Repenning, A., Ioannidou, A. & Ambach, J. *Learn to Communicate and Communicate to Learn.*
Journal of Interactive Media in Education, 98 (7)
[www-jime.open.ac.uk/98/7]

Published
19 Oct. 1998

ISSN:1365-893X

Learn to Communicate and Communicate to Learn

Alexander Repenning, Andri Ioannidou, James Ambach

Abstract: Thinking of a computer as an educational tool emphasizes a solitary interaction between the learner and the computer. A tool is something that is applied to an object in order to change it, and this usually implies a single user working on a single object or project. While tools are an important aspect of an educational experience, the tool metaphor can isolate individual learners from each other and from their teachers instead of cultivating a sense of educational community. Reconceptualizing the computer as a constructionist medium increases the computer's educational value, by allowing the development and support of communities of learners. As a medium, the computer is viewed as a collection of tools and capabilities that are used to communicate. This reconceptualization leads to a new class of computer applications that places constructionist activities within a more social context in which computers simultaneously provide opportunities for learning how to communicate as well as for enabling communication to enhance the learning experience. Computers are a particularly interesting educational medium, because not only do they allow communication via more traditional media (e.g. text, pictures and video), but they also enable the communication of ideas through the creation and sharing of computational objects (e.g. agents, simulations and analysis tools). As these computational objects become easier to create, share and combine, the educational opportunities afforded by the computer become more viable and effective.

This article describes the AgentSheets system and its use as a constructionist medium in K-12 and university educational settings. Specifically, AgentSheets supports the social creation of SimCity®–like interactive simulations by providing an environment for the definition and sharing of computational components, called agents, through the World Wide Web.

Keywords: End-user programming, computers and education, agents, World Wide Web, communities of learners, educational medium, interactive simulations, games, spreadsheets, visual programming, Java, interactivity, richness, accessibility, Educational Object Economy

Interactive Demonstrations: Agentsheets, Visual AgenTalk and the associated Behaviour Exchange are at <http://www.cs.colorado.edu/~l3d/systems/agentsheets>. The FishTank simulation is embedded as a Java applet (requiring a Java-aware Web browser), and two video clips can be viewed of LEGOsheet applications in use (requiring a QuickTime player).

Alexander Repenning, Andri Ioannidou & James Ambach
Department of Computer Science & Center for LifeLong Learning and Design, University of Colorado, Boulder, CO 80309-0430. U.S.A.
http://www.cs.colorado.edu/~ralex   http://www.cs.colorado.edu/~ambach
{ralex, andri, ambach}@cs.colorado.edu

1. Introduction

It is often unclear exactly what value computers are adding to education. Do they or could they enhance existing teaching practices by making teaching more effective and more efficient? Can they lead to new educational paradigms in which the traditional roles of teacher and student change more fundamentally? Are computers used best to convey information in more instructionist settings or are they better suited as environments and microworlds engaging students in more active roles of exploration? Common to these questions is the metaphor of computer as tool: a tool to write a report, a tool to compute the solution to an equation, a tool to draw a diagram, a tool to create a foldable origami structure (Eisenberg and Fischer, 1994), etc. As a tool, the role of the computer is to help the learner, or more generally the user, create something, including the solution to a problem. Even seemingly mundane assistance, such as a spelling checker in a word processor, makes the computer a better tool for doing a job more effectively and efficiently.

The notion of computer as tool, while useful, is too narrow because it limits the scope of observation to a solitary interaction between learners and machines, leaving out important social issues. Instead, this article suggests that the use of computers as a constructionist medium contributes to the way people think about things and communicate with each other. As an educational medium, the computer allows learners to communicate with each other all over the world via textual email or increasingly via multimedia home pages on the World Wide Web (the Web). Enhancing computer tools with the ability to easily share and exchange computational artifacts within a potentially large and distributed community not only teaches individuals to communicate their ideas in unique ways to people from varied backgrounds and experience—for instance, a third-grader can build a computer agent that behaves like the juggler he saw performing on the street—but also allows the individual to tap into a large and distributed collection of expertise that was not readily available before.

Viewed as a medium, the computer can enhance the very nature of how and what we communicate. It facilitates intersubjectivity (Figure 1), that is communication between learners, and intrasubjectivity (Figure 2), that is communication between the learner and him/herself (Vygotsky and Kozulin, 1996; Vygotsky and Vygotsky, 1980). Communication, in turn, is an essential part of learning. Teachers must communicate ideas and facts to students. Students communicate with other students in social learning situations to develop answers to questions, to reflect on their understanding, and to concretize their ideas.

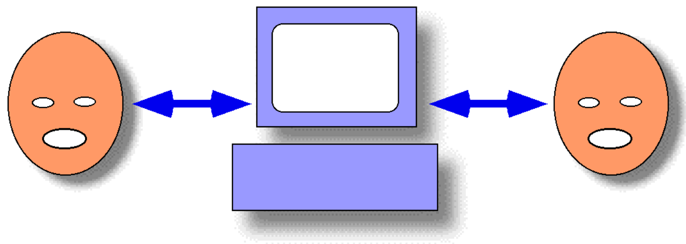


Figure 1: *Intersubjectivity through the computer: the medium enables communication between different learners.*

Like a piece of paper, the computer is not only useful as an information conduit between different people, but can also serve an essential role as a medium between the learner and him/herself. For instance, a piece of paper can extend cognition simply by serving as an external memory (Norman, 1986). Computational design environments (Fischer, 1994) enable learners to create situations which "talk back," pointing out sub-optimal design decisions.

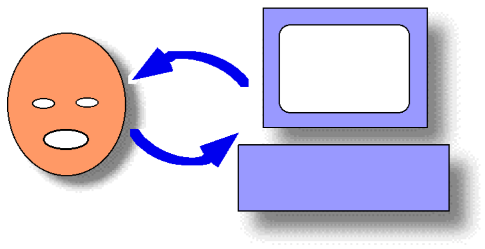


Figure 2: *Intrasubjectivity through the computer: the medium enables communication between the learner and him/herself.*

The myriad capabilities, including the ability to display multimedia, to support interaction, and to facilitate multiple interface modalities (i.e. handwriting recognition, speech recognition, different input and display devices), render the computer an extremely flexible medium that not only conducts information, but can actively process it. Unlike paper, overhead slides, or video tapes, computers can process formal information, such as computer programs and application documents, as well as manipulate informal information such as text and pictures.

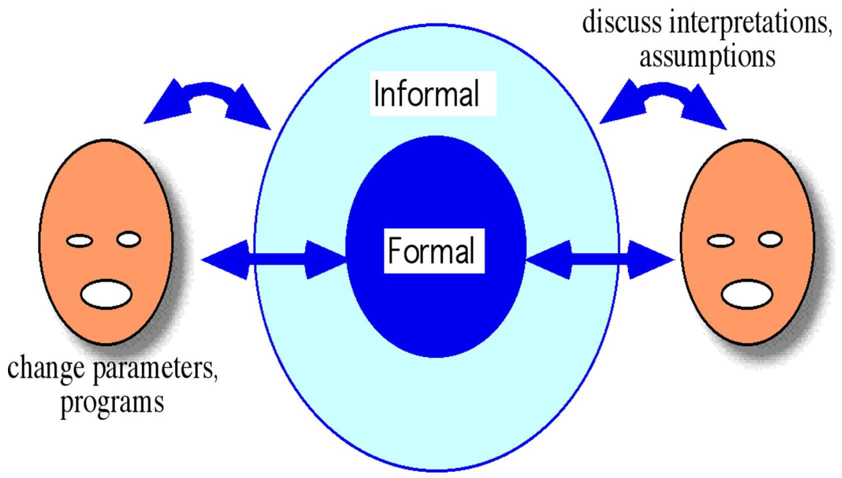

discuss interpretations, assumptions
Informal
Formal
change parameters, programs

Figure 3: *Computer as Interactive Communication Medium*

The ability to combine formal information (the products generated by the tool) with informal information (what people use to communicate directly with each other) enables the computer to become a much more active component of the communication. The trick to having the computer fulfill its potential as a communication medium is to not only simplify and enhance the tools that are a part of that medium, but to ensure that the products of those tools, simulations for example , can then be used to communicate and share ideas.

The notion of interactive simulations is quickly gaining in importance as a means to explore and communicate complex ideas. Learning and teaching requires processing and communication of information using media such as papers, videos or more recently Web pages. While the Web has the potential to feature all kinds of content it is currently typically used to capture static content such as text and pictures. Interactivity in the context of the Web is often reduced to navigation of hyper spaces. Simulations go beyond this kind of interactivity by allowing users to experience complex, dynamic relationships through role playing and what-if games. Moreover, simulations are particularly effective when they are built by the learner. According to constructionist learning theory (Papert and Harel, 1993) the design of artifacts, such as simulations, is a powerful means of constructing new knowledge. For instance, a good means to learn about food webs in life science is to have kids collaboratively design animal species that can interact with each other inside a simulation. The process of designing animals helps to get new insights about animals. Finally, a simulation running on a computer can illustrate consequences of a design that would have been very hard to think through in the head.

The remaining sections outline the salient characteristics of different media as they are used in this article, introduce the AgentSheets system, explain its use in creating interactive simulations that are shared via Web pages and describes our experiences in a number of projects that employ AgentSheets and AgentSheets-related systems as educational media.

2. The Media Cube

Out of the many dimensions one could select to classify media, we have chosen the following: media *richness, interactivity* and *accessibility* to ground the following discussion. Mok defined a subset of this framework relating the richness and interactivity dimensions of media (Mok, 1996). The cube (Figure 4) formed by these dimensions is used to classify different media and to help understand the value added (or lost) by moving to different spaces within the cube.

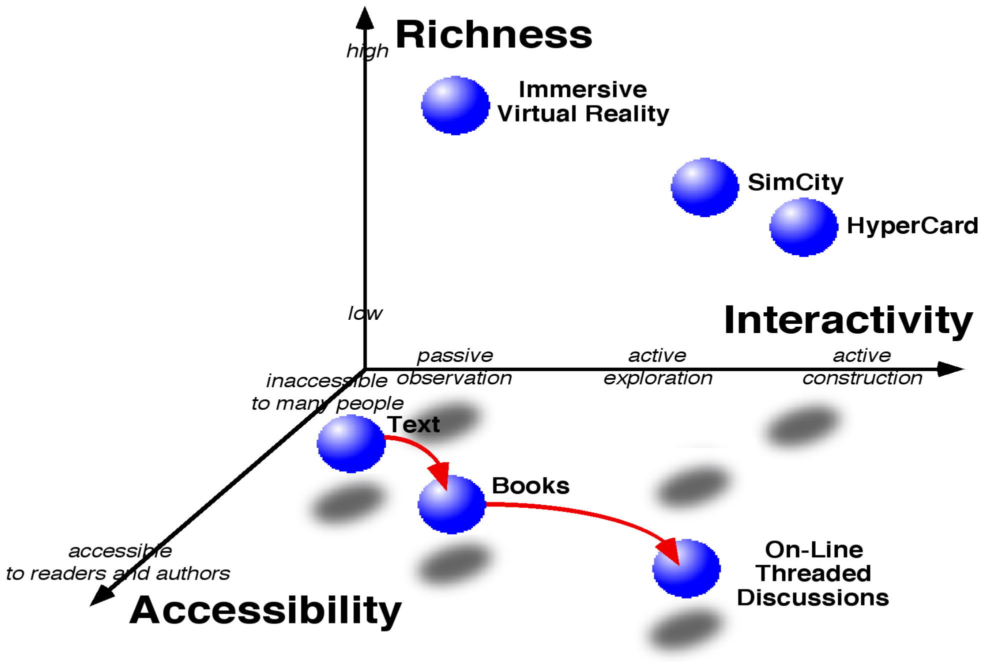


Figure 4: *The Media Cube*

Even though learning is implicit in these media dimensions, we do not explicitly discuss the effects they have on knowledge construction, but focus mainly on the effects these dimensions have on information communication.

The media cube framework provoked some interesting review discussion which is linked to the online article.[†] *(Editor)*

2.1 Interactivity

The interactivity dimension describes a continuum from passive observation to active exploration to active construction. If a medium provides a representation of something in the real world, then that medium's interactivity has to do with the degree to which the user of the medium can participate within that representation (Laurel, 1993). Movies and books are media best suited for passive observation because the user can do little to participate within the representation. Interactive video or Web pages provide the user with more active exploration of the representation by allowing choice points or the ability to follow links at the user's will. Interactive simulations like SimCity or multimedia authoring environments like Claris HyperCard® allow active construction within the representation defined by the medium. In SimCity it is possible to create new buildings and roads, and in HyperCard its possible to build new buttons, fields, cards and stacks.

2.2 Richness

The richness dimension describes the degree to which elements from the real world are incorporated or represented within the medium. The claim is that the physical world is intrinsically richer than any medium, because a medium only contains representations of real world objects. However, it is possible to think of certain media as being richer than others, especially if the real-world objects which are represented are done so with high fidelity. Fidelity describes how close the media representation resembles the real-world object. Thus, text would be considered low in the richness dimension, because the textual representation of a tree, for instance, bears little resemblance to a real tree. A full-immersion virtual reality environment would be considered a richer media because a tree represented in this environment will appear more tree-like. It is important to note that a medium's position within a dimension does not represent a judgment of that medium. We are not saying that immersive virtual reality is a better medium than text. We are merely describing a way to classify different media in order to ground our discussion.

[†] *Discussion on the media framework <http://www-jime.open.ac.uk/Reviews/get/repenning-98-7-reviews/7.html>*

2.3 Accessibility

The final dimension, accessibility, describes how available the medium is and how easy it is not only to read what is created by others, but also to author new content. There are two ways to increase the accessibility of a medium: the first is to allow a medium to be accessible by users in a different time and place from the original author (Table 1).

	Same Place	**Different Place**
Same Time	conversation	TV radio
Different Time	refrigerator notes traffic signs	email World Wide Web

Table 1: *Media Time and Place*

A conversation, for instance, is only accessible to those who are present during the conversation. A Web page, however, can be browsed by any of the millions of people with access to the Web, at any time that is convenient to the reader. Thus, Web pages are more accessible than a conversation.

The other way to increase a medium's accessibility is to move from a single-casting model to a multi-casting model (Table 2).

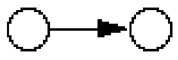
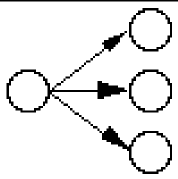
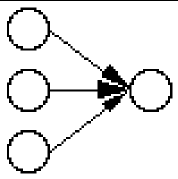
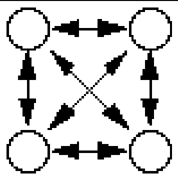
	To One	**To Many**
From One	 *Single Casting*	 *Broadcasting*
From Many		 *Multi-casting*

Table 2: *Media Casting*

Watching a video tape of a lecture by yourself is less accessible than being part of a seminar where teachers and students are free to exchange ideas with each other. In the case of the video tape, the learner has no control over the content being delivered, and can only absorb it. In the seminar, the content is shaped by all of the participants. The written word increased the accessibility of the spoken word by adding persistence. Once an idea was written, it was no longer ephemeral and could be studied and pondered by anyone with access to the medium. The advent of the printing press eventually allowed the mass production of books which increased the accessibility of the written word. Through mass-produced books, ideas expressed in written words became available to a large number of people increasing the variation and interpretation of the ideas originally presented in the text. Computational media such as threaded discussions or web sites with feedback mechanisms again increase the accessibility of text, by not only providing the persistence of the written word and potential audiences of millions of people (accessing it at different time, from different places), but by also providing the ability to capture immediate feedback from anyone accessing the medium (multi-casting). An Internet user can start a text-based discussion by stating a position in a newsgroup. That position can be read by millions of people all over the world at any time, and they can respond either in agreement or disagreement describing why they feel that way. The ability to solicit feedback from potentially large and distributed groups of users within the same medium that the original ideas were represented is a crucial aspect to highly accessible media and is something that is unique to computational media.

3. AgentSheets: A Communication Medium

AgentSheets [1] (Repenning, 1994; Repenning and Sumner, 1995) is a software environment for creating SimCity-like simulations, games and domain-oriented visual programming languages. AgentSheets applications include a collection of autonomous computational processes, called agents, that are comprised of a look, their on-screen representation, and a programmed behavior. Agents can be programmed to both perceive and act upon their environment (Figure 5).

Agents are placed together in a grid, or sheet, where they interact with each other. Agents can perceive mouse clicks, sound and keyboard input, the existence of other agents, attribute values, messages and even Web page content among other things. They can act by moving, changing their appearance, playing sounds, computing values, “speaking” text through voice synthesis, sending messages to other agents, opening URLs and performing several other possible actions.

[1] *The AgentSheets System: <http://www.cs.colorado.edu/~l3d/systems/agentsheets/>*

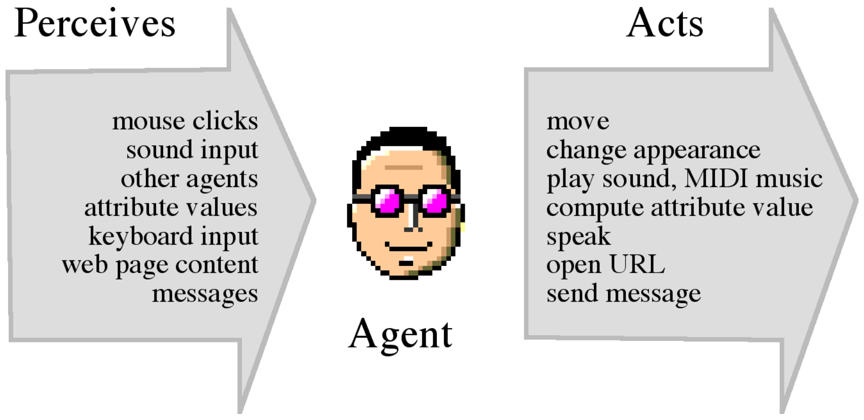


Figure 5: *Agents Perceive and Act*

End-users of AgentSheets applications create simulations and games by selecting agents from a gallery (Figure 6 (1)) and placing them into a worksheet (Figure 6 (2)). Once a worksheet is populated with agents, users can start the simulation by clicking on the Run button. Starting a simulation activates the agents' behavior and users can continue to interact with the agents while the simulation is running (Repenning and Sumner, 1994). In Figure 6, the end-user is designing a traffic simulation. Roads, cars, tracks, trains and stoppers are all agents, and when the user starts the simulation, the agents will act according to their programmed behavior: cars will move on roads and stop when the Stopper is down, and trains will move on tracks. Agents can be stacked on top of each other in a grid space. For instance, car agents can be on top of road agents, and train agents can be on top of track agents. Some agents have no appreciable behavior, but exist as background agents. In Figure 6, road and track agents do not really do anything, but car and train agents perceive them, and follow them.

AgentSheets also provides a collection of editors that allow users to create new AgentSheets applications and agents. There are depiction editors and operations that are used to create an agent's look, and there are programming editors and operations that are used to define an agent's behavior.

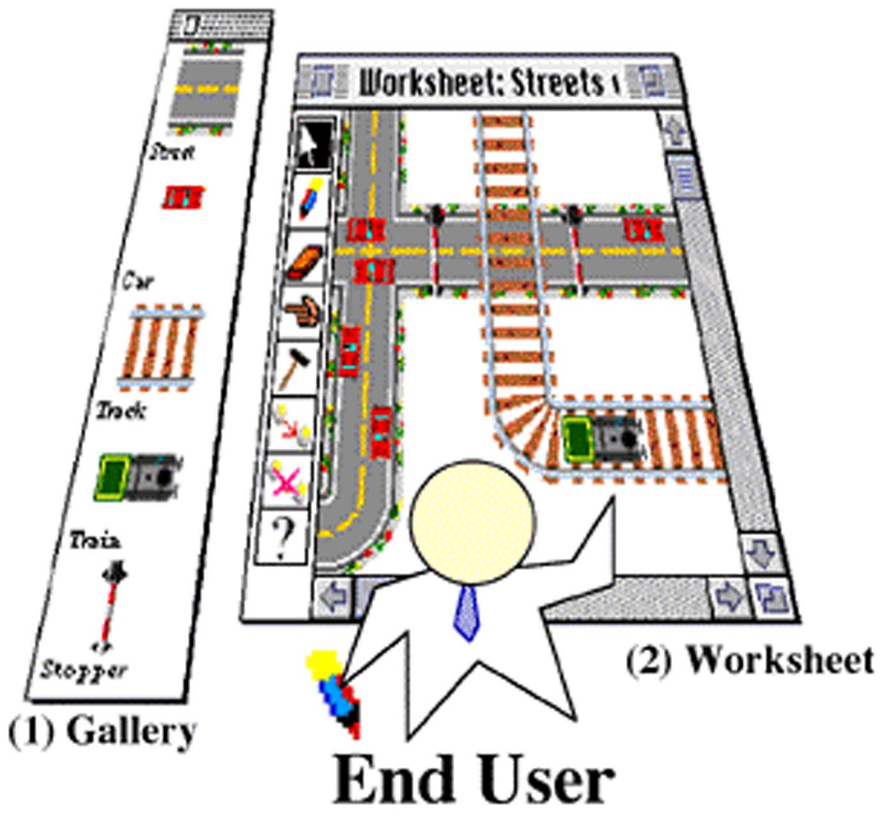


Figure 6: *AgentSheets and End-Users*

When AgentSheets was first introduced in 1993 (Repenning, 1993) it was thought of as a tool for creating dynamic simulations and visual programming environments. Thinking of the original AgentSheets in terms of the media cube (Figure 4) would place it somewhere just to the right of SimCity, and just above HyperCard. Using the AgentSheets depiction editor and textual programming language, AgenTalk, allowed the creation of fairly rich agents that could look, behave and even sound like their real-life counterparts. Furthermore, since AgentSheets provided programming tools in the form of editors and a textual programming language, the simulations allowed both active exploration (users are free to build their own simulations with the given agents) and active construction (users could define new agents), as long as the user knew AgenTalk. However, like SimCity and HyperCard, as a medium AgentSheets was inaccessible. Not only was it not easy to share AgentSheets simulations with lots of other people, unless they had access to your computer or you could give them a floppy, it was considerably more difficult to define new agents and applications than it was to use existing ones.

Since its inception, AgentSheets has been used in a number of different settings on a number of different projects and has evolved from a tool to a richer, more interactive and accessible medium. In doing so, AgentSheets has become more educationally viable and effective. The next section describes uses of AgentSheets in various educational settings.

3.1 AgentSheets in Use in Educational Settings

What value are computers adding to education? Do they or could they enhance existing teaching practices by making teaching more effective and more efficient? Can they lead to new educational paradigms in which the traditional roles of teacher and student change more fundamentally? In an attempt to find answers to these questions and test the applicability of the AgentSheets simulation technology to learning, we have used the system in educational settings of various disciplines, age/skill levels, and demographics. The idea of using simulations as constructionist media and having people learn and communicate by building simulations is powerful. It is also consistent with the educational technology strategy put forward by the President's Committee of Advisors on Science and Technology.

> *Within the framework of this newer paradigm [constructivism], technology is viewed not as tool for improving the efficiency of traditional instructional methods based largely on the unidirectional transmission of isolated facts and skills from teachers to students, but as one element of a new constructivist approach in which teachers concentrate instead on helping their students to actively construct their own knowledge bases and skill sets.* [2]

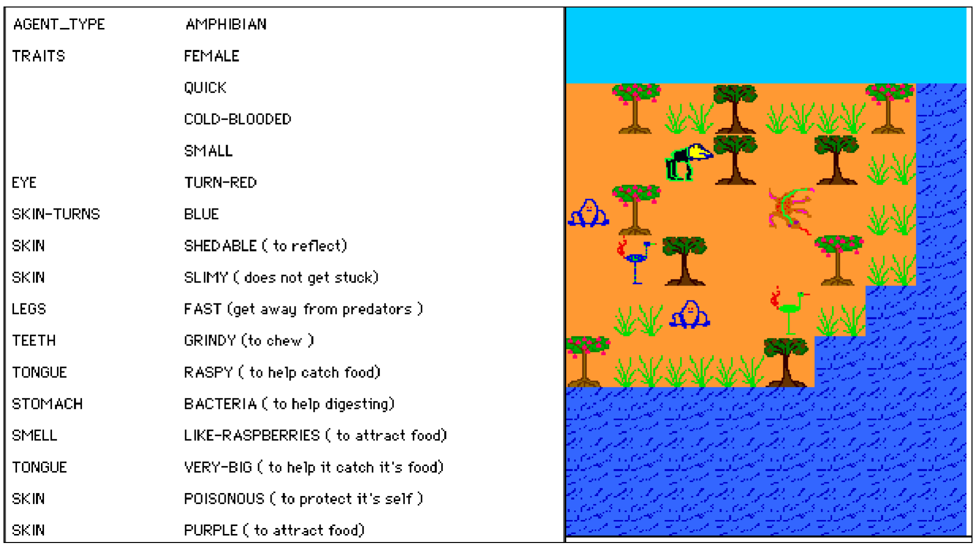
To illustrate the uses of the AgentSheets technology in educational settings we briefly describe relevant projects that took place in elementary, middle and high schools both locally and internationally.

3.1.1. *Elementary School*

AgentSheets was successfully used to support education at the level of elementary schools. Over the last 3 years Clayton Lewis has headed a project called Science Theater (sTc). The sTc objective was to explore the design and use of simulations by students as means to learn about complex topics. Results after the first two years were largely negative (Rader, Brand and Lewis, 1997; Rader, Cherry, Brand, Repenning and Lewis, 1998): many students experienced difficulties building relevant simulations successfully. Even students who did successfully

[2] *Report to the President on the Use of Technology to Strengthen K-12 Education in the United States by the President's Committee of Advisors on Science and Technology, March 1997. <http://www.whitehouse.gov/WH/EOP/OSTP/NSTC/PCAST/k-12ed.html>*

complete simulations by end large only implemented their preconceived notions of issues. The process of building a simulations did not help them break out of false mindsets. Based on these negative results after the first two years the project replaced the software used - Apple's Cocoa environment - with AgentSheets. The use of AgentSheets dramatically changed the picture allowing kids to successfully build more complex and complete simulations that overcame the previously mentioned problems. On key factor is Agentsheets extensible architecture that allowed the researchers to scaffold learning activities by building a programming language tailored to the domain of life science. Based on this success Prof. Lewis decided to continue exploring the notion of simulation from a more systemic perspective.



AGENT_TYPE	AMPHIBIAN
TRAITS	FEMALE
	QUICK
	COLD-BLOODED
	SMALL
EYE	TURN-RED
SKIN-TURNS	BLUE
SKIN	SHEDABLE (to reflect)
SKIN	SLIMY (does not get stuck)
LEGS	FAST (get away from predators)
TEETH	GRINDY (to chew)
TONGUE	RASPY (to help catch food)
STOMACH	BACTERIA (to help digesting)
SMELL	LIKE-RASPBERRIES (to attract food)
TONGUE	VERY-BIG (to help it catch it's food)
SKIN	POISONOUS (to protect it's self)
SKIN	PURPLE (to attract food)

Figure 7: *EcoWorld simulation (right) with its defining parameters (left).*

The life science curriculum developed consists of EcoWorld simulations (Figure 7) in which student collaboratively designed their own agents representing plants and animals. The goal of this activity was to have groups of students create sets of animals that add up to a complete sustainable ecosystem. Once the animals were created, they were let loose in a variety of worlds. Jointly the group members observed the stability and other characteristics of their creations. The groups also completed activity sheets capturing conditions and outcomes of experiments.

3.1.2 Middle School

Agentsheets was also used in extra-curricular settings, such as the Friday afternoon Computer Club at Centennial Middle School [3] . In this context, students volunteered to work on

[3] *Centennial Middle School <http://bvsd.k12.co.us/cent/>*

AgentSheets-related projects, one of which involved creating a scavenger-hunt type of game simulating the Pearl Street Mall, a popular area in Boulder. A tourist, controlled by the user, is walking on Pearl Street trying to buy things from different stores and in the process he is interacting with other characters in the game, such as pickpockets, jugglers, and bikers (Figure 8). This gave the students the possibility to express themselves and their immediate environment in a simulation game. In the process of creating their simulation, they made extensive use of the Behavior Exchange, an Agentsheets extension that allows trading of simulation components by dragging and dropping agents from a Web site and has helped to further facilitate collaborative simulation construction over the Web (see Section 4.3.1).

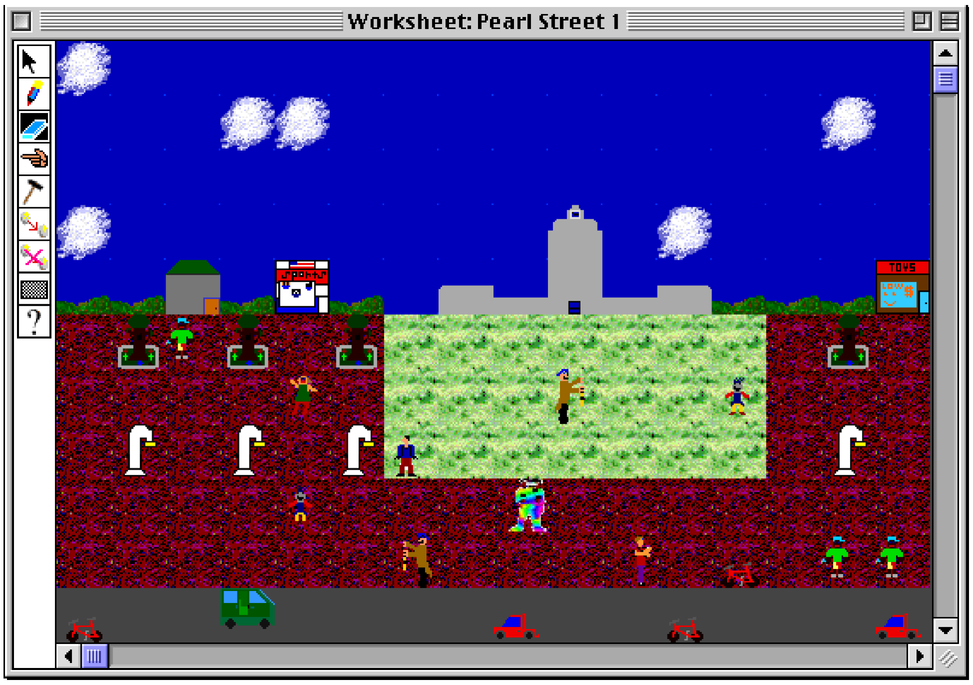


Figure 8: *The Pearl Street Project with a tourist going around the mall trying to buy things, while coming across street performers, pick-pockets and other pedestrians.*

Using agents created by fellow students inspired the students who were apprehensive about programming to move from being consumers of information to becoming producers of rich and expressive computational artifacts that are sharable over the Web (Repenning, Rausch, Ioannidou and Phillips, 1998). Consuming interesting agents motivated students to learn more about producing behavior and to contribute their own ideas to the Behavior Exchange. These students soon reached a high level of system understanding and were able to create interesting dynamic behavior.

Curiosity and a desire to 'join in on the fun', plays a major role in motivating end-users to move from being consumers to becoming producers on the Web. During the creation of the Pearl Street Simulation, talking about the behaving agents was one of the main topics in the students' conversations. For students who liked to draw the agents' looks but had not yet tried to program their behavior, listening to the conversations of the behavior-makers resulted in an increasing desire to create their own behaving agents. They wanted to engage in the discussions, to "become part of it", as they put it.

Presenting their work to a Web audience was also a source of motivation for producing sharable computational artifacts. Students were proud to present their creation to a wide audience and after getting positive feedback for their initial creation, they were motivated to continue building behaving agents. We claim that this sense of achievement is stronger with constructing behaving objects than it is with displaying static things on the Web such as text and pictures.

3.1.3. High School

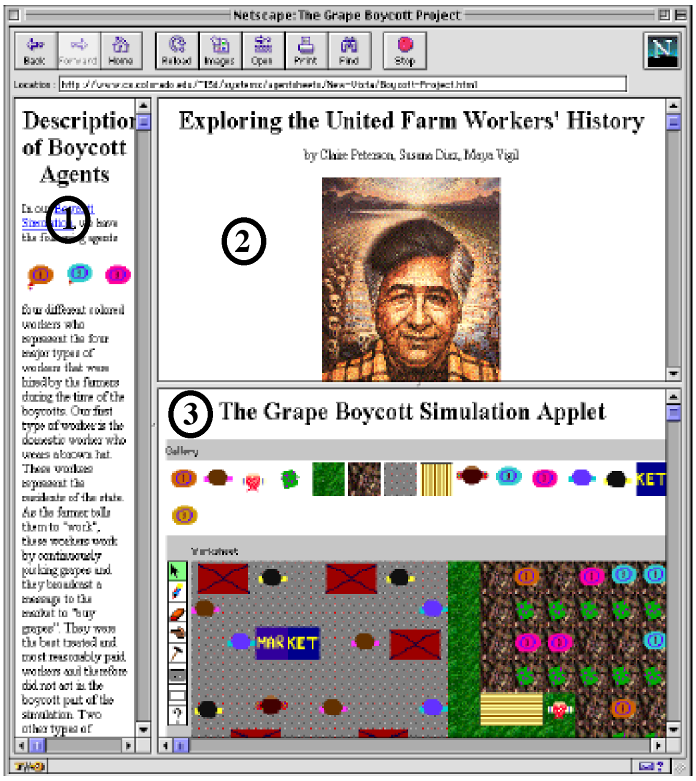
It is usually perceived that math and science classes are the most appropriate for introducing technology in general and simulation technology in particular. Agentsheets was used for the first time in a social sciences setting when John Zola, a teacher at New Vista High School was interested in using simulations in his "Protest and Reform" history class. In his class, students had the opportunity to study protest movements throughout U.S. history (e.g. the Civil Rights movement and the anti-Vietnam war movement) and to learn about theories of protest and social change.

After some initial efforts to create a content background and familiarity with the technology, students had the choice to develop a computer simulation, create a posterboard or write a paper as a final project for the course. Two groups of young women, initially intimidated by technology, chose the simulation option. One group researched and created a simulation based on the Cesar Chavez led Grape Boycotts and the United Farm Workers struggle (Figure 9), while the other group created a simulation on what happens when a large-scale peaceful protest becomes violent and encounters police resistance. Each group not only developed a simulation, but also crafted Web pages to display their work and extend the reach of their research. Despite the initial apprehension with technology, the projects exceeded the students' expectations on the excitement they experienced and the learning that occurred in the process, the quality of the result, and the appraisal their work has gotten since its publication on the Web. Both projects are accessible at the New Vista Project webpage. [4]

When they compared the experiences in creating a simulation to previous experiences in creating

[4] *The New Vista Projects for the "Protest and Reform" class <http://www.cs.colorado.edu/~l3d/systems/agentsheets/New-Vista/>*

a posterboard, students realized that the former requires a much deeper understanding of the topic than the latter. Creating a posterboard does not necessarily give an opportunity to students to explore a topic in depth. Students admit to becoming experts in cutting and pasting information from encyclopedias, books or Web pages without actually learning the material they are writing about. When building simulations, on the other hand, students do not have the option of mindless cutting and pasting. Creating a good and accurate simulation requires that the students develop a deep understanding of the subject they are simulating. In the Boycott project, for example, the students had to learn the history behind the Grape Boycott in order to create their simulation. Moreover they needed to have a level of understanding of this historical content well beyond that of regular type projects (e.g. a posterboard). The same held true with the Protest project. Creating the behaviors of the individual agents was one thing, but programming the interactions among the agents required a deeper understanding of how protests worked in the past and knowledge of human nature under those circumstances.



Description of Boycott Agents

In our Boycott Simulation, we have the following agents

four different colored workers who represent the four major types of workers that were hired by the farmers during the time of the boycotts. Our first type of worker is the domestic worker who wears a brown hat. These workers represent the residents of the state. As the farmer tells them to "work", these workers work by continuously picking grapes and they broadcast a message to the market to "buy grapes". They were the best treated and most reasonably paid workers and therefore did not act in the boycott part of the simulation. Two other types of

Exploring the United Farm Workers' History

by Claire Peterson, Susana Diaz, Maya Vigil

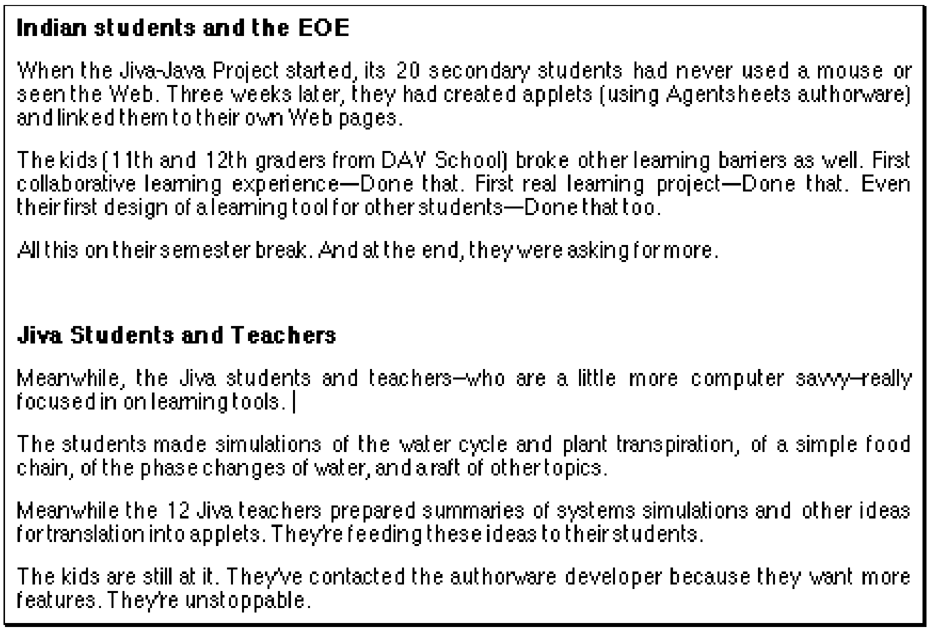
The Grape Boycott Simulation Applet

Figure 9: *The Boycott project Web page includes (1) descriptions of the agents, (2) historical background and related links, and (3) the Boycott Simulation applet.*

3.1.3. 3rd World Schools

Evidence of the general applicability of AgentSheets to a variety of demographics comes from India where a group of teachers and students picked up AgentSheets. In just 3 weeks the students advanced from basic mousing skills to creating complete interactive simulations as Java applets. And all of this was done without our help. Only after creating all their work we got contacted by their teacher pointing us to their Web page (Figure 10).

While the students initially learned about computers (mousing skills, typing, Web browsing) they moved on quickly to learn about curriculum-related materials with computers by building simulations in AgentSheets. Simulations that they built and published on the Web (Figures 11 and 12) include airspace interactions, forest ecosystem food chain, color-mixing activity based on an Indian holiday celebration, phase changes of water, water cycle and plant transpiration, pond-life food chains.



Indian students and the EOE

When the Jiva-Java Project started, its 20 secondary students had never used a mouse or seen the Web. Three weeks later, they had created applets (using Agentsheets authorware) and linked them to their own Web pages.

The kids (11th and 12th graders from DAV School) broke other learning barriers as well. First collaborative learning experience—Done that. First real learning project—Done that. Even their first design of a learning tool for other students—Done that too.

All this on their semester break. And at the end, they were asking for more.

Jiva Students and Teachers

Meanwhile, the Jiva students and teachers–who are a little more computer savvy–really focused in on learning tools. |

The students made simulations of the water cycle and plant transpiration, of a simple food chain, of the phase changes of water, and a raft of other topics.

Meanwhile the 12 Jiva teachers prepared summaries of systems simulations and other ideas for translation into applets. They're feeding these ideas to their students.

The kids are still at it. They've contacted the authorware developer because they want more features. They're unstoppable.

Figure10: *Excerpts from the website [5] created by teachers and students in India at the JIVA institute. They have used AgentSheets to create educational Java applets.*

[5] *The JIVA-Java project in India <http://surya.jiva.ac.in/jivajava/>*

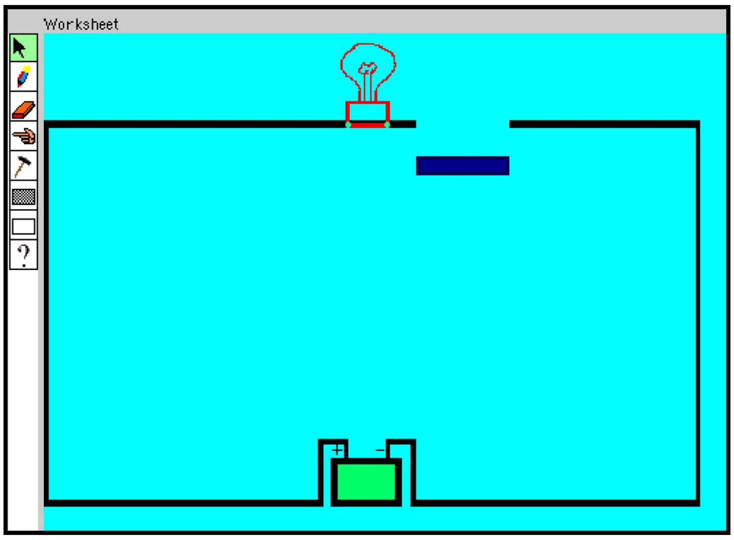


Figure 11: *The conductivity of different materials such as wood, iron, and copper can be explored by making them part of an active circuit with a lamp serving as indicator for the flow of current.*

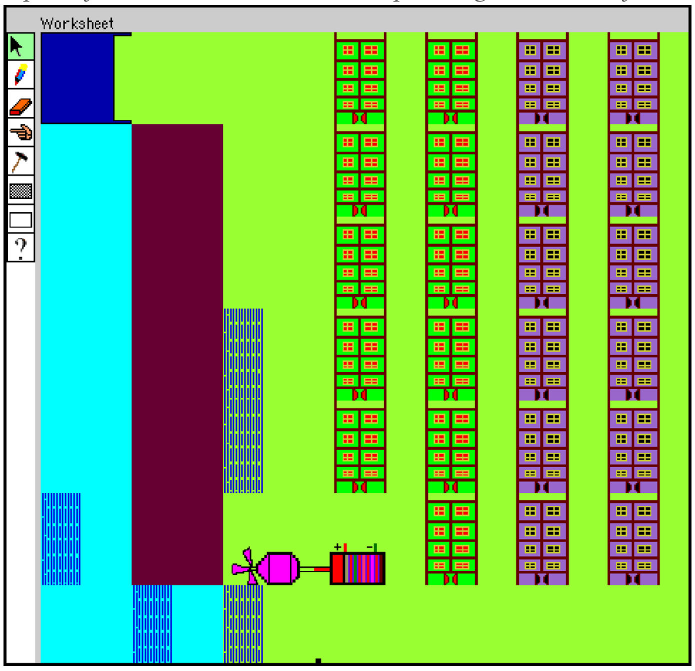


Figure 12: *Hydro Electricity Simulation. Water flow propels a generator which in turn powers a city*

The approach taken by the teachers at the JIVA institute is highly compatible with the first out of 6 principal recommendations by the President's Committee Of Advisors On Science And Technology [6] :

> *Focus on learning with technology, not about technology. Although both are worthy of attention, it is important to distinguish between technology as a subject area and the use of technology to facilitate learning about any subject area. While computer-related skills will unquestionably be quite important in the twenty-first century, and while such skills are clearly best taught through the actual use of computers, it is important that technology be integrated throughout the K-12 curriculum, and not simply used to impart technology related knowledge and skills. Although universal technological literacy is a laudable national goal, the Panel believes the Administration should work toward the use of computing and networking technologies to improve the quality of education in all subject areas.*

4. AgentSheets in the Media Cube

As a communication medium, the AgentSheets system can be explained along the dimensions of the media cube (Section 2): interactivity, richness and accessibility. In this presentation of AgentSheets, several features of the system are described. These include the language employed by AgentSheets, namely Visual AgenTalk, capabilities that allow it to enrich simulations with real-time data posted on the Web or from physical devices such as the MIT Programmable Brick, the Behavior Exchange Web-based repository of agents that allows users to exchange their creations, and the Ristretto agent-to-java-to-byte-code generator that allows end-users to publish their interactive simulations on Web pages.

4.1 Interactivity

As a constructionist (Papert and Harel, 1993) educational medium, AgentSheets has always relied upon a high degree of interactivity for any kind of user, from end-users to simulation and application designers. AgentSheets has implemented a layered approach, where different tools and editors are available, depending on the skills and desires of the particular user. End-users have always had the ability to either run pre-constructed simulations or build their own using the agents that exist in the Gallery (Figure 6 (1)). In the original AgentSheets, if a user was interested in creating their own agent, they had to define the agent behavior using a textual programming language called *AgenTalk*. While the *AgenTalk* tools and editors were readily

6 *Report to the President on the Use of Technology to Strengthen K-12 Education in the United States by the President's Committee of Advisors on Science and Technology, March 1997.*
<http://www.whitehouse.gov/WH/EOP/OSTP/NSTC/PCAST/k-12ed.html>

available, it is obvious that in order to truly increase the interactivity of AgentSheets for end-users, new approaches to programming would have to be employed. After experimenting with graphical-rewrite rules and programming-by-analogy (Repenning, 1995), a new end-user programming tool called *Visual AgenTalk* has been added to the AgentSheets environment to provide both a low threshold and a high ceiling for end-users wishing to define their own agents and agent behavior.

4.1.1 Visual AgenTalk: Allowing Active Construction

Visual AgenTalk (Repenning and Ambach, 1996; Repenning and Ambach, 1997) represents a new approach to end-user programming called tactile programming. In tactile programming, programs and program primitives are imbued with an interface that allows not only the setting of parameters, but also enables the program or program fragment to be dragged and dropped within different contexts for different reasons. Visual AgenTalk was designed to promote program comprehension, composition and sharing.

4.1.1.1 Program Comprehension

Programming primitives in Visual AgenTalk are called *commands* (Figure 13). Commands are small interactive forms that can be manipulated by end-users. Commands can be both general purpose as well as domain specific, and consist of a name and an arbitrary number of parameters. End-users set the values of the typed parameters via *type interactors* such as number fields, text fields, check boxes, and textual and iconic pop up menus. Type interactors limit user input to valid choices. For instance, a sound type interactor provides a pop-up menu offering only the names of the sounds available in the system. Some parameters, such as the key icon in Figure 13, can be visual references to agents in the application.

Elevating programs and program primitives onto the level of a direct manipulation, drag and drop user interface, turns them into tactile objects that are no longer confined to programs and program editors. Similar to dynamic programming languages such as Lisp and Smalltalk, any Visual AgenTalk command or collections of commands can be executed at any point in time. A command is applied to an agent by selecting the command from a palette, specifying the command's parameters, and dragging and dropping the command onto an agent in the worksheet.

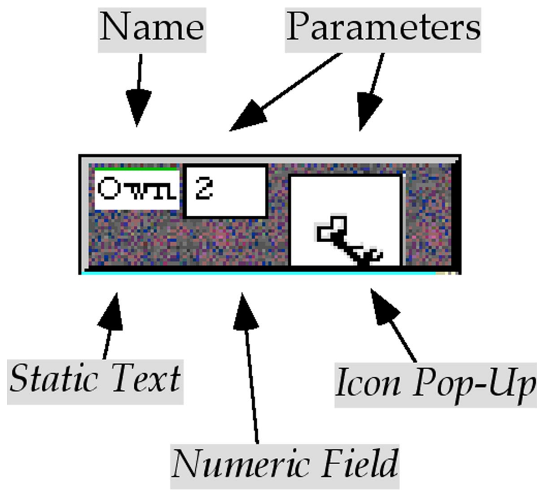


Figure 13: *A Visual AgenTalk Command*

Figure 14 shows a simple Turing machine implemented in AgentSheets. The top window is the worksheet containing a number of agents representing the tape pieces and the Turing machine head. Below the worksheet are two command palettes representing some of Visual AgenTalk's primitives. The condition command palette, on the left, holds commands that allow agents to perceive their world, and the action command palette, on the right, holds commands that enable agents to act on the world. In Figure 14, the "move" action command was selected, its direction parameter was set to (right) via a graphical pop up menu, and the command was then dragged on top of the Turing machine head. This powerful application and execution paradigm allows the end-user to explore and test the repertoire of commands and parameters within the *context* of any agent.

Dropping the command on the Turing head results in the command's execution. The Turing head will move according to the direction indicated in the command, in this case one position to the right (Figure 15).

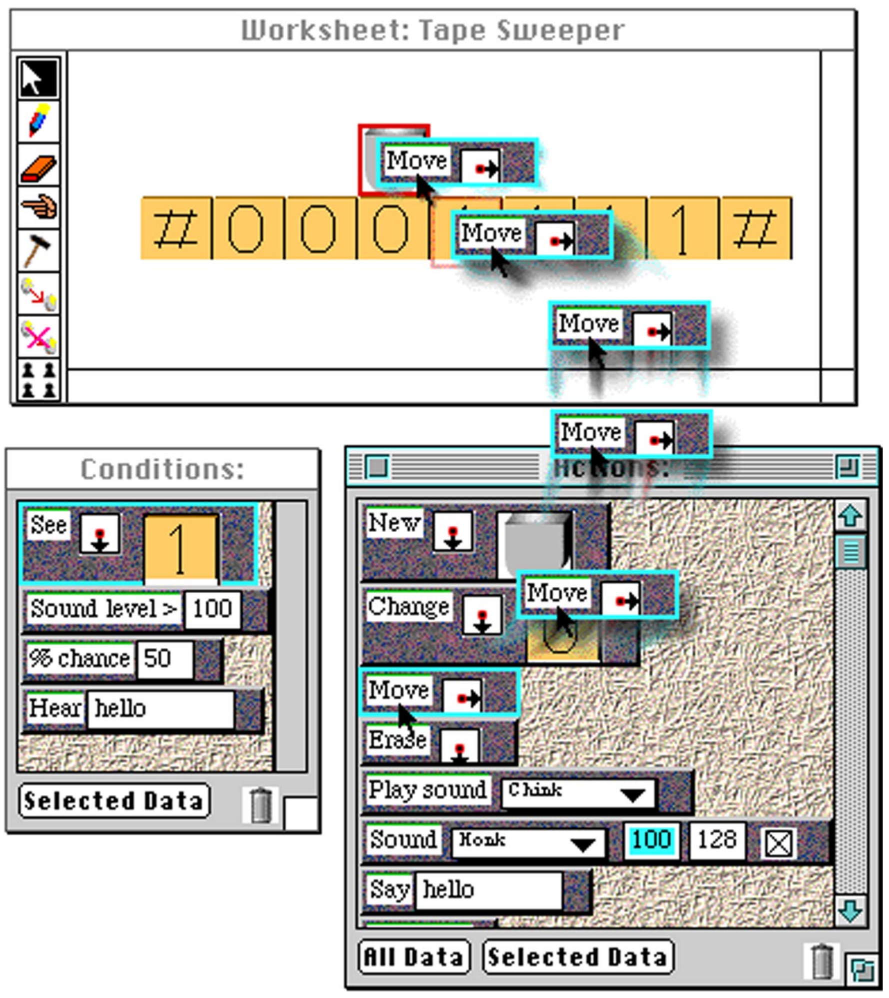


Figure 14: *Commands can be dragged and dropped onto agents to explore their function and to modify agents*

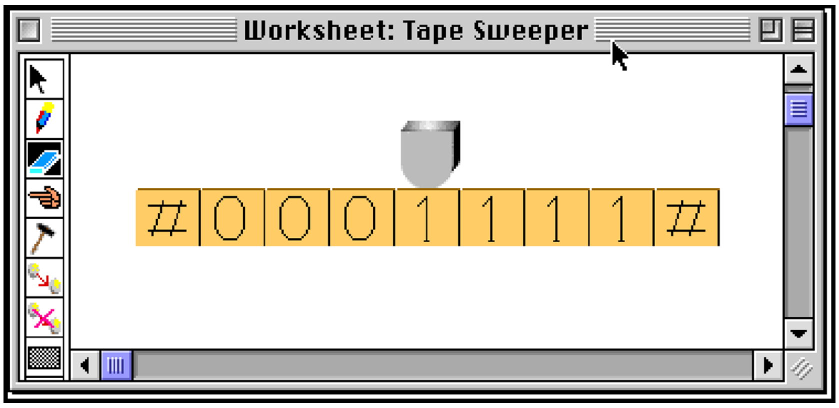


Figure 15: *The Turing machine head after moving one position to right.*

This tactile nature of Visual AgenTalk commands, enabled by the drag and drop interaction mechanism, extends the notion of object-orientation. Tactility encourages a more exploratory style of programming in which end-users comprehend the functionality of programming world objects by manipulating them in ways that were not possible before. As we indicate later this tactility has positive consequences for other media dimensions as well.

4.1.1.2 Program Composition

In addition to allowing commands to be directly applied to agents, a Visual AgenTalk command's tactility also supports program composition. Visual AgenTalk is a rule-based programming environment which treats rules as special kinds of commands. Rules are edited through rule editors (Figure 16). Similar to the notion of nested boxes in Boxer (diSessa, 1991), Visual AgenTalk makes use of containment to indicate part-whole relationships between commands. The IF part, on the left, and THEN part, on the right, of each rule are represented by flexibly sized containers into which commands can be dragged and dropped. The IF part is an implicit conjunction of all the conditions and the THEN part is an implicit action sequence. Rule sets are interpreted from top to bottom. If the rule interpreter finds a rule for which all the conditions are true then it will execute all of its actions, again from top to bottom, and return from one matching cycle.

The agent representing the Turing machine head gets programmed with two rules (Figure 16) capturing the functionality specified in Table 3.

If	Then
there is a "0" below	move right
there is a "1" below	change it to "0" and move right

Table 3: *Rules Governing the Tape Head's Behavior*

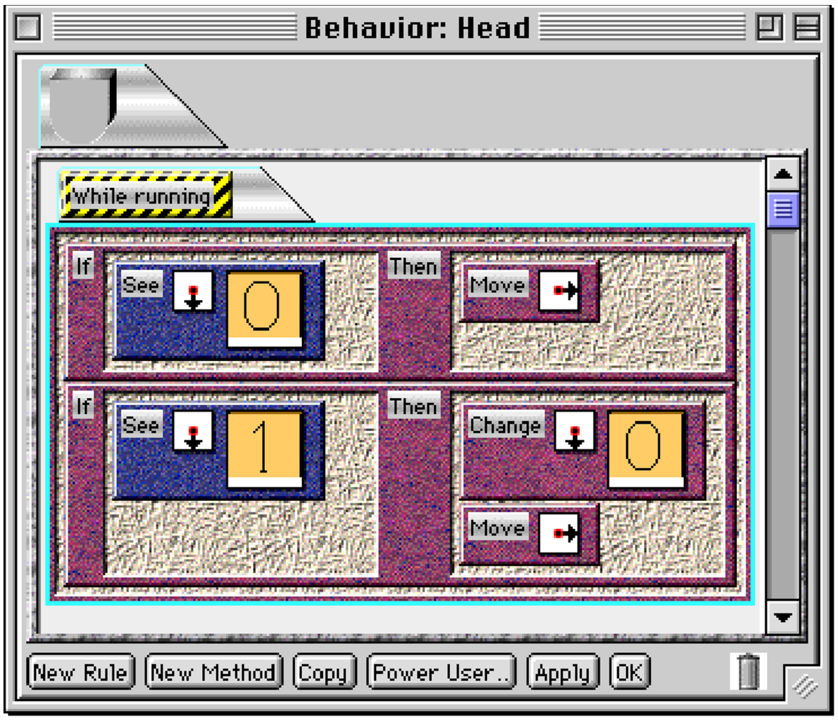


Figure 16: *Visual AgenTalk Rules for the Tape Head.*

Tactility in Visual AgenTalk does not only permit the composition of programs, in the form of rules, but also supports the comprehension of these programs. In Visual AgenTalk each part of the rule, the IF part, the THEN part, individual commands, and even entire rules can be dragged and dropped onto agents in the worksheet, in order to understand how that program or program fragment will execute. Feedback is provided by the system indicating whether the rule matches or not and what will happen if the actions are executed. If a rule or condition does not match, feedback is provided explaining why.

Visual AgenTalk can be easily extended as well. In keeping with the AgentSheets' tradition of being a layered environment, new editors exist to allow the definition of new Visual AgenTalk commands (Repenning and Ioannidou, 1997). Similar to programming in the original AgenTalk environment, more sophisticated users can define the behavior of new conditions and actions in a Lisp-like macro language. These macros are then interpreted using a tool called the Command Factory which produces a new command complete with an interface and the correct type interactors. Once the command is created it is automatically added to the correct palette and can be immediately applied to an agent in a worksheet or added to a rule editor.

4.1.1.3 Program Sharing

In addition to comprehension and composition, tactility is used to ease the sharing of programs and program fragments. This ability has significantly extended AgentSheets' accessibility as a medium and is further described within the Accessibility section.

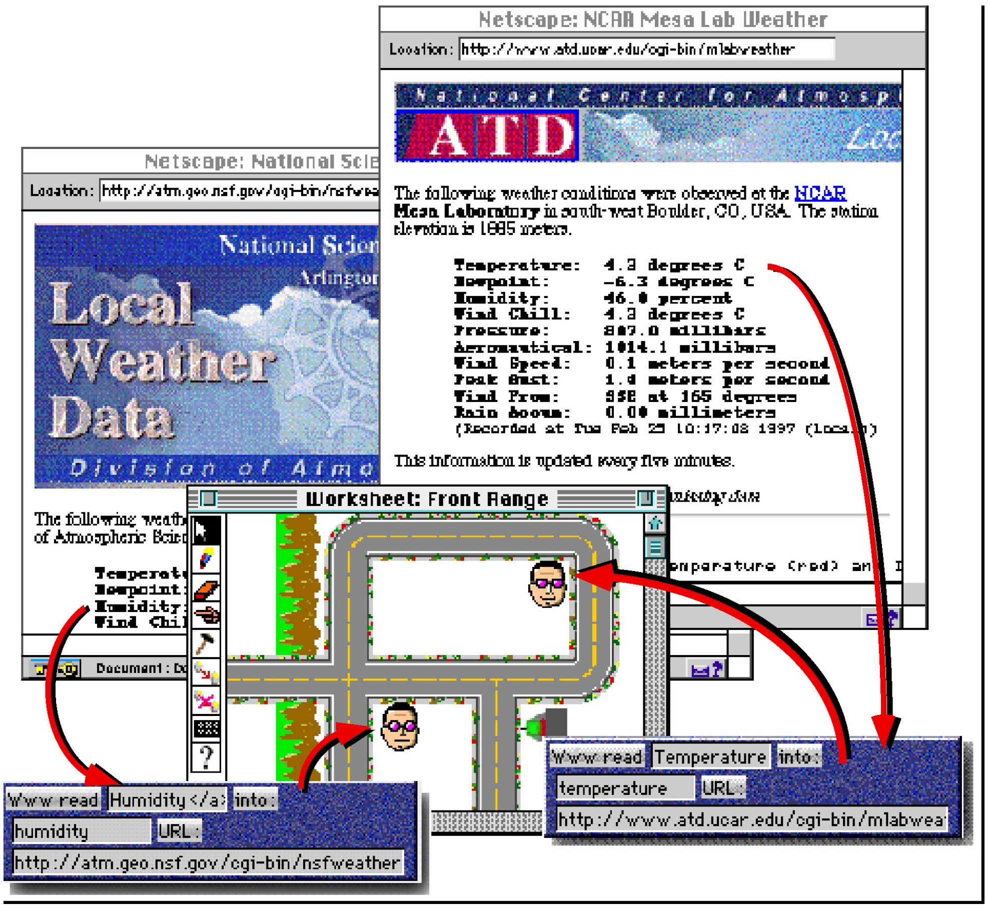
The extension of AgentSheets with Visual AgenTalk has made the environment significantly more interactive by giving a larger community of end-users the ability to create and modify their own agents and agent behavior. Within this capability lies AgentSheets true strength as an educational medium. While interacting with pre-designed simulations can provide educational insight into certain phenomena, it is even more educationally compelling to be able to build your own simulations. Visual AgenTalk, while still not a tool for all users, has been effectively used by learners with no previous programming experience.

4.2 Richness

One of the challenges to creating effective educational media is to allow that media to reflect and incorporate as many aspects of the learner's world as possible. When computational media reduce themselves to abstract, vague representations of objects in the learner's world, then they become less engaging. Papert's original Logo turtle was in fact a mechanical device that actually moved around and drew within the learner's world (Papert, 1980). The problem with the mechanical turtle, however, was that it was bulky and large and inaccessible to many users. Moving the turtle within the world of the computer impoverished the richness of the medium, but enhanced its accessibility. Recently, AgentSheets has addressed richness in two ways: agents are provided with enhanced ability to interact not only with each other, but also with the user as well as with other applications within the computer. Also, the LEGOsheets project, created using AgentSheets, was an attempt to move the computer back out into the real world of the learner while maintaining accessibility.

4.2.1 The Weather Agent: Simulation Meets Web Browser

The very notion of a simulation can be substantially enriched by adding mechanisms to acquire real and live information. A set of World Wide Web Visual AgenTalk commands allows end-users to program agents to access Web pages in real time and extract specific information from them. For instance, in a SimCity-like simulation of Boulder, Colorado two weather agents are placed in a simple map of Boulder. Each agent is accessing a different Web page reflecting weather data at two different locations in Boulder (Figure 17).


Netscape: NCAR Mesa Lab Weather
Location: http://www.atd.ucar.edu/cgi-bin/mlabweather
National Center for Atmosp
ATD
Netscape: National Scie
Location: http://atm.geo.nsf.gov/cgi-bin/nsfwea
National Scien
Arlington
Local Weather Data
Division of Atmo
Worksheet: Front Range
Document: Do
Www read
Humidity</a>
into:
humidity
URL:
http://atm.geo.nsf.gov/cgi-bin/nsfweather
Www read
Temperature
into:
temperature
URL:
http://www.atd.ucar.edu/cgi-bin/mlabwea

The following weather conditions were observed at the NCAR **Mesa Laboratory** in south-west Boulder, CO, USA. The station elevation is 1885 meters.

```
Temperature:   4.3 degrees C
Dewpoint:      -6.3 degrees C
Humidity:      46.0 percent
Wind Chill:    4.3 degrees C
Pressure:      807.0 millibars
Aeronautical:  1014.1 millibars
Wind Speed:    0.1 meters per second
Peak Gust:     1.4 meters per second
Wind From:     SSE at 165 degrees
Rain Accum:    0.00 millimeters
(Recorded at Tue Feb 25 10:17:08 1997 (Local))
```

This information is updated every five minutes.

The following weath
of Atmospheric Scien

```
Temperat
Dewpoint:
Humidity:
Wind Chil
```

Figure 17: *End-users can program agents to pull weather information, such as temperature and humidity, from the Web with the WWW read command*

Data extracted from the Web pages (which get updated every five minutes) such as temperature, humidity, wind direction and speed, can be directly fed into computation. For instance, the two Boulder weather stations that maintain the different Web sites typically report slightly different temperatures. A simple diffusion equation can be used to extrapolate the temperature in arbitrary locations in the Boulder simulation and using the colorization commands that AgentSheets provides, create a color map of the temperature.

The ability to connect a simulation to this kind of live information results in a new genre of information media unifying notions of simulation and Web browser. The live simulation world becomes a highly dense representation of information that otherwise would be widely scattered on many different Web pages interspersed with other irrelevant information and annoying advertisement. Agents become autonomous pull mechanisms providing an ideal balance between information pull and push technologies. The Visual AgenTalk program created by the end-user defines where, what and when agents pull information out of existing Web pages. At the same time, the end-user program defines how the extracted information will be pushed into the simulation. For instance, information pulled out of Web pages can be combined into a sentence that, by means of voice synthesis, will be spoken to the user. The ability to use data pulled out of Web pages allows the computation of new information. An agent could access a number of Web-based price lists featuring a certain color printer. The agent could compare the prices and announce to the user if the prices are significantly different or lower than a specified threshold.

4.2.2 Moving from the Computer to the Real World: LEGOsheets

LEGOsheets [7] (Gindling, Ioannidou, Loh, Lokkebo and Repenning, 1995) is a programming, simulation and manipulation environment created in AgentSheets for controlling the MIT Programmable Brick. The Brick [8], a hand-held computer developed at the MIT Media Lab [9], receives input from sensors, such as light and touch sensors, and can control effectors, such as motors, lights and beepers. The combination of LEGOsheets and the Brick gives children the ability to create physical artifacts (vehicles and robots) and program them with interesting behaviors (Resnick, 1994; Resnick, Martin, Sargent and Silverman, 1996). A lot of the excitement and engagement with LEGOsheets arises from the physical aspect that the Brick provides. It is interesting to create a simulation of a car running around on a computer screen, but it is richer and more interesting when the car is programmed to do so in the real world. Will

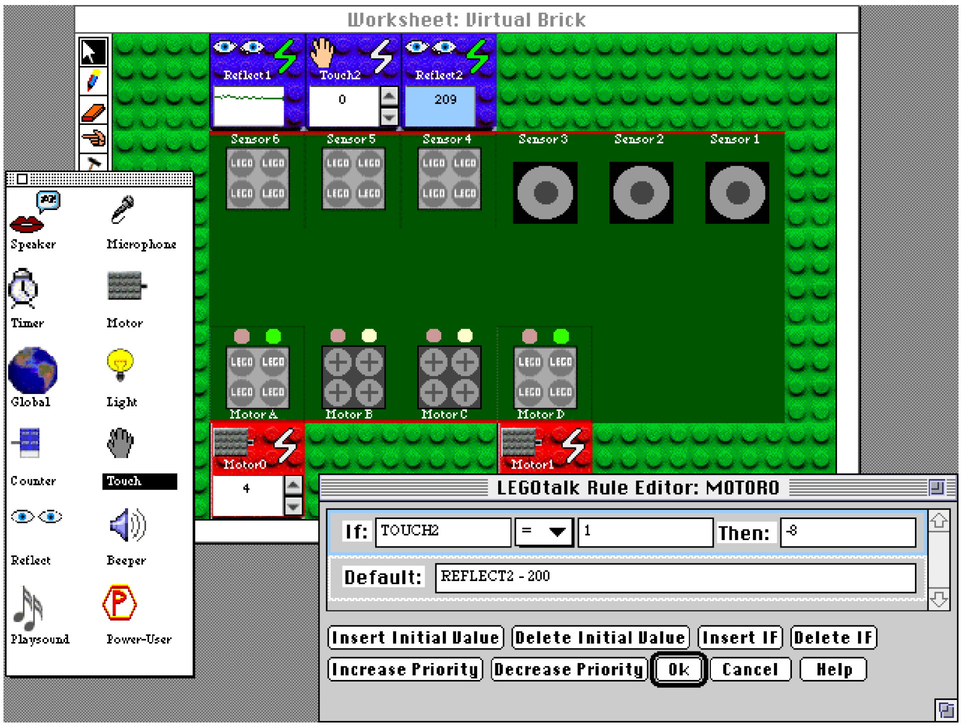
[7] *The LEGOsheets System <http://www.cs.colorado.edu/~l3d/systems/legosheets/>*

[8] *MIT's Programmable Brick <http://lcs.www.media.mit.edu/groups/el/projects/programmable-brick/>*

[9] *The MIT Media Lab <http://www.media.mit.edu/>*

it navigate a road? How will it cope with obstacles? When running in the real world, a lot more than has been anticipated can occur.

When using LEGOsheets (Figure 18) and the Brick to build autonomous artifacts, such as vehicles (Figure 19, left), the richness of the resulting behavior does not come from the complexity of the program itself, but from its interactions with the real-world (Simon, 1981). A simple program should provide an opportunity for interesting exploration of the vehicle's behavior in the world, including more social activities.



0 209 4

If: TOUCH2 = 1 Then: -8

Default: REFLECT2 - 200

Figure 18: *The LEGOsheets environment*

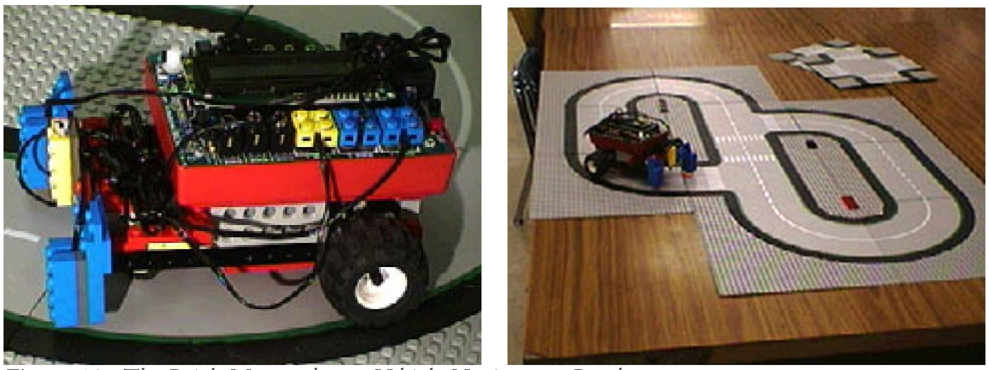


Figure 19: *The Brick Mounted on a Vehicle Navigates a Road*

The simple program shown in Figure 20, written by middle school students, enables the vehicle to navigate on a LEGOroad system (Figure 19, right), by picking up signals from the road pieces with two reflectance sensors and using the signals to drive the two motors of the vehicle accordingly.

```
MOTOR1:
 if: REFLECT2 > 185  then: 8
 if: REFLECT1 > 185  then: -8
 default: 6
MOTOR0:
 if: REFLECT1 > 185  then: 8
 if: REFLECT2 > 185  then: -8
 default: 6
```

Figure 20: *The Navigate Program*

When LEGOsheets and the Brick were presented to a group of third graders at Whittier Elementary school, the students came up with a game of dynamically creating worlds for the vehicle to move in (Figure 21). The richness of the Brick enables a more social setting in which several students were collaborating to build various worlds. The process of programming became a highly dynamic and social activity in which children engaged in discourse and planning.

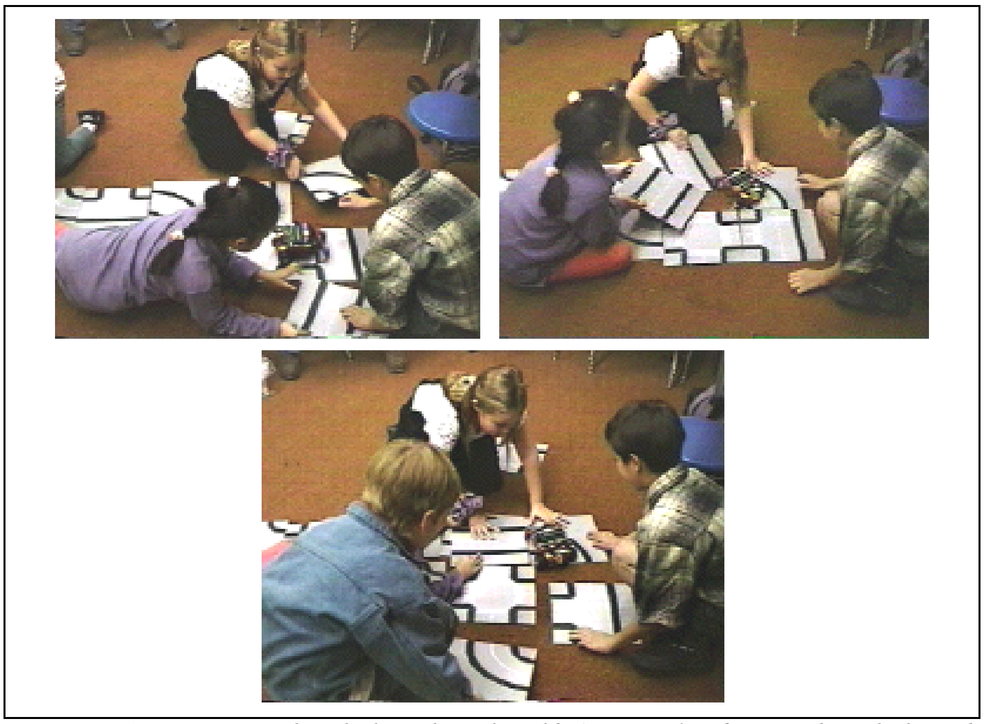


Figure 21: *"Programming" the vehicle in the real world (Images taken from a video, which can be viewed in the Web version of this article)*

In a very different setting, LEGOsheets and the Brick were used by Jim Johnson, a professor in the Department of Fine Arts at the University of Colorado for creating art (Ambach and Repenning, 1996). The artist was initially approached by a group of graduate students who wanted to build a more domain-specific user interface on top of LEGOsheets, so that Jim Johnson could program the Brick himself, and use it to create art. The notion of small ubiquitous computing devices crawling on big pieces of canvas was intriguing to him, but the prospect of programming seemed daunting. Moreover, it was not clear what this new medium would be good for. What exactly could the Brick do on the canvas? At first, he envisioned using the Brick similar to the way the turtle is programmed in Logo (forward 100, right 90). However, he soon realized that this approach was too analytical and controlled. He was more interested in how the Brick interacted with the world.

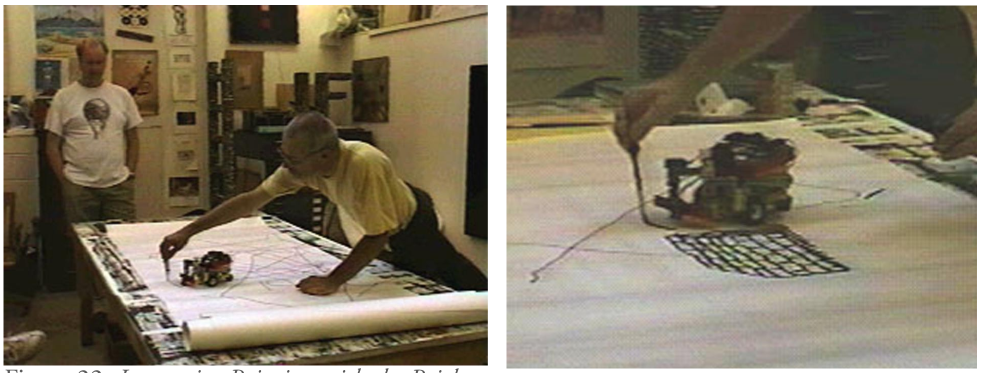


Figure 22: *Interactive Painting with the Brick*

We then created a few different programs that Jim could work with, such as a vehicle that reacted to black lines in a 'random' way. He experimented with these behaviors in a variety of ways. On a big canvas, Jim would leave brush strokes of black ink and the vehicle would react to them. This way, the artist could control the direction that the vehicle took by drawing lines in its path, forcing the vehicle to turn (Figure 23). A marker mounted on the vehicle left traces on the paper, creating an intricate pattern. Jim also used the same program to explore its behavior on pre-drawn figures on the canvas. He drew different figures (circles, triangles, squares etc.) on pieces of paper and placed the vehicle on them. He then explored the different patterns that emerged (Figure 23).

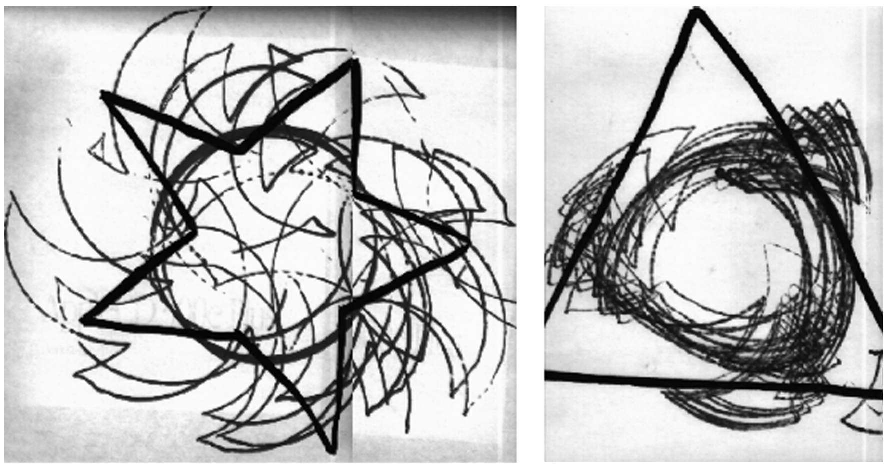


Figure 23: *Jim Johnson's Brick Art*

In another drawing program, a reflectance sensor and a touch sensor were used to create a painting vehicle that traced objects. After using it with various things in his studio, Jim used the vehicle to trace him (Figure 24). The variety of possibilities that the interaction of the Brick with the world provided were not limited by the internal programs, but were open to the creativity and imagination of the artist.

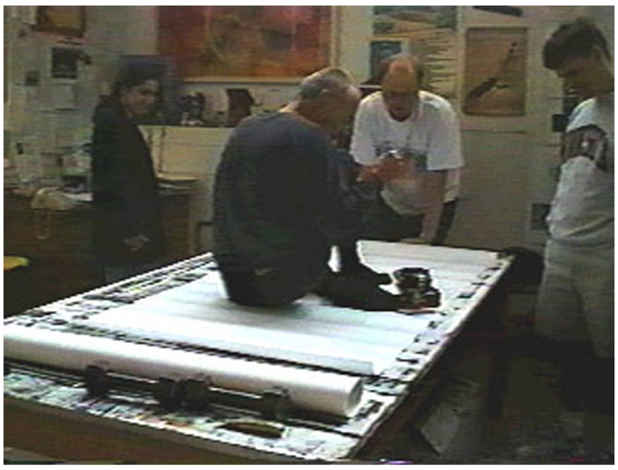


Figure 24: *The Brick Traces Jim Johnson, (Image taken from a video, which can be viewed in the Web version of this article)*

Jim Johnson wrote:

> *One of the most interesting aspects of this project for me is in the fact that using the brick as a drawing tool has forced me to consider analog forms. The kind of drawing I had been doing before I started using the computer to make art could be called an analog form of art. Now, I'm being brought back to it by a little kinetic computer. Ironic. No?*

The richness originating from the physical aspects of the combination of LEGOsheets and the Brick, renders the system eligible as a medium used by children and adults alike for communicating ideas. Whether those ideas stem from interacting with the Brick to create art or from engaging in dynamic games of programming vehicles to navigate through a LEGOroad maze as illustrated above, these interactions with both the virtual and the real world is a motivation for more social forms of "programming" that moves beyond the traditional, solitary style of programming and therefore enables groups of users to interact and communicate their ideas.

4.3 Accessibility

In addition to the powerful new programming paradigms to build simulations, an environment such as AgentSheets also needs to feature mechanisms to support reuse (Rosson and Carroll, 1996). No matter how powerful a programming environment is, there simply is not sufficient time to create entire simulations from scratch, especially in an educational setting. To address this issue of reuse and to increase its accessibility, Agentsheets was enhanced by the Behavior Exchange system which is a Web-based repository that allows a community of users to efficiently share and exchange their agents with other Agentsheets users. Yet another attempt to increase the issue of accessibility was the creation of the Ristretto agent-to-java-byte-codes generator that allows users to publish their simulations on the Web.

4.3.1 *The AgentSheets Behavior Exchange*

The Behavior Exchange [10] , [11] (Repenning and Ambach, 1997; Repenning *et al.*, 1998), is an evolving Web-based information space that allows users to exchange agents that have been created in AgentSheets through the Web. It contains two different kinds of information:

- *Informal Information*: Information not interpreted by the computer. The look of an agent, e.g., a scuba diver , textual descriptions concerning what the agent does, who created it, why and how it is used belong into the informal information category.
- *Formal Information*: Information interpreted by the computer. All the rules that determine the behavior of an agent are considered formal information.

The combination of informal and formal information turns these agents into a social currency of exchange and, therefore, the Web repository of agents becomes a forum for AgentSheets users to exchange novel ideas and designs through the shared artifacts. Some users (the producers) produce agents and share them by putting them into the Behavior Exchange. Other users (the consumers) identify interesting agents that reside in the repository, drag and drop them into their AgentSheets simulation and can immediately use them. The Behavior Exchange not only increases the accessibility of agents, but it also transforms the design of computational artifacts to a social activity.

[10] *Behavior Exchange <http://agentsheets.cs.colorado.edu/Behavior-Exchange/>*

[11] *Martin Rausch's Masters Thesis on the Behavior Exchange (Adobe Acrobat document) <http://www.cs.colorado.edu/~l3d/systems/agentsheets/Documentation/Rausch-Thesis.pdf>*

4.3.1.1 Exchanging Agents

For the Behavior Exchange, the Web serves as a medium through which the formal and informal information of agents is exchanged. The collaborative process of exchange includes the following activities:

- *Building*. Producers create new agents for their simulation.
- *Sharing*. Producers share some or all agents they have created with the rest of the AgentSheets community, by putting them into the Behavior Exchange.
- *Locating*. Consumers locate relevant agents in the Behavior Exchange by browsing or searching the repository.
- *Acquiring*. Consumers drag interesting agents and drop them into their design environment.
- *Comprehending*. Consumers "play" with agents to understand what agents do and evaluate usefulness.
- *Modifying*. Consumers change agents to fit their needs by modifying agent behavior and look if necessary. Consumers can share modified agents again with community by putting them back into Behavior Exchange and in turn become producers.
- *Packaging*. Consumers may want to put their final simulation product into a format accessible to a larger community of users, turning the entire simulation into a Java applet (see § 4.3.2).

Producers and consumers are just roles of users that can change over time (Repenning et al., 1998). Most Behavior Exchange users first play with a number of agents created by others before they create agents that they want to share with the community. This shift was observed during the Pearl Street project at the Centennial Middle School's Computer Club, where students used the Behavior Exchange in building a scavenger hunt game taking place in Pearl Street, a popular mall area in the students' town (for more details on this project, please see Section 3.1.2).

4.3.1.2 Scenario: Exchanging a Diver

The following scenario illustrates our framework by elaborating the collaboration activities.

Building Agents

Bob, is working on a fish tank simulation. He wants to have a scuba diver that can breathe in this fish tank. He creates a scuba diver and a bubble agent for which he defines two icons (Figure 25).

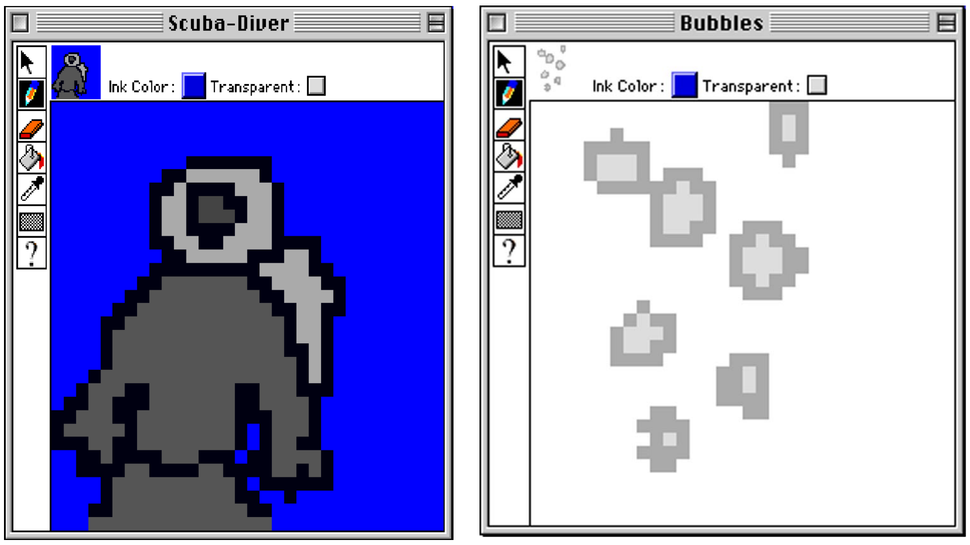


Figure 25: *Icons for the scuba diver and air bubbles.*

The new scuba diver agent can already be put into the fish tank but has no behavior yet. Bob wants the scuba diver to create bubbles when the space bar on the keyboard is hit.

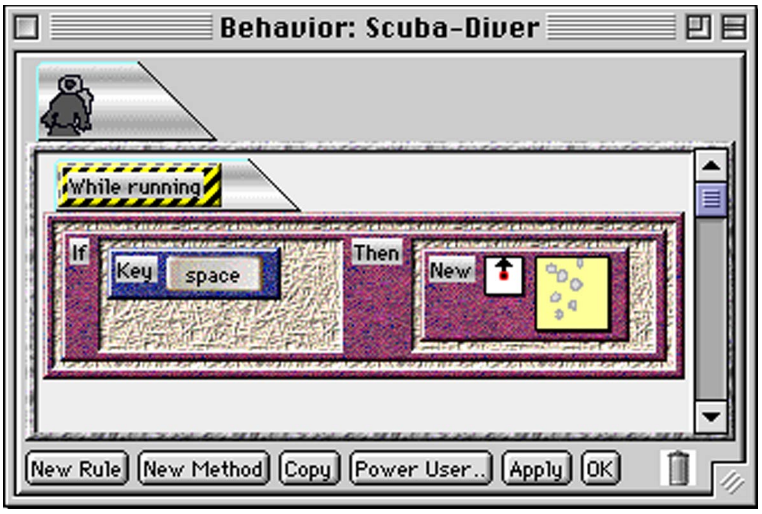


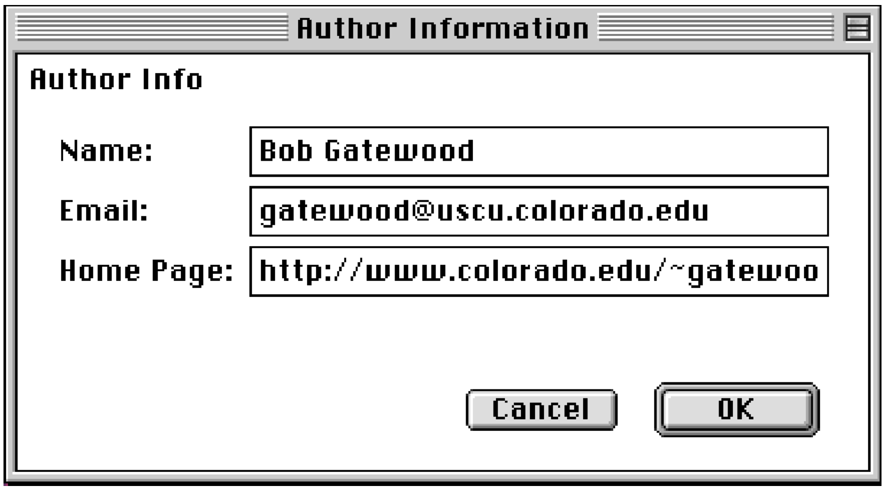
Figure 26: *Behavior Editor. A rule for the diver. IF the space key is pressed THEN the diver creates a new air bubble.*

Using the behavior editor, Bob defined this simple behavior in a single rule (considerably more complex behaviors can be built including procedural abstractions, numerical operations, colorization (Repenning and Ioannidou, 1997), but a simple example is used to focus on the exchange of agents).

AgentSheets includes a rich set of condition and action commands that can be dragged into IF-THEN rules structures. Command parameters are all defined using direct manipulation. For instance, Bob specifies the key that he want to use to trigger the bubbles by clicking at the Key condition and pressing the key of his choice on the keyboard.

Sharing Agents

Bob wants to share his scuba diver and bubble agents by submitting them to the Behavior Exchange. Submitting his first agent the system prompts him to fill in the author information form (Figure 27).



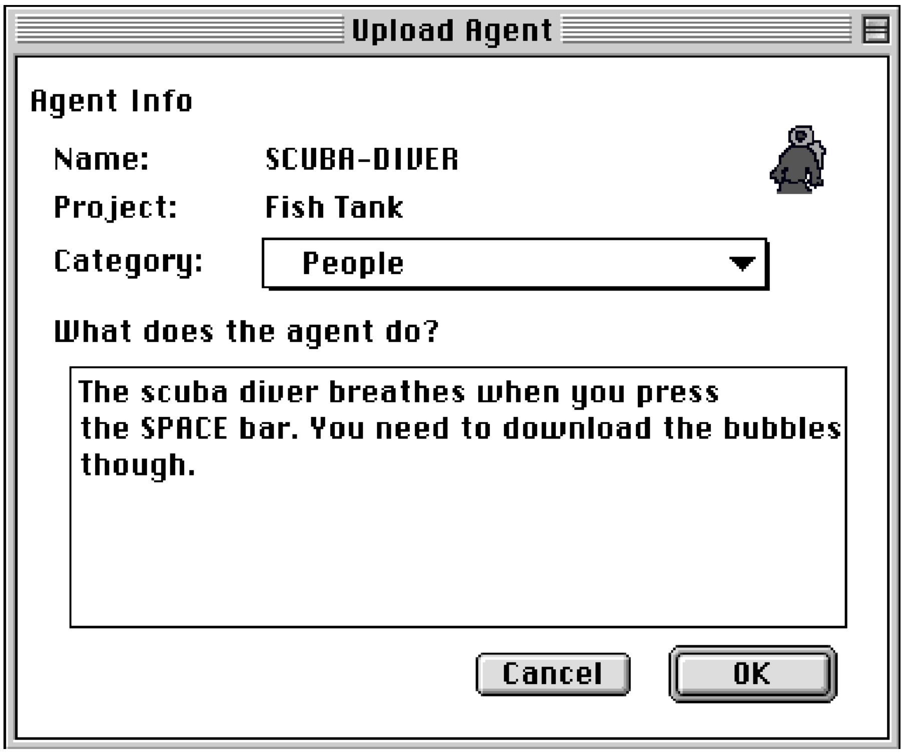
Author Information

Author Info

Name:	Bob Gatewood
Email:	gatewood@uscu.colorado.edu
Home Page:	http://www.colorado.edu/~gatewoo

Figure 27: *Author information need to be filled in only once.*

For each agent that Bob submits he has to fill in the agent Upload form in which he categorizes the agent and provides some informal information about what the agents is and how it should be used (Figure 28).



Agent Info

Name:	SCUBA-DIVER
Project:	Fish Tank
Category:	People

What does the agent do?

The scuba diver breathes when you press the SPACE bar. You need to download the bubbles though.

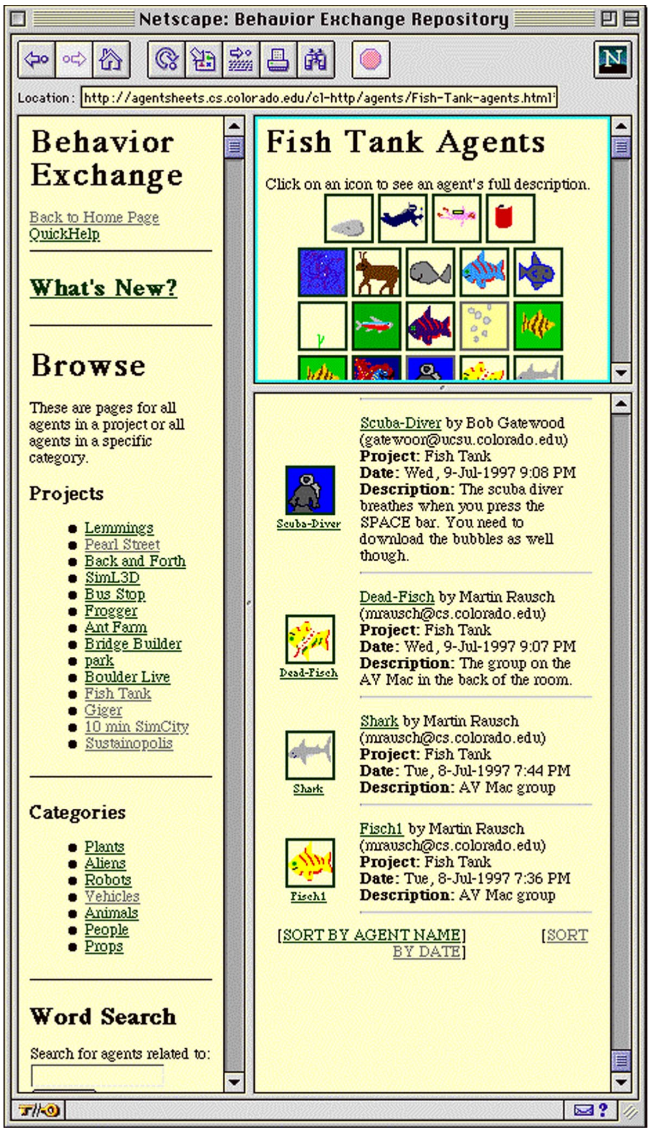
Figure 28: *Bob categorizes his agent and add some notes*

The informal agent upload information helps users of the Behavior Exchange who retrieve this agent to understand the context in which it was created.

Bob's new agents are directly uploaded to the Behavior Exchange Web server making them ready for other people to be used.

Locating Agents

Beth, a middle-school student who is an AgentSheets user, visits the Behavior Exchange with the goal to find some new agents for her fish tank (the fish tank comes with the AgentSheets distribution). AgentSheets takes her directly to the Behavior Exchange via Netscape. Beth has a number of ways to locate interesting agents. She can browse the Behavior Exchange by selecting projects or by selecting categories. If she has more specific ideas she can also used a word search to find agents. Beth, selects the Fish Tank project and browses the agent thumb nails presented to her. She is interested in the scuba diver. Clicking the scuba diver reveals the comments made by Bob about the scuba diver. In this case the comment are quite important since they point out that Beth should also take the bubble agent if she is interested in the scuba diver (Figure 29).


Netscape: Behavior Exchange Repository

Location: http://agentsheets.cs.colorado.edu/cl-http/agents/Fish-Tank-agents.html

Behavior Exchange

Back to Home Page
QuickHelp

What's New?

Browse

These are pages for all agents in a project or all agents in a specific category.

Projects

- Lemmings
- Pearl Street
- Back and Forth
- SimL3D
- Bus Stop
- Frogger
- Ant Farm
- Bridge Builder
- park
- Boulder Live
- Fish Tank
- Giger
- 10 min SimCity
- Sustainopolis

Categories

- Plants
- Aliens
- Robots
- Vehicles
- Animals
- People
- Props

Word Search

Search for agents related to:

Fish Tank Agents

Click on an icon to see an agent's full description.

Scuba-Diver by Bob Gatewood (gatewoon@ucsu.colorado.edu)
Project: Fish Tank
Date: Wed, 9-Jul-1997 9:08 PM
Description: The scuba diver breathes when you press the SPACE bar. You need to download the bubbles as well though.

Dead-Fisch by Martin Rausch (mrausch@cs.colorado.edu)
Project: Fish Tank
Date: Wed, 9-Jul-1997 9:07 PM
Description: The group on the AV Mac in the back of the room.

Shark by Martin Rausch (mrausch@cs.colorado.edu)
Project: Fish Tank
Date: Tue, 8-Jul-1997 7:44 PM
Description: AV Mac group

Fisch1 by Martin Rausch (mrausch@cs.colorado.edu)
Project: Fish Tank
Date: Tue, 8-Jul-1997 7:36 PM
Description: AV Mac group

[SORT BY AGENT NAME] [SORT BY DATE]

Figure 29: *Behavior Exchange showing some agents from the fish tank project.*

Acquiring Agents

Beth acquires the scuba diver and the bubble agents by simply dragging them out of the Web page into the AgentSheets gallery (Figure 30).

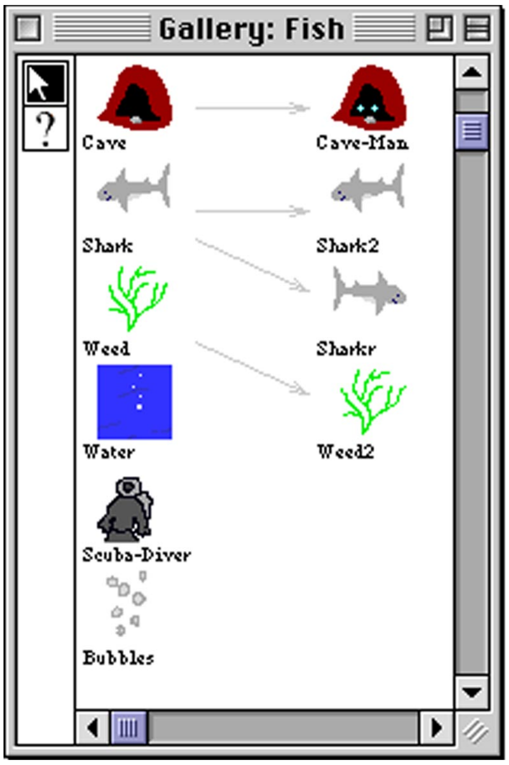
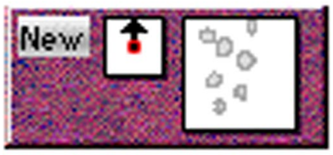


Figure 30: *Gallery of agents including diver and bubble agent*

The Behavior Exchange Web pages completely encapsulate agents. Dragging agents out of a Web page will copy all necessary resources including icons and the behaviors.

Comprehending Agents

Beth gets full access to the newly acquired agents. She inspects the behavior of the scuba diver agent by opening up its behavior editor. She can test any of the divers behavior within her own AgentSheets environment. She can test rules or even individual conditions and actions by simply dragging and dropping them onto a diver, or any other agent, in her worksheet. For



instance, when dragging the New action, out of the diver behavior (Figure 31) onto the sea weed, , in the fish tank, a bubble will be created above the sea weed. If the simulation is running the bubble will float onto the surface of the tank and pop.

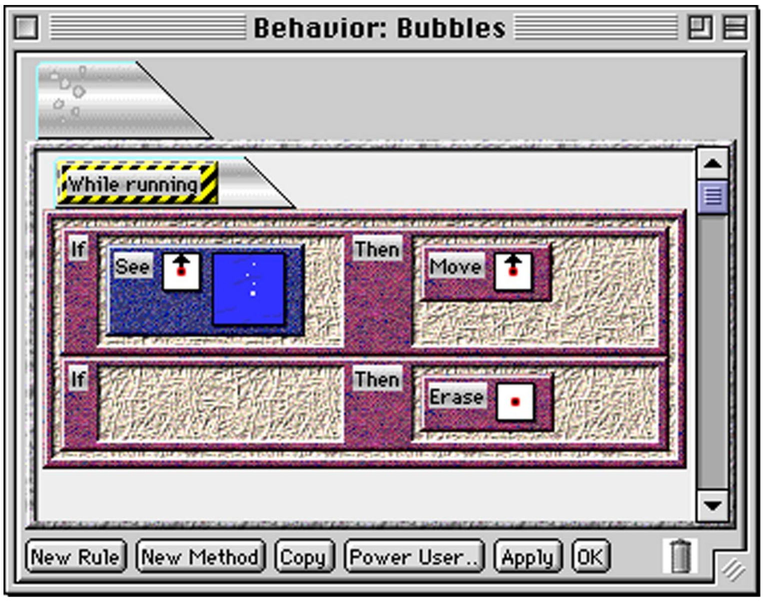


Figure 31: *Bubbles float up through the water. If there is no more water they pop.*

We believe that the ability to tinker with language pieces without the need to first combine them into complete programs is crucial to the usability of programming approaches for non programmers. Due to the graspability of language components, this approach, that we call Tactile Programming (Repenning and Ambach, 1996), supports exploration and significantly simplifies debugging which can be hard in rule based languages (Gilmore, Pheasey, Underwood and Underwood, 1995; Rader *et al.*, 1997).

Modifying Agents

After understanding the behavior of the scuba diver, Beth modifies the current divers behavior. She wants to have a more autonomous diver creating bubbles without having to press any key and she wants to count the bubbles. She replaces the Key condition with a OnceEvery-n-seconds condition and adds a Visual AgentTalk formula incrementing the value of a new attribute called “bubbles” (Figure 32).

The combination of tinkering support with ease of change can overcome some of the problems encountered in rewrite rule-based programming and programming by example systems. For instance to change a graphical rewrite rule in Cocoa users need to recreate the example situation in which the rule was created. This can be tedious to the point where users resist modification or prefer to create new rules masking old ones instead of modifying the old rule (Rader *et al.*, 1997).

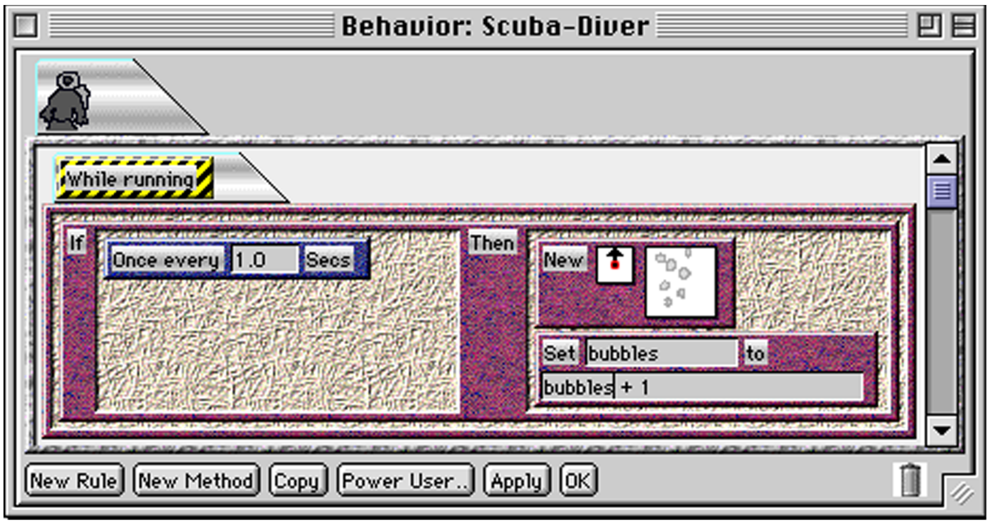


Figure 32: *Modified diver behavior creating a new bubble every second and counting the number of bubbles.*

As illustrated in the scenario above, different users use the Behavior Exchange in different ways. More passive users use the Behavior Exchange as a source of reusable artifacts such as agents, whereas more active users contribute their agents to the Behavior Exchange. Many of them have started their explorations by reusing other people's agents and thus have improved upon their own abilities until they were confident enough to contribute their own agents. Motivated by the experiences gained when reusing agents created by other designers, those users who used to be passive, decide to become more active by contributing their own creations.

Currently the Behavior Exchange features agents in categories such as Animals: Fish, Sharks; People:
Scuba Divers, Aliens; Transportation: Roads , traffic lights and cars ; Special

Effects: Fire and Flash; Musical Instruments, including pianos , and trumpets .

4.3.2 Ristretto™: the Agent-to-Java Generator

While Java enables its users to create interactive applets and make them publicly available in their Web pages, it is still a programming language that requires expertise that usually only professional programmers possess and is neither available to nor desirable for end-users. End-users are people who typically do not care about the notion of programming per se; still, they would like to have an increased sense of control over computers and be able to create artifacts, such as interactive Java applets, that traditionally could only have been created by professional programmers. We claim that in behavior processing (the kind of programming the occurs in AgentSheets), Java should be at the level that Postscript is in word processing. When users create a document in word processing and want to print it out, they just push a button that does exactly that. It is of little or no interest to them that, in order to be printed, the document is first translated into Postscript and sent to a printer where a Postscript interpreter creates the printed version. Just as users of word processors do not have to learn Postscript to print their document, end-users in behavior processing should not have to learn Java to create an applet out of their simulations.

Recently, some "instant" Java tools that "claim to put Java applet creation within reach of everyone" were developed (Venditto, 1997). However, these tools are usually limited to parameter manipulations that allow the users to change existing applets in various ways, such as set foreground and background colors and fonts. Even though this is an important step towards end-user creation of applets, we believe that there should be tools that take this idea even further by allowing end-users to create their own applets from scratch. AgentSheets provides the Ristretto™ generator [12] layer (Repenning and Ioannidou, 1997) to allow end-users to create Java applets out of their simulations, without having to learn Java, or even worry about its existence.

Referring to our scenario in the previous section, suppose Beth created a Fish Tank simulation in AgentSheets in which fish swim around and sharks eat fish. It would probably be impossible for her to create the same simulation as a Java applet for Beth to include in her homepage. In fact, it would probably take one or two days, for a professional programmer to achieve this. With Ristretto, however, this process becomes a matter of minutes for end-users as well as more sophisticated users. Once the users are happy with their simulation, by pressing a single button, Ristretto generates a complete Java applet that can be embedded in Web pages and then be accessed remotely through the internet by other users. In just a few seconds, Ristretto compiles every agent behavior directly into Java class files consisting of Java byte codes and compiles agents depictions into GIF files. Translating agent rules directly into Java bytecode results in very efficient applets because the only run-time interpretation left is in the Java virtual machine.

[12] *Ristretto agent-to-java-byte-code generator <http://www.agentsheets.com/products.html>*

Beth makes her entire simulation to be accessible to other users. All she has to do is to press the Ristretto™ button in order to turn her Fish Tank including her modified divers into a Java applet [13] (Figure 33). The applet created is ready to run on a large number of different Java runtime support environments, including Netscape Navigator, MS Internet Explorer, JDK Applet Runner, and MRJ Apple Applet Runner, allowing the simulation generated to be used on a wide range of hardware platforms.

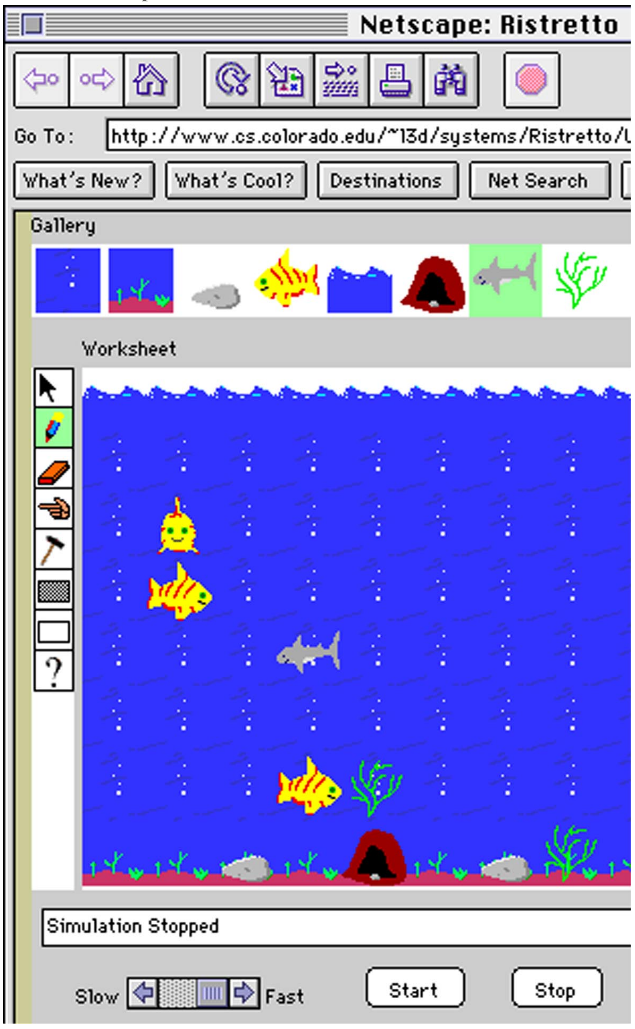


Figure 33: *Fish Tank applet generated with Ristretto agent-to-Java-byte-code generator.*

13 *The Fish Tank Applet <http://www.cs.colorado.edu/~l3d/systems/Ristretto/UserManual/fishtank/MyApplet.html>*

The Ristretto-generated applet allows its users to edit the simulation world. Users can add and remove agents (in the fish tank: water, ground, rocks, small fish, waves, caves, sharks, and weeds), can start and stop the simulation, and can change the speed at which the simulation is running. Furthermore, the Ristretto-generated applet can be controlled through JavaScript or interact with other applets through inter-applet Java functions

5. Future

The previous sections of this article have described the evolution of AgentSheets to a communication medium and our experiences using it within different educational settings. Thinking of AgentSheets in terms of a medium has directed our development efforts and will continue to do so. Now, the challenge is to think of AgentSheets not only as a medium itself, but as a way to contribute to a larger, more varied medium where all kinds of educational simulations and applications can be easily combined in order to develop compelling and specific educational experiences. We have started to consider not only the technology that needs to exist in order for this medium to become a reality, but also the possibilities that it would afford.

5.1 Combining Simulation with Analysis: SimCalc and AgentSheets

In the AgentSheets context, agents can be seen as fine-grained, communicating components. The patterns of communication can be defined using Visual AgenTalk by spatially referring to other agents. In many cases it is useful to think of a worksheet, i.e., a world populated with dozens or even thousands of agents, as a higher level component itself capable of communicating with other components.

The combination of AgentSheets with the SimCalc system [14] has been explored as a mechanism for determining what kind of communication needs to occur between high-level components. SimCalc provides an excellent collection of real-time, mathematical analysis tools that can be used on data generated from AgentSheets simulations. The example shown below simulates a virus spreading though a community of agents. SimCalc defines the probability of the virus spreading as a function over time using a user-editable graph (Figure 34, top left). Running the simulation causes SimCalc to forward those values to AgentSheets. New Visual AgenTalk commands allow agents to access the SimCalc values and to control the virus-spreading behavior accordingly (Figure 34, bottom). Once the simulation starts running, agents move around in the worksheet infecting each other (Figure 34, right), according to the probabilities established in SimCalc. After the simulation is run, the total number of infected agents is reported back to SimCalc and plotted (not shown in Figure 34).

[14] *SimCalc System <http://www.simcalc.umassd.edu>*

SimCalc provides the user with a more abstract view of the virus spreading simulation, and enables the answering of new questions. Will the total number of infected people increase linearly or exponentially? AgentSheets provides a more concrete level of observation. Users can keep track of individual agents, change the situation in which the agents are in or even extend the simulation to include more factors. For instance, a new agent acting as a doctor , , is able to administer a vaccine and heal infected agents. In turn, new questions arise including the critical ratio of people to doctors in order to prevent an epidemic.

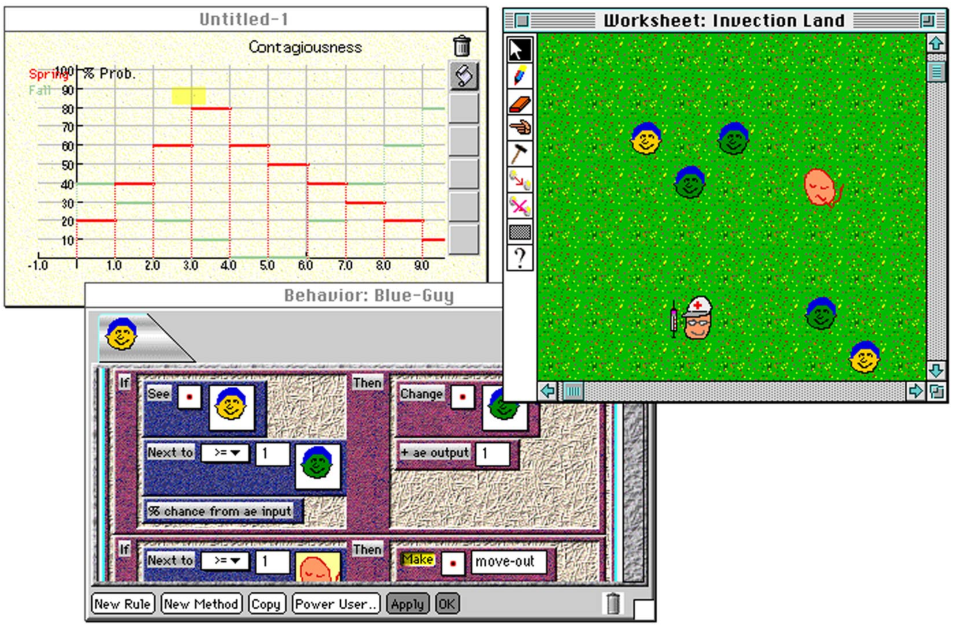


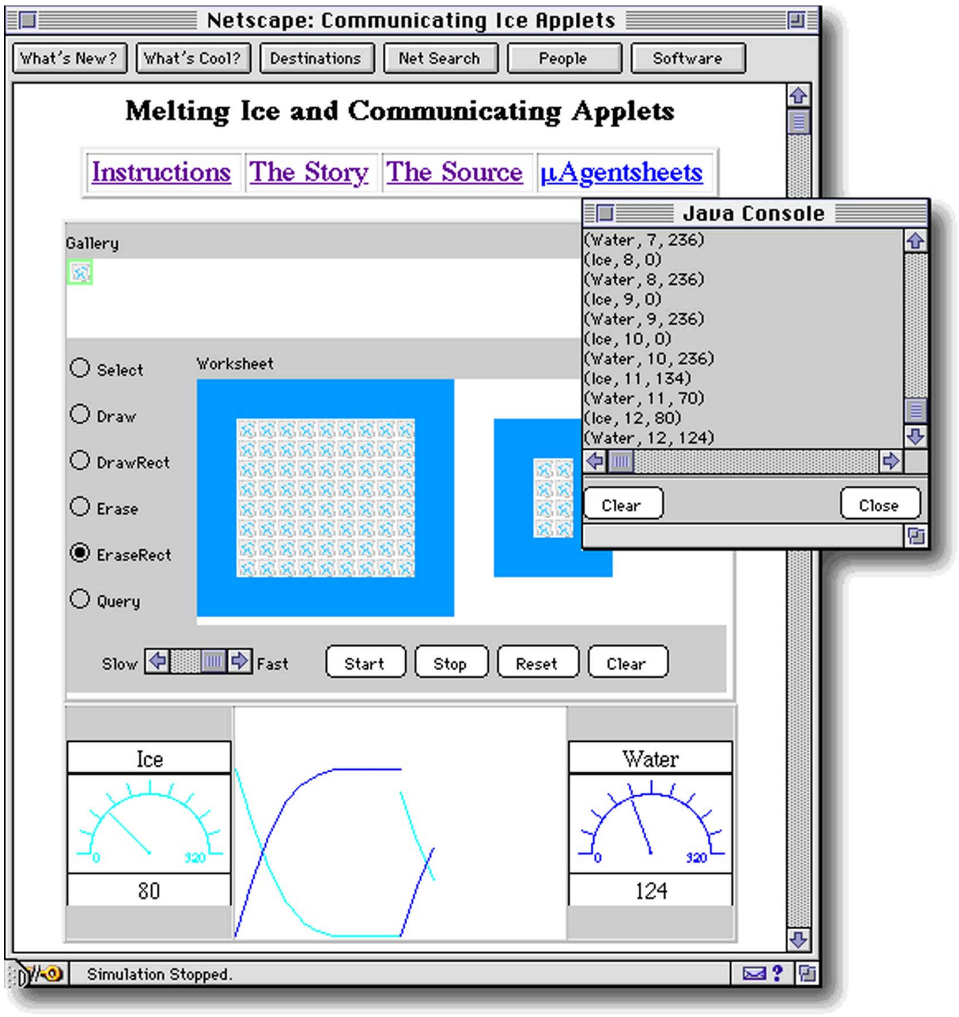
Figure 34: *SimCalc (top left) Combined with AgentSheets*

5.2 Educational Object Economy

Our experiences with AgentSheets and SimCalc have shown the promise of combining sophisticated applications in ways that are greater than the sum of the parts. In the AgentSheets and SimCalc example, both applications were extended allowing both the framing and solving of new types of problems that could not be solved using either of the applications individually. Working with the East/West Consortium, we will play a part in defining and creating a large, evolving, and platform-independent medium containing a varied collection of educational software components that can be easily combined. This "educational object economy" [15] (introduced in Spohrer, Sumner and Buckingham Shum, 1998, this issue) can potentially contain a large collection of reusable educational software components written in Java. To

[15] *Educational Object Economy, The EOE Foundation <http://www.eoe.org>*

experiment with this concept, we took a µAgentSheets applet (a simulation of Melting Ice) and combined it with other applets that we found in existing applet repositories such as Gamelan [16] and JARS.[17]



```
(Water, 7, 236)
(Ice, 8, 0)
(Water, 8, 236)
(Ice, 9, 0)
(Water, 9, 236)
(Ice, 10, 0)
(Water, 10, 236)
(Ice, 11, 134)
(Water, 11, 70)
(Ice, 12, 80)
(Water, 12, 124)
```

Figure 35: *The Melting Ice Applet Combined with other Applets*

16 *Gamelan Applet/Java Beans Repository <http://www.gamelan.com/pages/Gamelan.javabean.general.html>*

17 *JARS Applet/Java Beans Repository <http://jars.developer.com>*

Although these applets could not be combined without modifications to the source code, these modifications are not necessary or difficult now that new component communication technologies such as Java Beans [18] are available. In Figure 35, the Melting Ice applet at the top of the window is combined with a graphing applet (Figure 35, bottom, middle) and two instantiations of a Meter applet (Figure 35, bottom, right and left [19]). The graph is used to show the quantities of ice and water changing over time as the ice melts, and the meters show the exact number of ice and water agents at each time slice. Furthermore, the grapher applet prints out the raw data (Figure 35, middle, right) in a window where the data can be copied and pasted in to other tools. Again, combining our Melting Ice applet with other applets created a more compelling educational experience by providing multiple views of the same data. Although many technical and social issues need to be resolved before an educational object economy really exists, our experiments indicate that such a vast and varied medium will be beneficial both to educational software developers as well as to learners.

In the ESCOT grant that we are currently workingon, joined with researchers from SRI Palo Alto, University of Massachusetts Dartmouth, the MathForum and the ShowMe Center at the University of Missouri, and publishing companies such as Key Curriculum Press, we will:

> *…investigate how software innovations can accumulate, integrate, and scale up to meet the needs of systemic reform of K-12 mathematics and science education. Our goal is not a single software product, but an understanding of how "integration teams" comprised of developers, authors, teachers, Web facilitators, and others could compose lessons by combining graphs, tables, simulations, algebra systems, notebooks and other tools available from a shared library of re-usable components. Our integration teams will draw upon Java versions of such powerful tools as SimCalc MathWorlds, Geometer's Sketchpad and AgentSheets to begin addressing the needs of five new middle school mathematics curricula for empowering Web-based learning technology.* [20]

6. Conclusion

AgentSheets is a tool and medium that empowers casual computer users with no formal programming training to build and publish Web-based interactive simulations. Even though the design and authoring environment of the system is limited to the Macintosh platform, the run-time environment can run the Ristretto-generated applets on any platform (Windows/Intel and Unix machines) or browser (Netscape and Internet Explorer). Agentsheets has been successfully

[18] *Sun's Java Beans <http://java.sun.com/beans>*

[19] *Source of Meter applet in Figure 34 <http://techni.tachemie.uni-leipzig.de/~jar/jappl/meter/tmeter1.html>*

[20] *ESCOT Grant <http://wise.sri.com/escot/>*

used in various educational settings: at UniHill Elementary School in a 4th and 5th grade life sciences class, at Centennial Middle School in the context of their Computer Club, at New Vista High school in a “Protest and Reform” history class and an “Exposing the Human Grotesque” languages and arts class, and at the Jiva Institute in India. In some of these cases, such as the UniHill life-sciences application (Section 3.1.1), the language employed by AgentSheets had to be modified into a more domain oriented language to suit the needs of the topic area. While the general-purpose VAT language may not always be appropriate for specific domains and for use by young elementary school students, the system allows the extension as well as modification of the existing language to adapt it to the needs of the specific situation. At this point, however, end-users cannot customize or expand the VAT language themselves, since some Lisp expertise is required.

Moving towards a perspective of the computer as a communication medium enhances the educational value of the computer by changing it from a tool that is used in solitary to define and alter new objects to a medium that can be used to communicate in new ways with a vast number of people. Learning is essentially a communicative and social process, and we have found that thinking of AgentSheets as a medium has allowed us to expand its educational value by making enhancements that have increased its interactivity, richness and accessibility. We believe that viewing all educational software as media will eventually lead to the establishment of an even richer, interactive and more accessible educational media that will allow the easy combination of different kinds of educational software components. When such an educational object economy exists, the computer will begin to realize its true value as an educational medium.

Acknowledgments

We would like to thank Rich deLaura and Jeremy Rochelle for stimulating the SimCalc and AgentSheets combination and for providing invaluable conceptual support as well as concrete implementation effort.

References

Ambach, J., and Repenning, A. (1996). *Puppeteers and Directors: Supporting Artistic Design by Combining Direct-Manipulation and Delegation*. Proceeding of the Second International Symposium on Creativity and Cognition, (Loughborough, U.K.), 67-76. LUTCHI Research Center, Loughborough.

diSessa, A.A. (1991). *An Overview of Boxer*. Journal of Mathematical Behavior, , (10), 3-15.

Eisenberg, M., and Fischer, G. (1994). *Programmable Design Environments: Integrating End-User Programming with Domain-Oriented Assistance.* Proceedings of the 1994 ACM CHI Conference, (Boston, MA), 431-437. ACM Press: New York.

Fischer, G. (1994). *Domain-Oriented Design Environments*. (Ed.), Automated Software Engineering (pp. 177-203). Kluwer Academic Publishers: Boston, MA.

Gilmore, D., Pheasey, K., Underwood, J., and Underwood, G. (1995). *Learning graphical programming: An evaluation of KidSim*. Proceedings of the Fifth IFIP Conference on Human Computer Interaction, (London).

Gindling, J., Ioannidou, A., Loh, J., Lokkebo, O., and Repenning, A. (1995). *LEGOsheets: A Rule-Based Programming, Simulation and Manipulation Environment for the LEGO Programmable Brick*. Proc. of Visual Languages Conference, (Darmstadt), 172-179. IEEE Computer Society Press.

Laurel, B. (1993). *Computers as Theater*. Addison-Wesley Publishing Company: Reading, MA.

Mok, C. (1996). *Designing Business*. Adobe Press: San Jose, CA.

Norman, D.A. (1986). *Cognitive Engineering*. (Ed.), User Centered System Design (pp. 31-61). Lawrence Erbaum Associates, Publishers: Hillsdale, NJ.

Papert, S. (1980). *Mindstorms: Children, Computers, and Powerful Ideas*. Basic Books Inc.: NY.

Papert, S., and Harel, I. (Ed.). (1993). *Constructionism*. Ablex Publishing Corporation: Norwood, NJ.

Rader, C., Brand, C., and Lewis, C. (1997). *Degrees of Comprehension: Children's Understanding of a Visual Programming Environment*. Proceedings of the 1997 Conference of Human Factors in Computing Systems, (Atlanta, GA), 351-358. ACM Press.

Rader, C., Cherry, G., Brand, C., Repenning, A., and Lewis, C. (1998). *Principles to Scaffold Mixed Textual and Iconic End-User Programming Languages*. To appear in Proceedings of the 1998 IEEE Symposium of Visual Languages, (Nova Scotia, Canada). Computer Society.

Repenning, A. (1993). *Agentsheets: A Tool for Building Domain-Oriented Dynamic, Visual Environments*. Department of Computer Science, University of Colorado at Boulder.

Repenning, A. (1994). *Programming Substrates to Create Interactive Learning Environments*. Journal of Interactive Learning Environments, Special Issue on End-User Environments, 4 (1), 45-74.

Repenning, A. (1995). *Bending the Rules: Steps toward Semantically Enriched Graphical Rewrite Rules*. Proceedings of Visual Languages, (Darmstadt, Germany), 226-233. IEEE Computer Society.

Repenning, A., and Ambach, J. (1996). *Tactile Programming: A Unified Manipulation Paradigm Supporting Program Comprehension, Composition and Sharing*. Proceedings of the 1996 IEEE Symposium of Visual Languages, (Boulder, CO), 102-109. Computer Society.

Repenning, A., and Ambach, J. (1997). *The Agentsheets Behavior Exchange: Supporting Social Behavior Processing*. CHI 97, Conference on Human Factors in Computing Systems, Extended Abstracts, (Atlanta, Georgia), 26-27. ACM Press.

Repenning, A., and Ioannidou, A. (1997). *Behavior Processors: Layers between End-Users and Java Virtual Machines*. Proceedings of the 19967IEEE Symposium of Visual Languages, (Capri, Italy). Computer Society.

Repenning, A., Rausch, M., Ioannidou, A., and Phillips, J. (1998). *Using Agents as a Currency of Exchange between End-Users*. WebNet, (Orlando, Florida). Association for the Advancement of Computing in Education (AACE).

Repenning, A., and Sumner, T. (1994). *Programming as Problem Solving: A Participatory Theater Approach*. Workshop on Advanced Visual Interfaces, (Bari, Italy), 182-191.

Repenning, A., and Sumner, T. (1995). *Agentsheets: A Medium for Creating Domain-Oriented Visual Languages*. IEEE Computer, 28, (3), 17-25.

Resnick, M. (1994). *Behavior Construction Kits*. Communications of the ACM, 37, (7), 65, 71.

Resnick, M., Martin, F., Sargent, R., and Silverman, B. (1996). *Programmable Bricks: Toys to Think With*. IBM Systems Journal, Vol. 35, (3 & 4), 443-452.

Rosson, M.B., and Carroll, J.M. (1996). *The Reuse of Uses in Smalltalk Programming*. ACM Transactions on Computer-Human Interaction, 3, (3), 219-253.

Simon, H.A. (1981). *The Sciences of the Artificial*. MIT Press: Cambridge, MA.

Spohrer, J., Sumner, T. and Buckingham Shum, S. (1998). *Educational Authoring Tools and the Educational Object Economy: Introduction to this Special Issue from the East/West Group*. Journal of Interactive Media in Education, 98 (10). <http://www-jime.open.ac.uk/98/10>

Venditto, G. (1997). *Live Pages: 7 "Instant" Java Tools Let You Bring Multimedia to the Web*. In Internet World Magazine.

Vygotsky, L.S., and Kozulin, A. (1996). *Thought and Language*. MIT Press: Cambridge, MA.

Vygotsky, L.S., and Vygotsky, S. (1980). *Mind in Society: The Development of Higher Psychological Processes*. Harvard University Press: Cambridge, MA.