises and very explicit instructions that suggest fruitful areas of exploration in the simulation. Figure 4 show how an ACSE volume will be seen by the student when the environment is fully implemented.

The technical infrastructure of the MacGnome programming environment will continue to develop. We plan to expand the suite of ACSE Volumes, widen their use, and conduct formal and more thorough evaluation.

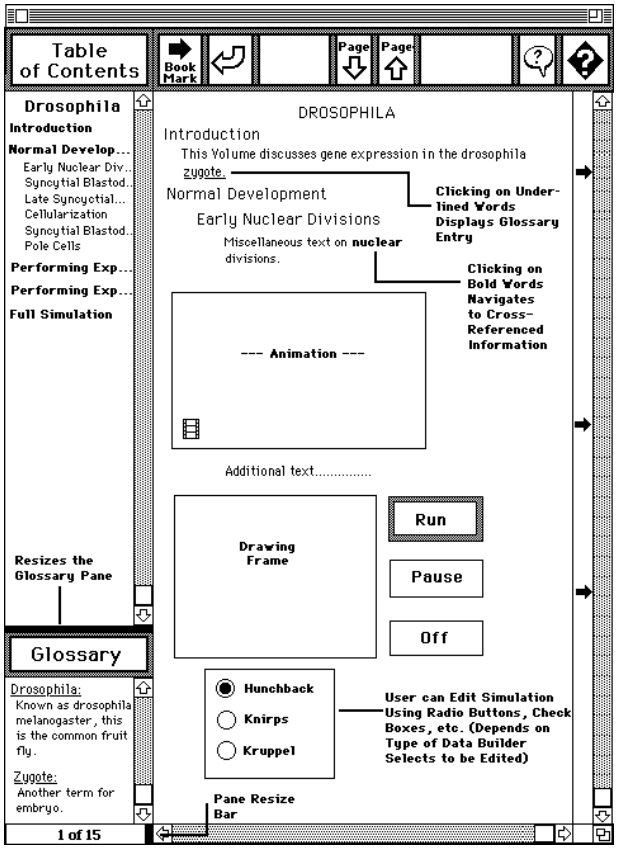


Figure 4: The Volume View Design

9 References

Beare, R. “Numerical methods in teaching physics”, *Proceedings of the International Conference on Courseware Design and Evaluation*, pp 255-260, Ramat Gan, Israel, April 1986.

Chandhok, R., et al. “Programming Environments Based on Structure Editing: The GNOME Approach.” *Proceedings of the 1985 National Computer Conference*, AFIPS, 1985.

DiSessa, A. “Boxer: A Reconstructible Computational Medium.” *Studying the Novice Programmer.* Hillsdale, NJ: L. Erlbaum Associates, 1989.

Fazarinc, Zvonko, “Overhead in Writing Physics Courseware”, *Proceedings of the Asia-Pacific Conference on Computer Education*, pp 428 - 431, October 1988, Shanghai China.

Garlan, David. “Views for Tools in Integrated Environments.” *Carnegie Mellon University Computer Science Department Technical Report CMU-CS-87-147* (Ph.D. thesis), 1987.

Goldenson, D. and Lewis, M. “Fine Tuning Selection Semantics in a Structure Editor Based Programming Environment: Some Experimental Results,” *ACM SIGCHI Bulletin*, 20, October 1988.

Goldenson, D. “Structure Editor Based Programming Environments: An Overview with Initial Results on Student Preference.” *Technology Across the Curriculum: Proceedings of the Texas Computer Education Association*, Dallas, 1988.

Goldenson, D., “The Impact of Structure Editing on Introductory Computer Science Education: The Results So Far,” *ACM SIGCSE Bulletin*, June 1989.

Goldenson, D., “Teaching Introductory Programming Methods Using Structure Editing: Some Empirical Results,” *Proceedings of 1989 National Educational Computing Conference*, Boston, June 1989.

Miller, P. L., and Chandhok, R. P. “The Design and Implementation of the Pascal GENIE.” *Proceedings of the 1989 ACM Computer Science Conference*, Louisville, KY, February 1989.

Myers, B. A., Chandhok, R. P., and Sareen, A. “Automatic Data Visualization for Novice Pascal Programmers.” *Proceedings of 1988 IEEE Workshop on Visual Language*, Pittsburgh, PA, October 1988.

Papert, S., and Solomon, C. “Teaching Children Thinking.” *Studying the Novice Programmer.* Hillsdale, NJ: L. Erlbaum Associates, 1989.

Roberts, J., Pane, J., Stehlik, M., and Carrasquel, J. “The Design View: A Design Oriented High Level Visual Programming Environment.” *Proceedings of 1988 IEEE Workshop on Visual Language*, Pittsburgh, PA, October 1988.

Vorthmann, S. “Syntax-Directed Editor Support for Incremental Consistency Maintenance.” *Georgia Institute of Technology Technical Report GIT-ICS-90/03* (Ph.D. thesis), June 1990.