Collaborative Use & Design of Interactive Simulations

Alexander Repenning, Andri Ioannidou, Jonathan Phillips

University of Colorado, Center for LifeLong Learning & Design

Abstract: Interactive simulations hold great potential as a communication vehicle capable of improving the usefulness of technology in education. While some benefit can be gained by simply using pre-built simulations, learners benefit most from designing all or at least some aspects of their own simulations. The challenge is to enable this design-as-learning activity without turning students into programmers. A component-based approach cannot only simplify the design of interactive simulations but at the same time serves as collaboration-enabling technology connecting students, teachers, publishers, and researchers. A general framework called the Use & Design Spectrum is introduced to conceptualize collaboration issues of simulation use and design. The AgentSheets simulation-authoring tool is used to provide specific examples of collaborations.

Keywords: agents & intelligent systems, end-user programming, interactive simulations, use vs. design, collaboration, multimedia

Introduction

The notion of *interactive simulations* is quickly gaining importance as a means to explore, comprehend and communicate complex ideas [Turkle 1995]. What makes a bridge collapse, how does a virus spread, why is the sky blue, what is the foundation of a stable Ecosystems? These are just some examples out of an infinite universe of questions that can be explored with simulations. Similar to video technology, learners can observe complex dynamic processes unfold over time but in contrast to video technology, learners can more actively interact with simulations and play hands-on "what if" games. The crucial combination of affordable computer hardware, high-resolution graphics, and Internet connectivity creates a rich environment, in which people can run, share and build interactive simulations. Simulations can also be featured as Java applets that can be accessed remotely through Web browsers or can be featured as JavaBean components that can be freely combined with other educational components into rich learning activities.

Interactive simulations are gaining momentum in education. A recent and highly publicized ETS study [Wenglinsky 1998] concluded that drill-and-practice technology has turned out to be largely ineffective, and that simulation technology based on constructivist learning principles [Yager 1995] provides measurable learning advantages. In a report [Shaw 1997] on educational technology to the President of the USA, a committee of science advisors presented the most promising constructivist applications of technology, with simulations on top of the list. The time is ripe to not only intensify research but also to create effective simulation-authoring tools and most importantly to design usable simulation content.

Who designs educational simulations and how are they used? Should students, teachers, publishers or researchers build simulations? From an educational perspective there is a high potential for learning if students design their own simulations. This actively engages learner in the process of model building and active inquiry. Unfortunately, designing simulations and implementing them with traditional programming languages is a hard task [Rader, Brand, & Lewis 1997]. In the past we have explored the use of new end-user programming paradigms such as Graphical Rewrite Rules [Repenning 1994; Repenning 1995; Smith 1996; Smith, Cypher, & Spohrer 1994], analogies [Craig 1997; Perrone & Repenning 1998] and Tactile Programming [Repenning & Ambach 1996] to significantly lower the threshold of programming. While end-user programming is a highly effective means to build tailored simulations, there is a need for a more general framework to use and design simulations.

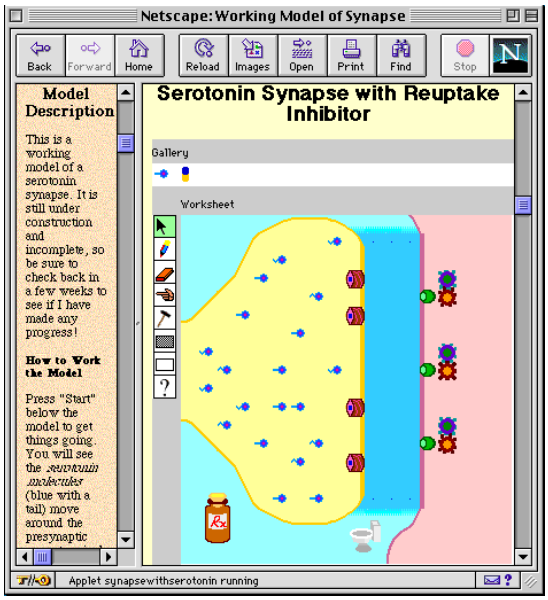
This paper briefly presents the AgentSheets simulation-authoring tool, introduces the *Use <=> Design Spectrum* framework and provides elaborate discussions of technology-enhanced collaboration in education in the context of concrete cases.

AgentSheets: A Simulation Component Authoring Tool

Combining Java authoring, end-user programmable agents and spreadsheet technology, AgentSheets [Repenning & Sumner 1995; Repenning, 1998] (http://www.agentsheets.com) is an authoring tool that empowers casual computer users with no formal programming training to build and publish Web-based interactive simulation components. It enables users to create such simulation components, using its end-user programming language Visual AgenTalk (VAT). Users design agents' looks by drawing icons and create agents' behaviors by composing VAT conditions and actions from command palettes into rules. The palettes for conditions and actions include a wide variety of elements. For example, VAT provides conditions that check the appearance of agents and test the values of an agent's variables, and actions that set these variables to formulae, make sounds, change the appearance of neighboring agents, destroy agents, and create new agents. It employs a new approach to end-user programming called Tactile Programming. Tactile Programming primitives and programs not only have enhanced

Tactile Programming is well suited for collaborative use since it eases *composition, comprehension* and *sharing* of behaviors [Repenning & Ambach 1996].

A wide spectrum of users, ranging from elementary school students with no programming background to scientists have used AgentSheets and VAT to create interactive simulations and games in a variety of disciplines, including computer science, environmental design, fine art, robotics, music, history, and biology. In an elementary school science class, students created ecosystem simulations to explore food webs and sustainability issues. In a high school history class, students created social and historical simulations. Scientists working with NASA created simulations of E.coli bacteria fermenting in zero gravity.



Agent	Name	Description
	Serotonin Molecule	Moves in Presynaptic Nerve Terminal and is released into Synaptic Space.
	Reuptake Inhibitor	Forcibly pumps Serotonin back to Presynaptic Nerve Terminal.
	Prozac	Binds to Reuptake Inhibitors and blocks their ability to reuptake Serotonin.
	Serotonin Receptor	Activates Intracellular Machinery when Serotonin binds to it.
	Intracellular Machinery	Stimulates excitement and happiness.
	Medicine Bottle	Releases Prozac tablets into Synaptic Space.

Figure 1: Simulation running in a Web page explaining how Serotonin works in the Synapse and how Antidepressants affect the system (left) and agent descriptions for the Serotonin Synapse Simulation (right).

Figure 1 shows an interactive simulation explaining the effects of Prozac by modeling a serotonin synapse in the brain. The simulation was built by a psychiatrist for his patients. For this particular simulation, a suite of agents was created by specifying their look and behavior using the Visual AgenTalk language. Figure 2 illustrates two VAT rules from the behavior of the serotonin molecule defining animation and interaction with membranes.

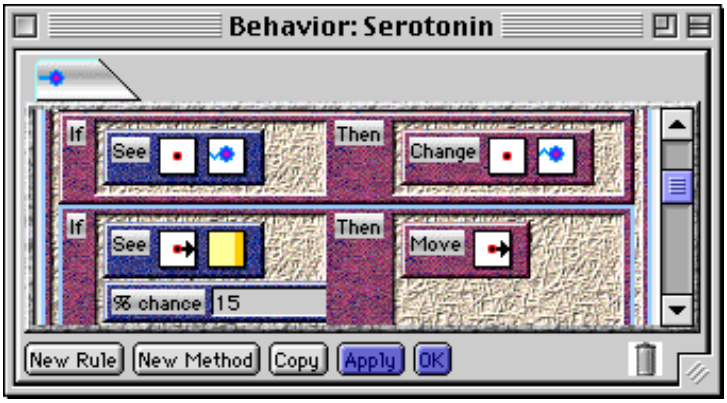


Figure 2: Part of the serotonin behavior which specifies that if the serotonin looks like , it will change to , or with 15% chance if the serotonin sees a membrane to its right, it will move that way.

The Web serves as an ideal platform to harness the communicative power of interactive simulations. Entire AgentSheets simulations can be exported as Java applets and JavaBeans, using the Ristretto™ Agent-to-Java-Byte-Code generator [Repenning & Ioannidou 1997]. This allows simulations to be published on the Web making them accessible to a larger community of users. Using Ristretto™, the serotonin simulation was turned into a Java applet (Figure 1) and was included in Web pages and can be found at http://www.csn.net/~wphillip/Synapse_applet/synapse.html.

No matter how much the programming process has been simplified through the use of new end-user programming paradigms there is a need for collaboration support. Collaboration allows people to use other people's simulations and learn how to design their own by using similar existing simulations. Also, collaboration enhances a dialog between designers and users,

enabling the design of domain oriented end-user programming languages [Cherry, Ioannidou, Rader, Brand, & Repenning 1999; Rader, Cherry, Brand, Repenning, & Lewis 1998] and resulting in more effective use of educational technology. To make simulations a feasible part of education and foster some kind of simulation literacy, there is a need to employ a more general research framework moving beyond issues of programming and computer applications. In the next section we introduce such a framework called the *Use <=> Design Spectrum.*

The Use <=> Design Spectrum

The *Use <=> Design Spectrum* describes a conceptual space of stepping stones between designing simulations from scratch and using existing simulations. One can think of ways to break simulations up into *components* allowing users to trade off between the use and the design of simulations. These simulation components become collaboration-enabling artifacts that are used, combined, shared, and extended by the members of an educational community including students, teachers, publishers and researchers. Figure 2 below shows a number of points along the *Use <=> Design Spectrum* corresponding to research projects and commercial applications.

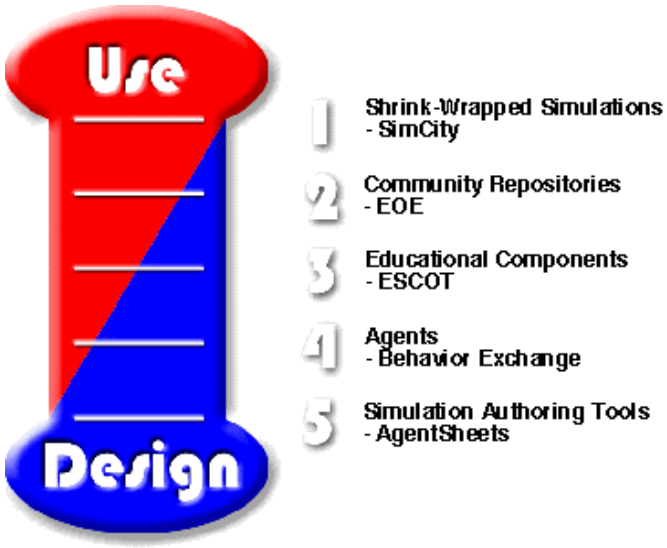


Figure 2. A graphical representation of our Use <=> Design Spectrum. The upper part of the figure shows ready-to-use simulations such as SimCity™; such simulations cannot be modified by the user—they are all "use" and no "design." At the bottom we have sophisticated design tools such as AgentSheets; someone must use the tools to design a simulation before it can be used. Other projects fall somewhere in between.

Level 1: Shrink-Wrapped Simulations (SimCity)

Shrink-wrapped packages such as SimCity™ are highly interactive simulations aimed primarily at entertainment, rather than education. In spite of this, these products certainly have significant educational aspects. For example, one possible objective of a SimCity player is to build the largest possible city by balancing taxes, crime rate, pollution and many other factors. However, the general lack of programmability is a major limitation of SimCity and similar packages. That is, *there is no end-user design*. These simulations have a single format and structure; they cannot be adapted to the needs of individual teachers and students.

Use	Design
Play the simulation game	Change simulation parameters, e.g, change the tax rate in SimCity

Level 2: Community Repositories (EOE)

Community repositories are collections of simulations and other interactive educational objects. The Educational Object Economy (EOE) is a collection of educational *Java applets.* One author of this paper is a founding member of the EOE organization (http://www.eoe.org) and was a member of the NSF-funded East-West Consortium that explored educational authoring tools and models of collaboration between industry, academia and publishers. Educational objects in the EOE are annotated with so-called meta information that categorizes objects in terms of subject domains, and includes information on how to use the object.

Use at this level consists of the locating and selection of relevant educational objects. *Design is very limited*. EOE objects include Java source code allowing professional programmers to re-design objects by modifying their program. However, the adaptation of educational objects in the EOE requires the mastery of general Java programming environments; these skills are not common among most students and teachers.

Use <=> Design levels are interconnected. Design tools associated with one level can be used to create artifacts used at some other level. Some EOE Java applets (level 2) have been created by students using the AgentSheets simulation-authoring tool (level 5). For instance, students at the Jiva Institute in India used AgentSheets to create simulations that they then uploaded to the EOE to make them accessible to other people.

Level 3: Educational Components (ESCOT)

The Educational Software Components of Tomorrow (ESCOT) project can be considered a conceptual extension of the EOE work. ESCOT explores technical mechanisms and collaboration models that let people efficiently combine *components* based on Sun's *JavaBeans* technology into interactive middle school math curriculum activities. In contrast, the educational objects in the EOE were created in isolation; there was no way to combine objects into more meaningful units. ESCOT users combine simulations built in AgentSheets or other JavaBean component generators, such as Java Geometer SketchPad, with other components such as SimCalc graphs into rich activities (e.g., Figure 3). In the case of an AgentSheets ecosystem simulation, students could track the number of individuals of each species in SimCalc to explore issues of ecosystem sustainability and other concepts.

Use	Design
Select relevant components based on their functionality and interfaces	Connect the components to build complete application in ways analogous to connecting electronic components into a working circuit.

Existing design tools at the component level, such as Sun's BeanBox, are crude, but more accessible than EOE programming tools. The ESCOT project focuses on finding new ways to simplify the design process by letting students and teachers assemble components.

Level 4: Agents (Behavior Exchange)

The Behavior Exchange allows users to exchange individual simulation agents. For instance, in a city pollution simulation, people can exchange agents such as cars, roads, trains, and factories. Agents downloaded from the Behavior Exchange are "glass boxes" that can be readily opened up to inspect their rules and modify their behavior. Modified agents can be uploaded again to the Behavior Exchange. This mechanism allows a community of users to build and incrementally improve simulation content.

Use	Design
Selecting relevant agents based on their descriptions and project context	Modify the behavior of agent, e.g., make a car stop at a red traffic light Modify look of agent; e.g., change color of agent Combine set of agents into new simulation

This ability to build simulations by combining and modifying agents makes the agent level ideal for supporting scaffolding [Guzdial 1994].

Level 5: Simulation Authoring Tools (AgentSheets)

AgentSheets is a simulation-authoring tool that allows non-specialists to build complex interactive simulations and deliver them as Java applets and JavaBeans over the Web.

Use	Design
Run simulations	Build agent behavior from scratch by composing conditions and actions into rules Design agent looks by drawing icons

The AgentSheets development environment has been at the core of numerous projects at major research institutions around the world. The research that led to the development of AgentSheets was mostly focused on end-user programming and exploring new programming paradigms for non computer-specialists.

The rest of this paper exemplifies the Use <=> Design Spectrum with specific cases of using technology in education at the levels of Educational Components, Behavior Exchange and Authoring Tools. The focus will be on component granularity, use, design, and their role in collaboration between students, teachers, publishers and researchers.

Educational Components: Assembling Activities

Educational components explored in the Educational Software Components of Tomorrow (ESCOT) project are at level 3 in the Use <=> Design spectrum. ESCOT builds on lessons learned from the EOE project. The EOE endeavored to create a Web-based library of educational software objects. The EOE Web site contains over 1000 Java applets designed for educational use. However, it is becoming clear that a library of objects is not enough to facilitate large-scale software reuse [Udell 1994]. Components must be designed for reuse within an integration framework that maintains consistency and makes use of standards. Without this kind of forethought, objects may be left on the mantel to gather cobwebs. The EOE project provided many insights into developing large-scale storage and retrieval systems of educational objects. ESCOT hopes to improve the quality of the objects stored through a shared conceptual framework employing educational standards.

The ESCOT project has been launched with the goal of facilitating the creation of technologically-enhanced educational content. Software components offer the promise of interconnecting currently existing islands of educational technology research. The vision of ESCOT is to design a technical framework as well as develop educational standards for 'plug together' educational components. With the framework and standards in place, third-party developers may contribute components designed for general assembly by educators and publishers. ESCOT is partnered with a subset of developers to initially seed this library of components and co-design the framework and standards under which they can be integrated.

By providing a framework where islands of research can be integrated, ESCOT hopes to provide a testbed where educational activities can be assembled from a suite of components ranging from geometry exploration tools, to graphing systems, to interactive simulations. The testbed introduces a collaboration model called Integration Teams consisting of teachers and developers jointly building interactive activities.

If independent pedagogical software entities are properly encapsulated into embeddable objects (embeddable into media such as html), then educators can incorporate them into stand-alone activities. The Internet provides a medium in which educators, publishers, and software developers may collaborate to assemble activities and provide feedback for developers. Furthermore, components and finished activities may be distributed easily online. However, incorporating educational software components into an instructionist activity does not always produce optimal learning content. Often, interaction between components is desirable. For example, a simulation might benefit from a graph representation of its internal values. Therefore, a collaborative environment where components can not only be embedded into instructionist wrapping, but also wired together is needed to create the best possible educational content. Furthermore, wiring components should be accessible to educators as well as developers.

Example: SimVirus

An example best illustrates the ESCOT collaborative framework employing educational components. The AgentSheets simulation-authoring tool can generate ESCOT-friendly JavaBean components. This allows an AgentSheets simulation (such as SimVirus, shown in the upper left corner of Figure 3) to connect to a SimCalc graphing utility (shown in the bottom left corner of Figure 3) which has also been extended to integrate into the ESCOT architecture.

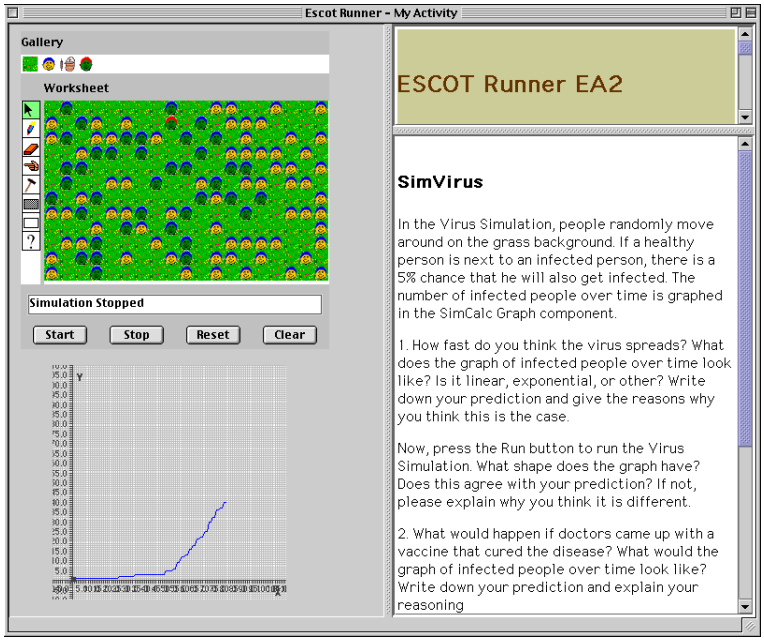


Figure 3: ESCOT activity featuring an AgentSheets simulation (upper left), SimCalc grapher (lower left), and an activity description (right)

The SimVirus simulation contains people that randomly move around on the grass background. If a healthy person is next to an infected person, with a 5% chance he will also get infected. Doctors also move around randomly in the community and heal sick people. The rate at which the virus spreads is difficult to see from the AgentSheets simulation alone. To better illustrate this concept, the number of infected people over time is graphed in the SimCalc component. By connecting the AgentSheets component to the SimCalc component an educational concept, the exponential spread of a virus, is better illustrated.

Collaboration

The rapidly growing collection of ESCOT components includes spreadsheets, databases, simulations, and geometry exhibits. The success of ESCOT depends on its ability to foster collaboration between teachers and developers. Developers and teachers work together to explore different integration frameworks and agree on a proposed framework, which will help facilitate large-scale integration. ESCOT partners will originally contribute seed components to the library. However, the growth of the component library will depend on third-party contributors. The elements of the framework which withstand the test of large-scale use will be adopted. Since ESCOT cannot survive without maintained interest from developers, it will accept modifications to the framework that expedite development efforts. So far, ESCOT has succeeded in generating initial development interest.

A key element in the ESCOT vision is a new division of labor in constructing technology-enhanced educational content. In the traditional sense, technical developers, with instructions from educators, create educational software. This places an extraordinary burden on developers while limiting the amount educators and publishers may contribute. ESCOT hopes to fuse educators and publishers into the development process by placing them in the driver's seat as the final assemblers of content. This allows technical developers to focus more specifically on technical content. Overall a more efficient, deeper level of collaboration can be achieved where output from contributors can more easily be combined and interchanged. The result should be more useful educational content.

By assembling ESCOT content, educators and publishers become users of educational components and designers of educational activities. Students are users of the finished content. Supporting the Use <=> Design spectrum allows teachers to interact with technology at their desired level of comfort. Where teacher comfort tapers off, developers may support them.

Limitations

Making software reuse work is a hard problem [Fichman & Kemere 1997]. We believe that if 90% of a component is useful and the remaining 10% need to be changed but cannot be changed by its user, then the component is 100% unusable. In the case of the EOE, users typically need to edit Java source to modify components. Within ESCOT, some modification is possible though JavaBean customization interfaces. If additional changes must be made, this can be supported though end-user programming, as is the case with AgentSheets-generated JavaBean components.

Behavior Exchange: Sharing Agents

At the educational component level, users can share entire simulations or connect simulations with relevant components. However, large components such as simulations need to be broken up into finer sub-components to increase reuse and adaptation. AgentSheets simulation components can be broken up into agents that can be shared and reused through the Behavior Exchange to promote collaboration. The Behavior Exchange is at level 4 of the Use <=> Design spectrum.

The Behavior Exchange [Repenning & Ambach 1997; Repenning, Ioannidou, Rausch, & Phillips 1998] is an AgentSheets-specific sub-component repository (http://www.agentsheets.com/behavior-exchange.html) that allows users to collaborate by exchanging agents that have been created in AgentSheets. It is an evolving Web-based information space (Figure 4) where users locate interesting agents and acquire the ones that seem relevant by copying them into the design environment. Users can then proceed to use a downloaded agent to comprehend what it does, evaluate usefulness, and decide whether to reuse it as is or modify it.

Agents acquired from the Behavior Exchange are not "black boxes" that can only be executed. Instead, they are "glass boxes" that can be readily opened to inspect their rules and modify their behavior, as the full specification of the agents' behavior comes along with them when they are downloaded. Modified agents can be uploaded again to the Behavior Exchange. This mechanism allows a community of users to build and incrementally improve simulation content. The ability to build simulations by combining and modifying agents makes the agent level ideal for supporting collaboration among users, whether they reside in the same physical location or not, and the scaffolding of the simulation creation process.

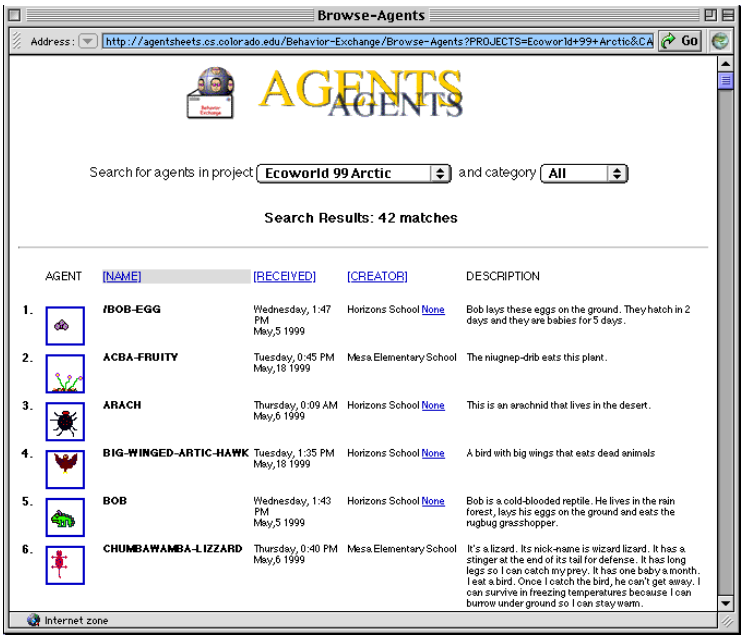


Figure 4: The Arctic EcoWorld Agents in the Behavior Exchange.

Building Social Simulations using the Behavior Exchange

The Behavior Exchange has been used in a variety of educational settings, both for enabling collaboration among the students in a group and for scaffolding the simulation construction process. In elementary schools, the Behavior Exchange enabled the collaborative creation of EcoWorlds simulations [Cherry, et al. 1999]. Students worked on individual animals and through the Behavior Exchange (Figure 4) combined them in ecosystems to explore issues such as food webs and sustainability. In high schools, the Behavior exchange was used to collaboratively create social simulations. The latter is described in this section.

At the New Vista High School, students used AgentSheets to create simulations as part of a Protest and Reform history class. In this class, students have an opportunity to study protest movements throughout United States history (e.g., the Civil Rights movement and the anti-Vietnam war movement), and to learn about theories of protest and social change. Although it is more common to use simulation technology in math and science classes, the teacher wanted to incorporate AgentSheets in his class to enable his students to go beyond the traditional posterboards and instead engage in an in-depth exploration of selected topics [Ioannidou, Repenning, & Zola 1998].

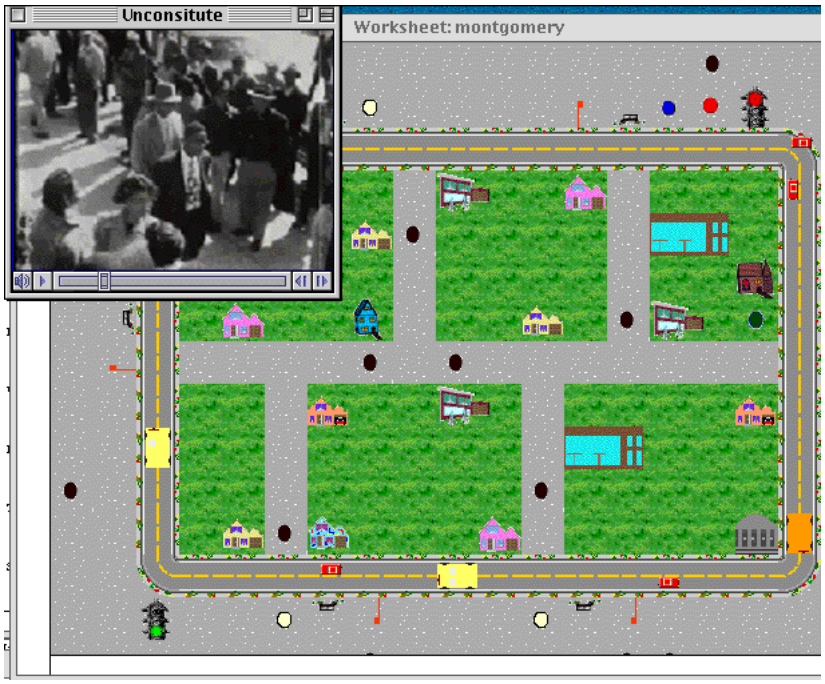


Figure 5: The Montgomery Bus Boycott simulation, which combines simulation with video.

During the second iteration of using AgentSheets in this class, all the students were required to create simulations as their final project. They had the choice, however, to create a physical simulation (a board game) or a computer simulation using AgentSheets. The projects using computer simulation spanned a variety of topics: the Montgomery Bus Boycott (a project about the bus boycotts resulting from Rosa Parks' refusal to give up her seat to a white person in 1955) (Figure 5), The Fling Strike (a project about the United Autoworkers sit-down strike of Flint, Michigan in 1936), and the Ludlow Massacre (a project about the massacre of miners in Ludlow, Colorado at the beginning of the century). Providing students access to the Behavior Exchange gave them an in-group collaboration tool and opened up a resource pool that they could utilize to get ideas and reuse either looks or behaviors of existing agents in their own simulations. Moreover, it scaffolded the simulation building process and avoided some of the struggles the students of the previous year had to go through, having nothing to base their work on.

Collaboration

Student-Student Collaboration: The Protest and Reform students used the Behavior Exchange as a means to collaborate within their groups. When different students of the same group were working on different machines and at different times, they used the Behavior Exchange as a means to merge the agents each of them was working on into a single group simulation. For example, the students working on the Ludlow simulation distributed the workload to the various members of the group. One was working on the strikers, the other on the militia and the third on the coal officials. They then merged their work into a single simulation using the Behavior Exchange.

The Behavior Exchange enables distance education. The fact that it is Web-based affords not only collaboration within a class of students in the same physical space, but also collaboration among students from different schools or even different countries.

For all projects, the Behavior Exchange proved to be a useful resource for finding and reusing simulation sub-components. Students located and selected agents, such as grass, houses, roads, cars and trains. In some cases, students used the agents as they were and combined them with their own to create new simulations, or they customized the agents by modifying either their look or behavior. For example, the students working on the Montgomery Bus Boycott simulation reused both the look and behavior of car and traffic-light agents, reused the moving behavior of cars for their buses, and reused the looks of buildings and roads. Having a base to start with, such as the example agents found in the Behavior Exchange, is a great instrument for scaffolding the simulation building process, since building simulations from scratch is difficult. Scaffolding [Guzdial 1994] in this context is a process enabling students to go through a smooth transition from using existing simulation sub-components to designing their own. This is also a form of what Laurel [Laurel 1992] calls, *time-displaced collaboration* between AgentSheets designers and users. AgentSheets designers upload their creations on the Behavior exchange and users locate them and use them at different times and places.

Student-Teacher-Researcher Collaboration: Teachers know a lot about educational content, but often know very little about technology. Technology researchers, on the other hand, know a lot about technology, but very little about the content. In isolation, each party has a difficult time envisioning what an educationally valuable simulation activity might look like. Complex design problems, such as creating these social simulations, require more knowledge than any single person can possess, as the knowledge relevant to the problem is distributed [Fischer 1999]. Bur rather than viewing this "symmetry of ignorance" [Rittel 1984] (or "asymmetry of knowledge") as an obstacle during design, it can be viewed as an opportunity for creativity. To account for "the symmetry of ignorance", the teacher and the researchers held "design sessions" with each of the groups to work with them in mapping the chosen topic to the capabilities of the technology. These sessions proved to be a very important preparatory activity for the students, as they engaged in brainstorming activities to find a good mapping of their existing knowledge of the topic to a computer simulation. The process of mapping one kind of representation (e.g., historical information) onto another (e.g., agent behaviors and interactions) may support student construction of knowledge about the topic [Ioannidou, et al. 1998]. This collaborative approach to the design of components is also reflected in the notion of Integration Teams within the context of ESCOT (please refer to section "Educational Components").

Simulation Authoring Tools: Building Simulations from Scratch

Repositories such as the Behavior Exchange can be a useful resource of reusable components. Users, however, cannot always rely on other people for creating simulations components or sub-components (namely, agents) that are relevant or interesting to them. Being a consumer of such components may work in certain cases and to a certain extend, but users need to have the ability to create such artifacts from scratch by themselves. Authoring tools employing end-user programming techniques and languages are needed for supporting the design of simulation components and sub-components.

AgentSheets is one such system that provides users with the ability to create simulation components from scratch. Moreover, an entire AgentSheets simulation can be exported as a Java applet and/or JavaBean, using the Ristretto™ Agent-to-Java-Byte-Code generator, and be published on the Web making it accessible to a larger community of users. AgentSheets-generated Java applets have been featured in the EOE, whereas AgentSheets-generated JavaBeans constitute one of the interoperable components in the ESCOT project.

Building Social Simulations from Scratch

As mentioned before, AgentSheets has been used in a Protest and Reform history class to build social and historical simulations. During the first iteration of the project [Cherry, et al. 1999; Ioannidou, et al. 1998], the two groups of students that chose to create AgentSheets simulations as their final projects, created their simulations from scratch. Having no predecessors in this experiment and with no social or historical AgentSheets simulations in existence, the students were left at the extreme end of the spectrum: *design*. Not only did they not have any existing simulations to use as a starting point, to draw ideas from, the Behavior Exchange was not yet in place at the time to enable the students to reuse agents behaviors or looks.

Nevertheless, the students proceeded to create what would be the first social and historical simulations in AgentSheets. One of the groups, which consisted of three girls initially intimidated by technology, selected the topic of the California Grape Boycott. The students created a Web page with a boycott simulation applet built in AgentSheets, as well as historical information on the boycott and links to related Web sites (Figure 6). The second group decided to create a simulation to explore what happens in peaceful protest marches that turn violent and involve police confrontations.

Figure 6: The Grape Boycott project Web page includes (1) descriptions of the agents, (2) historical background and related links, and (3) the simulation applet.

As the New Vista students developed their simulations, they also used the Java capabilities of AgentSheets to create applets and embed them in Web pages containing historical information about the subject and links to related Web sites. These Web pages provide a critical connection between the course content and the simulation technology — a simulation consisting of brightly colored icons moving on a screen does not convey much meaning to its intended audience unless the creators of the simulation situate it in an informative context. Both projects can be accessed at http://www.cs.colorado.edu/~l3d/systems/agentsheets/New-Vista/.

Collaboration

At this level, collaboration was not technology-supported, but people-supported. Students worked in groups to develop their simulations, while teachers and researchers served as mentors helping students break up the workload, providing them pointers to content information, and guiding them in creating the simulation.

Student-Student Collaboration: Students created their simulations in groups. This afforded the students various types and levels of collaboration. For example, we observed that the initially intimidated by the computer students found the task of creating a simulation less daunting when they all worked together, sharing ideas and helping each other out with the programming. Moreover, the dual tasks of creating the Web site and building the simulation allowed members of the group to distribute the workload among themselves according to their individual interests. At the same time, communication among the group members working on the different tasks needed to be maintained, for the group to produce a coherent final artifact. Finally, collaboration allowed the group to create a more complete project than any individual could have produced alone.

Conclusions

Interactive simulations hold great potential as a communication vehicle capable of improving the usefulness of technology in education. The challenges faced to effectively create simulation literacy in education are tremendous when just using pre-built simulations and even more so when designing new simulations to fit school curriculum. Research needs to move from the exploration of issues resulting from individual people using individual computer systems towards the exploration of collaborative issues. How should students, teachers, publishers, and researchers work together to create effective teaching material and engaging learning environments? In addition to social issues, how can technology support collaboration? This paper has introduced a conceptual framework called the *Use <=> Design Spectrum* populated with a number of ongoing research projects. Together, these projects provide a set of necessary stepping-stones between the *use* and the *design* of interactive simulations. Users including students, teachers, publishers, and researchers can move gradually from using pre-built simulations towards designing their own simulations by reusing increasingly fine grained components that are collected and organized by a community of users.

Acknowledgements

The authors would like to thank John Zola, teacher at New Vista High School, and his students. The Protest and Reform project was conducted under NSF AAT REC-9631396; ESCOT is supported by NSF grant REC 9804930; and the Behavior Exchange is supported by NSF SBIR DMI 9761360.

References

Cherry, G., Ioannidou, A., Rader, C., Brand, C., & Repenning, A. (1999). Simulations for Lifelong Learning. In Proceedings of National Educational Computing Conference (NECC) 1999. Atlantic City, NJ.

Craig, B. (1997). Behavior Combination Through Analogy. In Proceedings of the 1997 IEEE International Symposium on Visual Languages, (pp. 270-273). Capri, Italy: IEEEE Computer Society.

Fichman, R. G., & Kemere, C. F. (1997). Object Technology and Reuse: Lessons from Early Adopters. IEEE Computer, 30(10, October).

Fischer, G. (1999). Symmetry of Ignorance, Social Creativity, and Meta-Design. To appear in Proceedings of Creativity and Cognition 3 - Intersections and Collaborations: Art, Music, Technology and Science. Loughborough, UK.

Guzdial, M. (1994). Software-Realized Scaffolding to Facilitate Programming for Science Learning. Interactive Learning Environments.

Ioannidou, A., Repenning, A., & Zola, J. (1998). Posterboards or Java Applets? In A. Bruckman, M. Guzdial, J. Kolodner, & A. Ram (Ed.), International Conference of the Learning Sciences 1998, (pp. 152-159). Atlanta, GA: Association of the Advancement of Computing in Education.

Laurel, B. (1992). Notes on Characters, Agents, and Narrative Play. In AAII'92: Workshop on Artificial Intelligence and Interactive Entertainment.

Perrone, C., & Repenning, A. (1998). Graphical Rewrite Rule Analogies: Avoiding the Inherit or Copy & Paste Reuse Dilemma. In Proceedings of the 1998 IEEE Symposium of Visual Languages, (pp. 40-46). Nova Scotia, Canada: Computer Society.

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

Rader, C., Cherry, G., Brand, C., Repenning, A., & Lewis, C. (1998). Principles to Scaffold Mixed Textual and Iconic End-User Programming Languages. In Proceedings of the 1998 IEEE Symposium of Visual Languages, (pp. 187-194). Nova Scotia, Canada: Computer Society.

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. In Proceedings of the 1995 Symposium of Visual Languages, (pp. 226-233). Darmstadt, Germany: IEEE Computer Society.

Repenning, A., & Ambach, J. (1996). Tactile Programming: A Unified Manipulation Paradigm Supporting Program Comprehension, Composition and Sharing. In Proceedings of the 1996 IEEE Symposium of Visual Languages, (pp.

102-109). Boulder, CO: Computer Society.

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

Repenning, A., & Ioannidou, A. (1997). Behavior Processors: Layers between End-Users and Java Virtual Machines. In Proceedings of the 1997 IEEE Symposium of Visual Languages, (pp. 402-409). Capri, Italy: Computer Society.

Repenning, A., Ioannidou, A., Rausch, M., & Phillips, J. (1998). Using Agents as a Currency of Exchange between End-Users. In Proceedings of the WebNET 98 World Conference of the WW, Internet, and Intranet, (pp. 762-767). Orlando, FL: Association for the Advancement of Computing in Education.

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

Rittel, H. (1984). Second-Generation Design Methods. In Cross (Eds.), Developments in Design Methodology (pp. 317-327). New York: John Wiley & Sons.

Shaw (1997). President's Committee of Advisors on Science and Technology, Report to the President on the Use of Technology to Strengthen K-12 Education in the United States. In Panel on Educational Technology.

Smith, D. (1996). Making Programming Easier for Children. Interactions, III(5), 59-67.

Smith, D. C., Cypher, A., & Spohrer, J. (1994). KidSim: Programming Agents Without a Programming Language. Communications of the ACM, 37(7), 54-68.

Turkle, S. (1995). Life on Screen, Identity in the Age of the Internet. New York: Simon & Schuster.

Udell, J. (1994), Componentware. Byte, 1996

Wenglinsky, H. (1998). Does it Compute? The Relationship Between Educational Technology and Student Achievement in Mathematics. Princeton, NJ: Educational Testing Service.

Yager, R. (Ed.). (1995). Constructivism and Learning Science. Mahway, New Jersey: Lawrence Earlbaum Assoc.

Authors' Addresses

Alexander Repenning (ralex@cs.colorado.edu)
Center for LifeLong Learning & Design, Department of Computer Science, University of Colorado at Boulder, Campus Box 430, Boulder, CO 80309-0430. Tel. (303) 492-1349, Fax (303) 492-2844.

Andri Ioannidou (andri@cs.colorado.edu)
Center for LifeLong Learning & Design, Department of Computer Science, University of Colorado at Boulder, Campus Box 430, Boulder, CO 80309-0430. Tel. (303) 492-8136, Fax (303) 492-2844.

Jonathan Phillips (phillipj@cs.colorado.edu)
Center for LifeLong Learning & Design, Department of Computer Science, University of Colorado at Boulder, Campus Box 430, Boulder, CO 80309-0430. Tel. (303) 492-8136, Fax (303) 492-2844.