Software *by* Kids *for* Kids

Yasmin B. Kafai

For many children today, their first interaction with technology is at home playing video games. The number of hours spent in front of these screens must be in the order of hundreds of billions. While many researchers, parents, and educators have good reason to be concerned about the quantity of time spent in this manner, they tend to overlook that even so-called educational software often does not function as any more than enhanced page-turning devices displaying information to be learned and monitoring students' progress. There are currently few opportunities for children to go beyond button-pushing and mouse-clicking in their interaction with technology.

By asking children to program software for other children, we are turning the tables and placing children in the active role of constructing their own programs—and constructing new relationships with knowledge in the process. The most obvious benefit is that children learn about technology by building things of significance, such as game software. A far more promising aspect is that learning programming and learning about technology is not only good for its own sake but also good because it is supportive of other types of learning. As I will explain in more detail, the very process of programming game software to teach fractions (or any other subject topic, for that matter) to younger users allows children to engage in significant mathematical thinking and learning. But most importantly, through programming, children learn to express themselves in the technological domain. In the world of educational programming, these last two aspects of learning with technology have received far less attention than the traditional benefits of technological knowledge.

The Game Design Studio

A software design project starts with a simple instruction: "Design a computer game that teaches something about fractions to younger students." Everything else is left open. A class of students transforms their classroom into a game design studio for six months. During that period, they are:

- Learning programming;
- Thinking about interface designs;
- Designing graphical elements;
- Conceiving story structures, dialogue, and characters;
- Devising instructional strategies; and,
- Creating fraction representations.

Students meet every day for one hour to write in their notebooks about their ideas, plans, and designs. They also discuss issues related to programming, games, teaching, and fractions. They give presentations to each other and meet once a month with their prospective users. All the students create a fully finished product—a computer game—with its documentation, advertising, and packaging. Since 1991, several software design projects with a focus on various mathematical or science topics have been conducted successfully with students ages 8 to 11 in public elementary schools in Boston and Los Angeles.

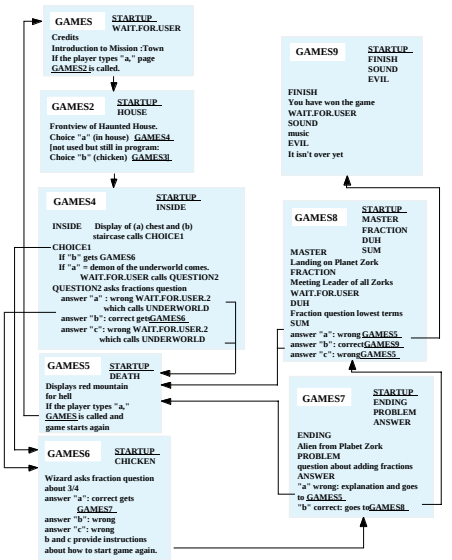


Figure 1.
Page structure and connections in Albert's game *Mission: Town*

The Benefits of Making Software for Learning

There is no doubt that students acquired extensive programming experience in this project. An overview of students' Logo pages reveals that most of them created complex, interconnected pieces of software. One example is Albert's game "Mission: Town" where at the end of the project consisted of more than 20 pages of code distributed over nine pages with multiple procedures (see Figure 1).

But most importantly, all the games provided evidence of students' efforts to integrate the content to be taught—fractions. Students created dozens of situations with fractions in their notebooks, but only the best designs found their way into the games. In this context, children engaged their fantasies and built relationships with other pockets of reality that went beyond traditional school approaches in mathematics education. The introductory screens of two students, Amy and Trevor, provide an example of this effort (see Figure 2).

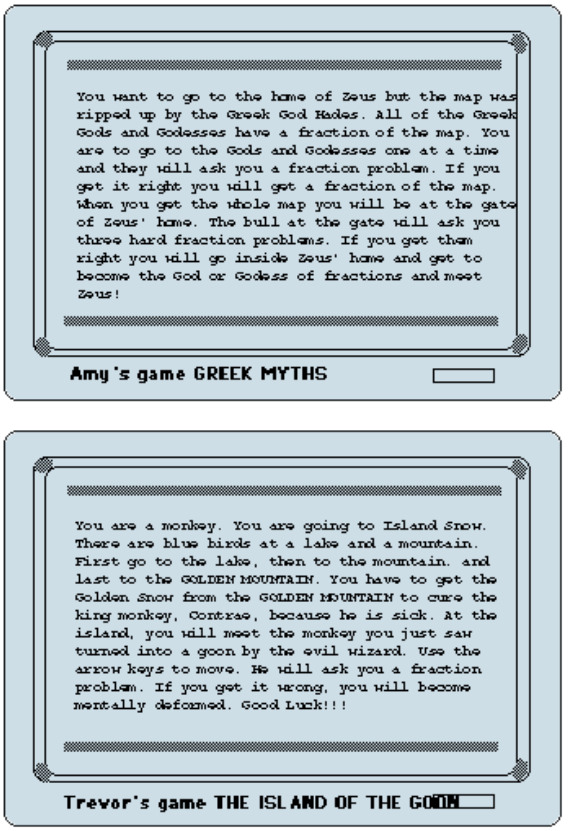
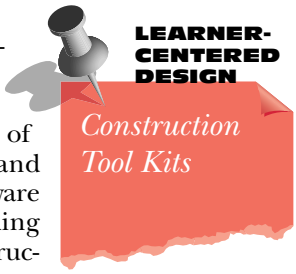


Figure 2.
Trevor and Amy's introductory game screens

A comment on the instructional drill-and-practice format adopted by most students in the design and implementation of their games: a correct answer produces a positive outcome, or "become the god or goddess of fractions," whereas the incorrect answer results in punishment, or "become mentally deformed." One explanation for this indeliberate consistency is that students emulated instructional models found in the commercial market and media. This should be a warning sign of how the format, content, and modalities of educational software influence children's thinking about the standards of instructional software.



Programming games are a medium for their personal and creative expression. This is of particular relevance if one is concerned with finding meaningful and relevant learning situations for students. Being engaged in this enterprise initiates learning and learning about learning. This is best expressed in a final review written by Rosemary, a 10-year-old game designer: "I made a game. It started out very slowly at first. It is very hard to put together your own game. You may think it is easy to do because of all the video games people play. They look so simple, but try making your own game and it's a totally different story. Well, I started out with very high expectations thinking that I could make a great game in a very short time. It turned out that I'm still not done with it even after about four or five months. Truthfully, I hope next time you play a computer or video game you will think about its maker."

More Tools and Toys for Young Software Designers

Designing games offers a rich learning environment for children to become engaged in a variety of issues and to learn about many more aspects of programming than I was able to address in this context. The idea of children making software for fun and learning is definitely not limited to school activities; it has a place at home and in the virtual playground. Constructive play is an important part of children's development. Children enjoy playing as much as making things. Much more attention and effort should be paid to providing virtual building blocks and tools for young children to experience and master the world.

Further research efforts will examine models of interdisciplinary and extended learning for young game software designers and how various information sources can be integrated into this effort. Again, the point is not about providing access and building adequate information structures for children (those are worthy subjects by themselves!) but to consider the other end: what children can make with information, how they can build their own information structures, and, ultimately, how they construct knowledge out of information.

Yasmin B. Kafai is an assistant professor at UCLA Graduate School of Education & Information Studies. http://www.gse.ucla.edu/facpage/kafai.html

The described research was conducted at the MIT Media Laboratory and sponsored by the National Science Foundation and the Nintendo Co. Ltd.