Making Learning a Part of Life

Hal Eden, Mike Eisenberg, Gerhard Fischer, and Alexander Repenning

We must reconceptualize our educational system. By emphasizing learning as a new form of labor and by acknowledging coverage is impossible and obsolescence is unavoidable, new educational strategies such as the integration of working and learning become necessities rather than options. Learning cannot stop with a high school diploma or an undergraduate degree. The world is constantly evolving, creating the challenge for individuals and organizations to deal with change and for schools and universities to prepare people for change. Learning should be a lifelong process allowing people to engage in authentic activities as those activities arise in their lives as thinkers, workers, collaborators, and players.

Our approach to the design of educational tools is to strike a proper balance between constructionist and instructionist learning approaches. On the one hand, the constructionist approach permits learners to engage in self-directed learning, but without guidance, learners can feel overwhelmed by an infinite number of options and a paucity of clear goals. Instructionist systems, in contrast, are designed with specific goals in mind but offer little scope for creativity.

At the Center for LifeLong Learning and Design, our goal is to employ the notion of design as a vehicle to engage learners in personally interesting and meaningful activities. Design environments developed by the center help learners to create both computational and physical artifacts through activities in which complex problems must be framed and complex alternatives evaluated. Rather than being completely open-ended, however, our design environments are domain-oriented. In contrast to general-purpose programming environments, our systems are tailored towards specific application domains; this specificity permits our systems and materials to offer a higher degree of contextualized instructional support to students.

Our focus on design leads naturally to a related focus on long-term activities. Our objective—as reflected in our systems—is not to convey specific information or narrowly defined skills as rapidly as possible. Instead, we are more interested in facilitating evocative learning situations. We believe our systems effectively complement the predominantly rational view of efficient learning. Learning situations should not be geared solely to train skills at high speed, but should also allow learners to develop a true passion for their subject resulting from the solution of self-selected problems. It is this kind of passion that is responsible for producing motivated and effective lifelong learners.

We are firmly convinced that technology by itself will not fix education and that many current technologies are limited in their usefulness by enforcing

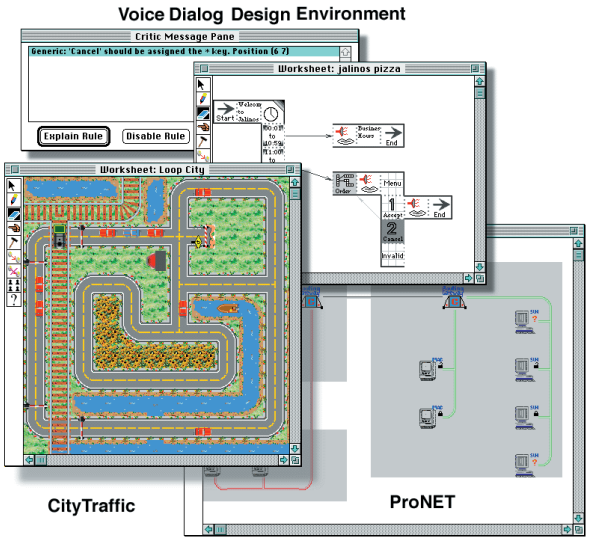

Figure 1.
Example design environments created with Agentsheets

an inadequate educational paradigm. Engagement in authentic problems implies that the choice of tasks and goals should be under the control of the user or learner, resulting in the requirement that systems are simultaneously learner-controlled and supportive. This requirement illustrates the limitation of general-purpose programming environments (such as Logo, Scheme, or SmallTalk), which fail on the supportive end, and intelligent tutoring systems, which fail on the learner-controlled end. Instead, we have developed tools to build domain-oriented design environments, such as Agentsheets, as well as specific design environments, such as HyperGami. Agentsheets and HyperGami have been used in different settings ranging from elementary school to industry.

Agentsheets: A Tool to Create Domain-Oriented Design Environments

Agentsheets is a programming substrate that has been used in the past four years to create a large number of domain-oriented design environments including simulation, visual programming, and game environments (http://www.cs.colorado.edu/~l3d/systems/agentsheets).

A typical use of Agentsheets is the design and implementation of SimCity™-like simulation games (Figure 1, CityTraffic). Allowing users to create a simulation game rather than just playing it, represents the constructionist aspect of Agentsheets. Students create their own worlds by defining the look as well as the behavior of agents. They can animate agents, make them play sounds, speak via voice synthesizers, or react to mouse, keyboard, timer, and microphone input. This activity serves as a context to learn through design (e.g., what are good ways to use traffic lights to maximize traffic throughout a city?), and to learn about design (e.g., how should complex problems be decomposed into smaller problems?).

The instructionist aspect of Agentsheets is captured in its domain-orientation and support for the creation of critiquing components. Domain-orientation, in the case of the CityTraffic application, supports the creation of road icons as well as defining the behavior of cars through semantic icon transformations and graphical rewrite rules that are geared towards the notion of flow. A critiquing component, in the case of the Voice Dialog Design Environment, built in collaboration with US West (Figure 1) allows designers to not only prototype telephone interfaces quickly, but also to verify compliance with different telephone standards and to check consistency of new designs with old ones. The ProNET system (Figure 1) goes one step further by extending critics to provide proactive design support. Designers outline basic network topologies that are then augmented proactively by ProNET with necessary components such as gateways or repeaters. The rationale for adding components can be requested by the designers.

We have used Agentsheets at the University of Colorado in undergraduate and graduate courses as a programming substrate to create a variety of design environments. Agentsheets has also been used elsewhere as a substrate in applied programming classes. The University of Washington, for instance, has created a number of Agentsheets applications including AgentBuilder, a visual programming language.

An exciting aspect of tools supporting lifelong learning comes out of activities connecting different age groups. LEGOsheets (http://www.cs.colorado.edu/~l3d/systems/legosheets) is a visual programming and simulation environment built in Agentsheets and created by undergraduates for 7- to 13-year-old students (Figure 2). In a one-year effort, and with the assistance of the MIT Media Lab, a team of University of Colorado undergraduates created this new visual programming environment to control programmable LEGOBricks. Designing artifacts that interact with the real world, such as the vehicle in Figure 2, raises interesting issues—especially when contrasted with virtual simulation environments, such as CityTraffic. The frequent interaction between students and creators, supported by Agentsheets, led to many new insights about end-user programming and culminated in an article presented by the undergraduates at an international conference.

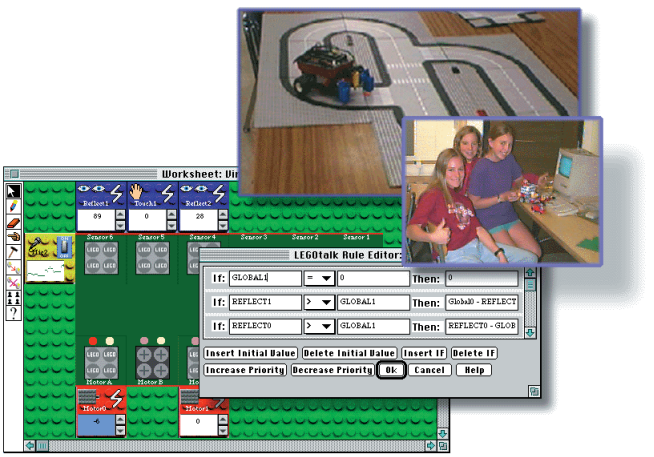

Figure 2. LEGOsheets: A rule-based approach to visually program and simulate a programmable brick controlling LEGO components. The brick is hooked up to a LEGO vehicle including two motors and two optical reflectance sensors. With only two rules the vehicle is able to negotiate a road system.

HyperGami: Computer-Enriched Activities

HyperGami (http://www.cs.colorado.edu/~l3d/systems/hypergami) is a design environment for the domain of "tangible solid geometry" that allows the creation of paper sculptures. The fundamental idea behind the program is that it permits users to create

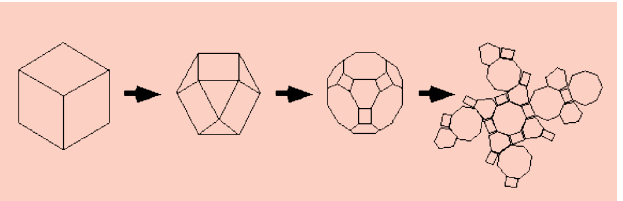

Figure 3.
Generating a new polyhedron and net in HyperGami. The cube (left) is used as a basis for the cuboctahedron (second from left), which in turn is truncated to form a new shape (second from right). The user asks the system to create a folding net for this shape (right). Like any HyperGami net, this new net may now be decorated and printed.

and view 3D objects on the screen. The system "unfolds" these objects to create a 2D folding net pattern that may then be decorated by a variety of means. Finally, the net may be printed out on a color printer and folded into an actual polyhedral model. In structure, HyperGami is a programmable design environment combining both direct manipulation tools (for decorating nets and viewing objects) and an "enriched" version of the Scheme programming language.

One of the key features of HyperGami is that it does not limit users to "standard" polyhedral models, but allows the design and creation of an endless range of customized 3D shapes (See Figure 3 for a simple illustration). As a result, the system transforms polyhedral modeling into something resembling a "mathematical art form." In our initial pilot studies with 8- to 13-year-olds during the past year, students employed the program to create both decorative geometric models and polyhedral sculptures (including sculptures of a hippopotamus and brontosaurus). Indeed, we ourselves have pursued this use of the program to create a mathematical children's book—an alphabet book of paper sculptures (Figure 4).

HyperGami illustrates several themes we have found important within the Center for LifeLong Learning and Design. It focuses on an especially rich mathematical domain (solid geometry), but grounds the educational experience for that domain in the process of design. It integrates—much as in LEGOsheets—both abstract computational work (through the customization and decoration of objects) and craft work that, refreshingly enough, takes place away from the computer screen. It does not view the learning process as frantically hurried, but unabashedly encourages (and rewards) patience and contemplation: one simply cannot build paper models at the pace of a video game. Most importantly, HyperGami seeks to present its users—both children and adults—with a form of dignified and creative mathematical work, and weaves its mathematical content into that work; the question of why one should want to learn mathematics is thus implicitly answered by the very materials used to teach it.

Figure 4.
Sample constructions for the letter "P"—polyhedral penguin sculptures created from simple variants of the dodecahedron, cuboctahedron, and pentagonal prism (Sculptures by A. Nishioka)

What Next?

One of our central problems is to give access to Agentsheets and HyperGami to a larger community of learners. We have started an effort to combine design environments with networking media in the form of our RemoteExplorium (http://www.cs.colorado.edu/~l3d/systems/Remote-explorium). Specifically, in HyperGami, we will pursue the presentation of our children's book nets on the World-Wide Web along with the development of classroom materials; longer-term projects include the expansion of the system into the creation of "dynamic sculptures" combining mathematical, engineering, and artistic elements.

In the case of Agentsheets we have created an initial multiuser Agentsheets prototype, supporting collaborative, synchronous design activity. A separate effort is WebQuest, using the realm of a game that can be authored by children as a Web exploration tool in the possible spirit of "Where on the Web is Carmen San Diego™." Our vision is to create Web end-user languages in support of distributed constructionism allowing users to test, create, and share interesting chunks of end-user programs via the Web.

Acknowledgments

The authors would like to thank the members of the Center for LifeLong Learning and Design at the University of Colorado, in particular Jim Ambach, Andri Ioannidou, Olav Lokkebo, Ann Nishioka, and Corrina Perrone, who have contributed substantially to the conceptual framework and the systems discussed in this article.

Hal Eden *is an associate director for the Center for LifeLong Learning and Design.* ***Mike Eisenberg*** *is an assistant professor in the Department of Computer Science, University of Colorado, Boulder.* ***Gerhard Fischer*** *is Director for the Center for LifeLong Learning and Design.* ***Alex Repenning*** *is a research assistant professor in the Department of Computer Science, University of Colorado, Boulder.*

This research was supported by the National Science Foundation under grants. IRI-9258684, RED-9253245, and REC-9553371.