Published in *The Human-Computer Interaction Handbook: Fundamentals, Evolving Technologies, and Emerging Applications*, edited by Julie Jacko and Andrew Sears. Lawrence Erlbaum and Associates, 2002.

HCI for Kids

Amy Bruckman, Georgia Institute of Technology
Alisa Bandlow, Georgia Institute of Technology

Table of Contents

1. Designing for Children

How is designing computer software and hardware for kids different from designing for adults? At the time of this writing, little formal research has been done on this topic.

Most research done to date has focused on designing educational software, and evaluation is primarily of learning outcomes, not usability. However, usability is a prerequisite for learning. In student projects in Georgia Tech's graduate class "Educational Technology: Design and Evaluation," many student designers never are able to show whether the educational design of their software is successful. What they find instead is that usability problems intervene, and they are unable to even begin to explore pedagogical efficacy. If children can't use educational technology effectively, they certainly won't learn through the process of using it. Usability is similarly important for entertainment, communications, and other applications.

In designing for children, people tend to assume that kids are creative, intelligent, and capable of great things if they are given good tools and support. If children can't use technologies we've designed, it is our failure as designers. These assumptions are constructive, because users generally rise to designers' expectations. In fact, the same assumptions are useful in designing for adults. Designers of software for children start out at an advantage, because they tend to believe in their users. However, they may be at a disadvantage, because they no longer remember the physical and cognitive differences of being a child.

In this chapter, we will:

- describe how children's abilities change with age, as it relates to HCI;
- discuss how children differ from adults cognitively and physically, for those characteristics most relevant for HCI;
- review recommendations on laboratory-based testing with kids;
- discuss participatory design with children as design partners; and
- review genres of computer technology for kids, and design

recommendations for each genre.

2. How are Children Different?

As people develop from infants to adults, their physical and cognitive abilities increase over time (Kail, 1991; Miller & Vernon, 1997; Thomas, 1980). The Swiss psychologist Jean Piaget was a leading figuring in analyzing how children's cognition evolves (Piaget, 1970). Piaget showed that children don't just lack knowledge and experience, but also fundamentally experience and understand the world differently than adults. He divided children's development into a series of stages:

Sensori-motor (birth—2 years)

Pre-operational (ages 2-7)

Concrete Operational (ages 7-11)

Formal Operational (ages 11 and up)

(Piaget, 1970, pp. 29-33)

Contemporary research recognizes that all children develop differently, and individuals may differ substantially from this typical picture (Schneider, 1996). However, this general characterization remains useful.

In the sensori-motor stage, children's cognition is heavily dependent on what their senses immediately perceive. Software for children this young is difficult to design. Little interaction can be expected from the child. Obviously, all instruction must be given in audio, video, or animation, since babies can't read. Furthermore, babies generally cannot be expected to use the mouse effectively, even with large targets. "Jumpstart Baby" by Knowledge Adventure (http://www.knowledgeadventure.com) is recommended for ages 9 to 24 months. The child is

presented with a mobile of spinning icons, each representing a different activity. One icon spins at a time. The child can either click on the icon for the activity he/she wishes to play or hit any key while the icon for that activity is spinning. This eliminates the need for the child to click on a specific mouse target. Within each activity, the infant simply hits any key to advance the animation through fixed patterns. For example, in the jigsaw puzzle activity, hitting a key puts a puzzle piece in place. The user can't chose which piece or where it goes. The puzzle is finished one piece at a time by simply having the child hit the keyboard periodically.

Another title aimed at this age group is "Play with the Teletubbies." Also by Knowledge Adventure, it is aimed at ages one to four years. In this program, an animated world runs semi-autonomously. To give the child greater feedback, mouse movement is accentuated by surrounding the cursor with a shower of sparkles. The sparkles intensify when the mouse moves, and when it is over an active part of the scene. The animation proceeds on its own most of the time, but the child's actions add additional sounds and may modify the animation slightly. For example, click on a Teletubby, and it waves hello, and then continues what it was doing. Clicking on a specific part of the scene is occasionally necessary to move the story forward. Repetition is used extensively.

"Jumpstart Baby" is designed in accordance with adult expectations of what a baby should like. The narrator, a teddy bear, addresses the child and invites him/her to play. A motherly voice helps the child play hide and seek to find where teddy is hiding. The design is in strong conformance with adult stereotypes of what babies like. In contrast, Teletubbies is out of harmony with those stereotypes. Many adults find the television show and software bizarre and grating, but it is wildly popular with toddlers. The designers of the original BBC television series, Anne Wood and Andy Davenport, used detailed observations of young children's play and speech in their design. Wood comments, "Our ideas always come from children. If you make something for children, the first question you must ask yourself is, 'What does the world look like to

children?' Their perception of the world is very different to that of grown-ups. We spend a lot of time watching very young children: how they play; how they react to the world around them; what they say" (Davenport & Wood, 1997). Focus groups also played an important role (BBC, 1997). Young children are so radically different from adults that innovative design requires careful fieldwork.

While toddlers' interaction with software on a standard desktop computer affords limited possibilities, specialized hardware can expand the richness and complexity of interactions. For example, "Music Blocks" by Neurosmith is recommend for ages two and up. Five blocks fit in slots in the top of a device rather like a "boom box" portable music player. Each block represents a phrase of music. Each side of the block is a different instrumentation of that musical phrase. Rearranging the blocks changes the music (http://www.neurosmith.com). Interaction of this complexity would be impossible for two-year-olds using a screen-based interface, but is quite easy with specialized hardware.

Figure 1: Children playing with Music Blocks.

In the pre-operational stage (ages 2-7), children's attention span is brief. They can only hold one thing in memory at a time. They have difficulty with abstractions. They can't understand situations from another person's point of view. While some children may begin to read at a young age, designs for this age group generally assume the children are still pre-literate. It is reasonable to expect children at this age can click on specific mouse targets, but they must be relatively large. Use of the keyboard is still generally avoided by most designers.

In the concrete operational stage (ages 7 to 11), "we see children maturing on the brink of adult cognitive abilities. Though they cannot formulate hypothesis, and though abstract concepts such as ranges of numbers are often still difficult, they are able to group like items and categorize" (Schneider, 1996). Concrete operational children are old enough to use relatively sophisticated software, but young enough to still appreciate a playful approach. It is reasonable to expect simple keyboard use. Children's ability to learn to type grows throughout this age group. It's reasonable to expect relatively fine control of the mouse.

Finally, by the time a child reaches the formal operational stage (ages 12 and up), designers can assume the child's thinking is generally similar to that of adults. Their interests and tastes, of course, remain different. Designing for this age group is much less challenging, because adult designers can at least partially rely on their own intuitions.

In the next sections, we'll focus on several characteristics of children most relevant for HCI research:

- Dexterity,
- Speech,
- Reading,
- Background knowledge, and
- Interaction style.

2.1 Dexterity

Young children's fine motor control is not equal to that of adults (Thomas, 1980), and they are physically smaller. Devices designed for adults may be difficult for children to use. Joiner, Messer et al. (1998) note that "the limited amount of research on children has mainly assessed the performance of children at different ages and with different input devices." Numerous studies confirm that children's performance with mice and other input devices increases with age (Joiner, 1998). Compared to adults, children have difficulty holding down the mouse button for extended periods and have difficulty performing a dragging motion (Strommen, 1994). Kids have difficulty with marquee selection. Marquee selection is a technique for selecting several objects at once using a dynamic selection shape. In traditional marquee selection, the first click on the screen is the initial, static corner of the selection shape (typically a rectangle). Dragging the mouse controls the diagonally opposite corner of the shape, allowing you to change the dimensions of the selected area to encapsulate the necessary objects. Dragging the mouse away from the initial static corner increases the size of the selection rectangle, while dragging the mouse towards the initial static corner decreases the size of the selection rectangle. A badly placed initial corner can make it difficult and sometimes impossible to select/encapsulate all of the objects. Berkovitz (1994) experimented with a new encirclement technique: the initial area of selection is specified with an encircling gesture and moving the mouse outside of the area enlarges it.

Kids may have trouble double-clicking, and their small hands may have trouble using a three-button mouse (Bederson et al., 1996). As with adults, point-and-click interfaces are easier to use than drag-and-drop (Inkpen, 2001; Joiner, 1998). Inkpen (2001) notes that "Despite this knowledge, children's software is often implemented to utilize a drag-and-drop interaction style. Bringing solid research and strong results […] to the forefront may help make designers of children's software think more about the implications of their design choices."

Strommen (1998) notes that since young children can't reliably tell their left from their right, interfaces for kids should not rely on that distinction. In his Actimates interactive plush toy designs, the toys' left and right legs, hands, and eyes always perform identical functions.

2.2 Speech

Speech recognition has intriguing potential for a wide-variety of applications for children. O'Hare and McTear (1999) studied use of a dictation program by 12-year-olds and found that they could generate text more quickly and accurately than by typing. They note that dictation automatically avoids some of the errors children would otherwise make, because the recognizer generates correct spelling and capitalization. This is desirable in applications where generating correct text is the goal. If instead the goal is to teach children to write correctly (and, for example, to capitalize their sentences), then dictation software may be counter-productive.

While O'Hare et al. (1999) were able to use a standard dictation program with 12-year-olds, Nix, Fairweather, and Adams (1998) note that speech recognition developed for adults will not work with very young children. In their research on a reading tutor for children 5 to 7 years old, they first tried a speech recognizer designed for adults. The recognition rate was only 75%, resulting in a frustrating experience for their subjects. Creating a new acoustic model from the speech of children in the target age range, they were able to achieve an error rate of less than 5%. Further gains were possible by explicitly accounting for common mispronunciations and children's tendency to respond to questions with multiple words where adults would typically provide a one-word answer. Even with the improved acoustic model, the recognizer still made mistakes. To avoid frustrating the children with incorrect feedback, they chose to have the system never tell the child they were wrong. When the system detects what it believes to be a wrong answer, it simply gives the child an easier problem to attempt.

2.3 Reading

The written word is the main vehicle for most communication between humans and computers. Consequently, designing computer technology for children with developing reading skills presents a challenge. Words must be chosen that are at an appropriate reading level for the target population. Larger font sizes are generally preferred. Bernard, Mills et al. (2001) found that kids 9 to 11 years old prefer 14-point fonts over 12-point. Surprisingly, at the time of this writing, this is the only known empirical study in this area. Most designers follow the rule of thumb that the younger the child, the larger the font should be.

Designing for pre-literate children presents a special challenge. Audio, graphics, and animation must substitute for all functions that would otherwise be communicated in writing. The higher production values required can add significantly to development time and cost.

2.4 Background Knowledge

Many user interfaces are based on metaphors (Erickson, 1990) from the adult world. Jones (1992) notes that children are less likely to be familiar with office concepts like file folders and in-out boxes. In designing an animation system for kids, Halgren, Fernandes and Thomas (1995) found many kids to be unfamiliar with both the metaphor of a frame-based film strip and that of a VCR. It's helpful to choose metaphors that are familiar to kids, though kids often have success in learning interfaces based on unfamiliar metaphors if they are clear and consistent (Schneider, 1996).

2.5 Interaction Style

Children's patterns of attention and interaction are quite different from those of adults. Children are easily distractible. Hanna, Risden and Alexander (1997) used a funny noise as an error message and found that the children repeatedly generated the error to hear the noise.

Similarly, Halgren and colleagues (1995) found that children would click on any readily visible feature just to see what would happen, and they might click on it repeatedly if it generated sound or motion in feedback. They chose to redesign their interface to hide advanced functionality in drawers. Children found the drawer metaphor familiar. "By hiding the advanced tools, the novice users would not stumble onto them and get lost in their functionality. Rather, only the advanced users who might want the advanced tools would go looking for more options. This redesign allows the product to be engaging and usable by a wider range of ages and abilities" (Halgren, Fernandes, & Thomas, 1995).

Children are more likely than adults to work with more than one person at a single computer. They enjoy doing so to play games (Inkpen, 1997) and may be forced to do so because of limited resources in school (Stewart, Raybourn, Bederson, & Druin, 1998). Teachers may also create a shared-computer setup to promote collaborative learning. When multiple children work at one machine simultaneously, they need to negotiate sharing control of input devices. Giving students multiple input devices increases their productivity and their satisfaction (Inkpen, 1997; Inkpen, Gribble, Booth, & Klawe, 1995; Stewart et al., 1998). Inkpen et al. compared two different protocols for transferring control between multiple input devices: give and take. In a give protocol, the user with control clicks the right mouse button to cede it to the other user; in a take protocol, the idle user clicks to take control. In one study with 12-year-olds and another with 9 to 13-year-olds, they found that girls solve more puzzles with a 'give' protocol, but boys are more productive with a 'take' protocol (Inkpen, 1997; Inkpen et al., 1995). (For more on issues of gender and HCI, see the chapter by Justine Cassell in this volume.)

3. Children and Usability Testing

Several usability guidelines developed for work with adults become more important when applied to children. For example, it is important to emphasize that it is the software which is being tested,

not the participant (Rubin, 1994). Children might become anxious at the thought of taking a test, and test taking may conjure up thoughts of school. The researcher can emphasize that even though the child is participating in a test, the child is not the one being tested (Hanna, Risden, & Alexander, 1997). Rubin recommends that you show the participant where the video cameras are located, let them know what is behind the one-way mirror, and whether or not people will be watching. With children, showing them behind the one-way mirrors and around the lab gives them "a better sense of control and trust in you" (Hanna et al., 1997).

Hanna, Risden and Alexander (1997) have developed a set of guidelines for laboratory-based usability testing with children:

- The lab should be made a little more child-friendly by adding some colorful posters, but avoid going overboard as too many extra decorations may become distracting to the child.
- Try to arrange furniture so that children are not directly facing the video camera and one-way mirror, as the children may choose to interact with the camera and mirror rather than the doing the task at hand.
- Children should be scheduled for an hour of lab time. Preschoolers will generally only be able to work for 30 minutes but will need extra time to play and explore. Older children will become tired after an hour of concentrated computer use, so if the test will last longer than 45 minutes, children should be asked if they would like to take a short break at some point during the session.
- Hanna and colleagues suggest that you "Explain confidentiality agreements by telling children that designs are 'top-secret'." Parents should also sign the agreements, since they will inevitably also see and hear about the designs.
- Children up to 7 or 8-years-old will need a tester in the room with them for reassurance and encouragement. They may become agitated from being alone or following directions

from a loudspeaker. If a parent will be present in the room with the child, it is important to explain to the parent that he/she should interact with the child as little as possible during the test. Older siblings should stay in the observation area or a separate room during the test as they may eventually be unable to contain themselves and start to shout out directions.

- Hanna suggests that you should "not ask children if they want to play the game or do a task – that gives them the option to say no. Instead use phrases such as "Now I need you to…" or "Let's do this…" or "It's time to…""

3.1 Disagreement Over Use of Video

There is some disagreement over the use of video cameras in research. Druin (1999) and her design team prefer not to use video cameras during observations of children. They found that children tended to "freeze" or "perform" when they saw a video camera in the room. There are also technical difficulties to deal with. Her research team found that, even with smaller cameras, it was difficult to capture data in small bedrooms and large public spaces. The sound and speech captured in public spaces was difficult to understand or even inaudible. Finally, it was difficult to know where to place cameras because they didn't know where children would sit, stand or move in the environment. However, she does encourage her design team to use video cameras (along with journal writing, team discussion, and adult debriefing) as a way to record their brainstorming sessions and other design activities.

Goldman-Segall (1996) argues that digital video data is an important part of ethnographic interviews and observations. When using video, the researcher doesn't have to worry about remembering or writing down every detail: "she can concentrate fully on the person and on the subtleties of the conversation." The researcher also has access to "a plethora of visual stimuli which can never be 'translated' into words in text," such as body language, gestures, and facial

expressions. It is especially important to be able to review the body language of children as they interact with software. Hanna, Risden and Alexander (1997) state that children's "behavioral signs are much more reliable than children's responses to questions about whether or not they like something, particularly for younger children. Children are eager to please adults, and may tell you they like your program just to make you happy." Video is extremely useful in being able to study these behavioral signs as the researchers may miss some important signs and gestures during the actual observation or interview.

3.2 Children as Design Partners

Participatory Design is an "approach towards computer systems design in which the people destined to *use* the system play a critical role in *designing* it" (Schuler & Namioka, 1993). With children, this idea is even more important: since they are physically and cognitively different from adults, their participation in the design process may offer significant insights. Schuler writes:

> "[Participatory Design] assumes that the workers themselves are in the best position to determine how to improve their work and their work life... It views the users' perceptions of technology as being at least as important to success as fact, and their feelings about technology as at least as important as what they can do with it." (Schuler & Namioka, 1993) p. xi)

Empowering children in this way and including them in the design process can be difficult due to the traditionally unequal power relationships between kids and adults.

3.3 Cooperative Inquiry

Druin (1999) has developed new research methods that include children in various stages of the design process, developing new technologies for children with children. This approach,

called *cooperative inquiry*, is a combination of participatory design, contextual inquiry, and technology immersion. Children and adults work together on a team as research and design partners. She reiterates the idea that "Each team member has experiences and skills that are unique and important, no matter what the age or discipline" (Alborzi et al., 2000).

In this model, the research team frequently observes children interacting with software, prototypes, or other devices to gain insight into how child users will interact with and use these tools. When doing these observations, adult and child researchers both observe, take notes, and interact with the child users. During these observations, there are always at least two note-takers and one interactor, and these roles can be filled by either an adult or child team member. The interactor is the researcher who initiates discussion with the child user and asks questions concerning the activity. If there is no interactor or if the interactor takes notes, the child being observed may feel uncomfortable, like being "on stage" (Druin, 1999). Other researchers have found that the role of interactor can be useful for members of the design team. Scaife and Rogers (1999) have successfully involved children as informants in the development of ECOi, a program that teaches children about ecology. They wanted to get the kids to help them co-design some animations in ECOi. Rather than just having the software designer observe the children as they played with and made comments about the ECOi prototypes, the software designer took on the role of interactor to elicit suggestions directly. Through these on-the-fly, high-tech prototyping sessions, they learned that "it was possible to get the software designer to work more closely with the kids and to take on board some of their more imaginative and kid-appealing ideas" (Scaife & Rogers, 1999).

When working as design partners, children are included from the beginning. The adults do not develop all the initial ideas and then later see how the children react to them. The children participate from the start in brainstorming and developing the initial ideas. The adult team members need to learn to be flexible and learn to break away from carefully following their

session plans, which is too much like school. Children can perform well in this more improvisational design setting, but the extent to which the child can participate as a design partner depends on his/her age. Children younger than 7 years have difficulty in expressing themselves verbally and being self-reflective. These younger children also have difficulty in working with adults to develop new design ideas. Children older than 10 are typically beginning to become preoccupied with pre-conceived ideas of the way "things are supposed to be." In general, it has been found that children ages 7-10 years old are the most effective prototyping partners. They are "verbal and self-reflective enough to discuss what they are thinking," and understand the abstract idea that their low-tech prototypes and designs are going to be turned into technology in the future. They also don't get bogged down with the notion that their designs must be similar to pre-existing designs and products.

Through her work with children as design partners, Druin (1999, 2001) has discovered that there are stumbling blocks on the way to integrate children into the design process and to help adults and children work together as equals. One set of problems deals with the ability of children to express their ideas and thoughts. When the adult and children researchers are doing observations, it is best to allow each group to develop its own style of note-taking. Adults tend to take detailed notes, and children tend to prefer to draw cartoons with short, explanatory notes. It is often difficult to create one style of note-taking that will suit both groups. Since children may have a difficult time communicating their thoughts to adults, low-tech prototyping is an easy and concrete way for them to create and discuss their ideas. Art supplies such as paper, crayons, clay, and string allow adults and children to work on an equal footing. A problem that arises in practice is that since these tools are child-like, adults may believe that only the child needs to do such prototyping. It is important to encourage adults to participate in these low-tech prototyping sessions.

The second set of problems emerge from the traditionally unequal power relationships between adults and children. In what sense can children be treated as peers? When adults and children are discussing ideas, making decisions, or conducting research, traditional "power structures" may emerge. In conducting a usability study, the adult researcher might lead the child user through the experiment rather than allowing the child to explore freely on his/her own. In a team discussion, the children may act as if they are in a school setting by raising their hands to speak. Adults may even inadvertently take control of discussions. Is it sensible to set up design teams where children are given equal responsibilities to those of adult designers? Getting adults and children to work together as a team of equals is often the most difficult part of the design process. It is to be expected that it may take a while for a group to become comfortable and efficient when working together. It can take up to 6 months for an "intergenerational design team to truly develop the ability to build upon each other's ideas" (Druin et al., 2001). To help diffuse such traditional adult-child relationships, adults are encouraged to dress casually, and there always should be more than one adult and more than one child on a team. A single child may feel outnumbered by the adults, and a single adult might create the feeling of a school environment where the adult takes on the role of teacher. Alborzi (2000) starts each design session with 15 minutes of snack time, where adults and children can informally discuss anything. This helps both adults and children to get to know each other better as "people with lives outside of the lab" (Alborzi et al., 2000) and to improve communication within the group.

Although there have been many successes in having children participate as design and research partners in the development of software, there are still many questions to be answered about the effectiveness of this approach. Scaife and Rogers (1999) attempt to address many of the questions and problems faced when working with children. The first question deals with the multitude of ideas and suggestions produced by children. Children say outrageous things. How do you decide which ideas are worthwhile? When do you stop listening? The problem of

selection is difficult since in the end it is the adult who will decide which ideas to use and which ideas to ignore. Scaife and Rogers suggest creating a set of criteria to "determine what to accept and what not to accept with respect to the goals of the system… You need to ask what the trade-offs will be if an idea or set of ideas are implemented in terms of critical 'kid' learning factors: that is, how do fun and motivation interact with better understanding?" (Scaife & Rogers, 1999)

In addition to deciding which of the children's ideas to use, there is also the problem of understanding the meaning behind what the child is trying to say. Adults tend to assume that they can understand what kids are getting at, but kid talk is not adult talk It is important to remember that children have "a different conceptual framework and terminology than adults" (Scaife & Rogers, 1999).

Another problem with involving children, particularly with the design of educational software, is that "children can't discuss learning goals that they have not yet reached themselves" (Scaife & Rogers, 1999). Can children make effective contributions about the content and the way they should be taught, something which adults have always been responsible for? Adults have assumptions about what is an effective way to teach children. Kids tend to focus on the fun aspects of the software rather than the educational agenda. There may exist a mismatch of expectations if kids are using components of the software in unanticipated ways. Involving children in the design and evaluation process may help detect where these mismatches occur in the software.

4. Genres of Technology for Kids

Technology for kids falls into two broad categories: education and entertainment. When game companies try to mix these genres, they may use the term "edutainment." New products for kids increasingly include specialized hardware as well as software.

4.1 Entertainment

Designers of games and other entertainment software rarely write about how they accomplish their job. Talks are presented each year at the Game Developer's Conference (http://www.gdconf.com), and some informal reflections are gathered as conference proceedings. Attending the conference is recommended for people who wish to learn more about current issues in game design. The magazine *Game Developer* is the leading publication with reflective articles on the game design process.

Most game designers are men, and they work by simply designing games that they themselves would like to play. This simple design technique is easy and requires little if any background research with users. With this approach, they are able to appeal quite effectively to the core gaming audience: young men and teenage boys. However, gaming companies are increasingly recognizing that people outside that group represent a large potential market for their products. Designing for teenagers is relatively easy. Designing for very young children, however, presents substantial challenges. The younger your target audience, the more it is necessary to use sound design methodology, consulting with target users at every stage of the design process. (For more on interactive entertainment, see the chapter by Jesse Schell in this volume.)

Brenda Laurel pioneered the use of careful design methods for non-traditional game audiences in her work with the company Purple Moon in the mid-1990s. Laurel aimed to develop games that appeal to pre-teen girls both to tap this market segment and also to give girls an opportunity to become fluent with technology. Many people believe that use of computer game leads to skills that later give kids advantages at school and work. (See Justine Cassell's chapter on gender and HCI, this volume.) Through extensive interviews with girls in their target age range, Purple Moon was able to create successful characters and game designs. However, the process was so time consuming and expensive that the company failed to achieve profitability fast enough to please its investors. The company was closed in 1999, and its characters and games were sold

to Mattel. Purple Moon perhaps did more research than was strictly necessary, particularly because their area was so new. The broader lesson is that the game industry typically does not budget for needs analysis and iterative design early in the design process. "Play testing" and "quality assurance" typically take place relatively late in the design cycle. Designers contemplating incorporating research early in their design process must consider the financial cost. (For more on game design and evaluation, see the chapter by Dennis Wixon in this volume.)

Game designer Carolyn Miller (1998) highlights seven mistakes ("kisses of death") commonly made by people trying to design games for kids:

> "Death kiss #1: Kids love anything sweet"
>
> Miller writes that "sweetness is an adult concept of what kids should enjoy." Only very young children will tolerate it. Humor and good character development are important ingredients. Don't be afraid to use off-color humor, or to make something scary.
>
> "Death kiss #2: Give 'em what's good for 'em"
>
> She advises, "don't preach, don't lecture, and don't talk down—nothing turns kids off faster."
>
> "Death kiss #3: You just gotta amuse 'em"
>
> "Don't assume that just because they are little, they aren't able to consume serious themes."
>
> "Death kiss #4: Always play it safe!"

Adult games often rely on violence to maintain dramatic tension. Since you probably won't want to include this in your game for kids, you'll need to find other ways to maintain dramatic tension. Don't let your game become bland.

"Death kiss #5: All kids are created equal"

Target a specific age group, and take into consideration humor, vocabulary, skill level, and interests. If you try to design for everyone, your game may appeal to no one.

"Death kiss #6: Explain everything"

In an eagerness to be clear, some people over-explain things to kids. Kids are good at figuring things out. Use as few words as possible, and make sure to use spoken and visual communication as much as possible.

"Death kiss #7: Be sure your characters are wholesome!"

Miller warns that if every character is wholesome, the results are predictable and boring. Characters need flaws to have depth. Miller identifies a number of common pitfalls in assembling groups of characters. It's not a good idea to take a "white bread" approach, in which everyone is white and middle class. On the other end of the spectrum, it's also undesirable to take a "lifesaver approach" with one character for each ethnicity. Finally, you also need to avoid an "off-the-shelf" approach, in which each character represents a stereotype: "You've got your beefy kid with bad teeth; he's the bully. You've got the little kids with glasses; he's the smart one." Create original characters that have depth and have flaws that they can struggle to overcome (Miller, 1998).

4.2 Education

To design educational software, we must expand the concept of user-centered design (UCD) to one of learner-centered design (LCD) (Soloway, Guzdial, & Hay, 1994). There are several added steps in the process:

- Needs analysis
  - For learners
  - For teachers
- Select pedagogy
- Select media/technology
- Prototype
  - Core application
  - Supporting curricula
  - Assessment strategies
- Formative evaluation
  - Usability
  - Learning outcomes
- Iterative design
- Summative evaluation
  - Usability
  - Learning outcomes

In our initial needs analysis, for software to be used in a school setting, we need to understand not just learners but also teachers. Teachers have heavy demands on their time and are held accountable for their performance in ways that vary between districts and between election years.

Once we understand our learner and teacher needs, we need to select an appropriate *pedagogy*—an approach to teaching and learning. For example, behaviorism views learning as a process of stimulus and reinforcement (Skinner, 1968). Constructivism sees learning as a process of active construction of knowledge through experience. A social-constructivist perspective emphasizes learning as a social process (Newman, Griffin, & Cole, 1989). (A full review of approaches to pedagogy is beyond the scope of this chapter.)

Next, we're ready to select the media we will be working with, matching their affordances to our learning objectives and pedagogical approach. Once the prototyping process has begun, we need to develop not just software or hardware, but (for applications to be used in schools) also supporting curricular materials and assessment strategies.

"Assessment" should not be confused with "evaluation." The goal of assessment is to judge an individual student's performance. The goal of evaluation is to understand to what extent our learning technology design is successful. An approach to assessing student achievement is an essential component of any school-based learning technology. For both school and free-time use, we need to design feedback mechanisms so that learners can be aware of their own progress. It is also important to note whether learners find the environment motivating. Does it appeal to all learners, or more to specific gender, learning style, or interest groups?

As in any HCI research, educational technology designers use formative evaluation to informally understand what needs improvement in their learning environment, and guide the process of iterative design. Formative evaluation must pay attention first to usability, and second to learning outcomes. If students can't use the learning hardware or software, they certainly won't learn through its use. Once it's clear that usability has met a minimum threshold, designers then need to evaluate whether learning outcomes are being met. After formative evaluation and iterative design are complete, a final summative evaluation serves to document the effectiveness of the design and justify its use by learners and teachers. Summative evaluation must similarly pay attention to both usability and learning outcomes.

A variety of quantitative and qualitative techniques are commonly used for evaluation of learning outcomes (Gay & Airasian, 2000). Most researchers use a complementary set of both quantitative and qualitative approaches. Demonstrating educational value is challenging, and research methods are an ongoing subject of research.

This represents an idealized learner-centered design process. Just as many software design projects don't in reality follow a comprehensive user-centered design process, many educational technology projects do not follow a full learner-centered design process. Learner-centered design is generally substantially more time consuming than user-centered design. While it may in some cases be possible to collect valid usability data in a single session, learning typically takes place over longer periods of time. To get meaningful data, most classroom trials take place over weeks or months. Furthermore, classroom research needs to fit into the school year at the proper time. If you are using Biologica (Hickey, Kindfield, Horwitz, & Christie, 2000) to teach about genetics, you need to wait until it is time to cover genetics that school year. You may have only one or two chances per year to test your educational technology. It frequently takes many years to complete the learner-centered design process. In the research community, one team may study and evolve one piece of educational technology over many years. In a commercial setting, educational products need to get to market rapidly, and this formal design process is rarely used. (For more on the development of educational software, see the chapter by Chris Quintana in this volume.)

4.3 Genres of Educational Technology

In 1980, Taylor divided educational technology into three genres:

1. Computer as tutor
2. Computer as tool
3. Computer as tutee

Supposing that we are learning about acid rain. If the computer is serving as *tutor*, it might present information about acid rain and ask the child questions to verify the material was understood. If the computer is a *tool*, the child might collect data about local acid rain and input

that data into an ecological model to analyze its significance. If the computer is a *tutee*, the child might program his or her own ecological model of acid rain.

With the advent of the Internet, we must add a fourth genre:

4. Computer-supported collaborative learning (CSCL)

In a CSCL study of acid rain, kids from around the country might collect local acid rain data, enter it into a shared database, analyze the aggregate data, and talk online with adult scientists who study acid rain. This is in fact the case in the NGS-TERC Acid Rain Project (Tinker, 1993).

Genres of Children's Software	Description
Entertainment	Games created solely for fun and pleasure.
Educational	Software created to help children learn about a topic using some type of pedagogy – an approach to teaching and learning.
Computer as Tutor	Often referred to as "drill and practice" or "computer-aided instruction" (CAI), this approach is grounded in behaviorism. Children are presented with information and then quizzed on their knowledge.
Computer as Tool	The learner directs the learning process, rather than being directed by the computer. This approach is grounded in constructivism, which sees learning as an active process of constructing knowledge through experience.
Computer as Tutee	Typically, the learner uses construction kits to help reflect upon what he or she learned through the process of creation. This approach is grounded in constructivism and constructionism.
Computer-supported Collaborative learning (CSCL)	Children use the Internet to learn from and communicate with knowledgeable members of the adult community. Children can also become involved in educational online communities with children from different geographical regions. This approach is grounded in social constructivism.
Edutainment	A mix of the entertainment and educational genres.

4.3.1 Computer as Tutor

In most off-the-shelf educational products, the computer acts as tutor. Children are presented with information and then quizzed on their knowledge. This approach to education is

grounded in behaviorism (Skinner, 1968). It is often referred to as "drill and practice" or "computer-aided instruction" (CAI). The computer tracks student progress and repeats exercises as necessary.

Researchers with a background in artificial intelligence have extended the drill and practice approach to create "intelligent tutoring systems." Such systems try to model what the user knows and tailor the problems presented to an individual's needs. Many systems explicitly look for typical mistakes and provide specially-prepared corrective feedback. For example, suppose a child adds 17 and 18 and gets an answer of 25 instead of 35. The system might infer that the child needs help learning to carry from the ones to the tens column and present a lesson on that topic. One challenge in the design of intelligent tutors is in accurately modeling what the student knows and what their errors might mean.

Byrne (1999) has experimented with using eye tracking to improve the performance of intelligent tutors. Using an eye tracker, the system can tell whether the student has paid attention to all elements necessary to solve the problem. In early trials with the eye tracker, he found that some of the helpful hints the system was providing to the user were never actually read by most students. This helped guide their design process. They were previously focusing on how to improve the quality of hints provided; however, that is irrelevant if the hints are not even being read (Byrne, Anderson, Douglass, & Matessa, 1999).

An interesting variation on the traditional 'computer as tutor' paradigm for very young children is the Actimates line interactive plush toys. Actimates Barney and other characters lead children in simple games with educational value, like counting exercises. The 'tutor' is animated and anthropomorphized. The embodied form lets young children use the skills they have in interacting with people to learn to interact with the system, enhancing both motivation and ease of use (Strommen, 1998; Strommen & Alexander, 1999). (For more on computer-based tutoring systems, see the chapter by Henry Emurian in this volume.)

4.3.2 Computer as Tool

When the computer is used as a tool, agency shifts from the computer to the learner. The learner is directing the process, rather than being directed. This approach is preferred by constructivist pedagogy, which sees learning as an active process of constructing knowledge through experience. The popular drawing program Kid Pix is an excellent example of a tool customized for kids' interests and needs. Winograd comments that Kid Pix's designer Craig Hickman "made a fundamental shift when he recognized that the essential functionality of the program lay not in the drawings that it produced, but in the experience for the children as they used it" (Winograd, 1996). For example, Kid Pix provides several different ways to erase the screen—including having your drawing explode, or be sucked down a drain.

Simulation programs let learners try out different possibilities that would be difficult or impossible in real life. For example, Biologica (an early version was called "Genscope") allows students to learn about genetics by experimenting with breeding cartoon dragons with different inherited characteristics like whether they breathe fire or have horns (Hickey et al., 2000). Model-it lets students try out different hypotheses about water pollution and other environmental factors in a simulated ecosystem (Soloway et al., 1996).

The goal of such programs is to engage students in scientific thinking. The challenge in their design is how to get students to think systematically, and not simply try out options at random. Programs like Model-It provide the student with "scaffolding." Initially, students are given lots of support and guidance. As their knowledge evolves, the scaffolding is "faded," allowing the learner to work more independently (Guzdial, 1994; Soloway et al., 1994).

4.3.3 Computer as Tutee

Seymour Papert comments that much computer-aided instruction is "using the computer to program the child" (Papert, 1992), p. 163). Instead, he argues that the child should learn to program the computer and through this process gain access to new ways of thinking and understanding the world. Early research argued that programming would improve children's general cognitive skills, but empirical trials produced mixed results (Clements, 1986; Clements & Gullo, 1984; Pea, 1984). Some researchers argue that the methods of these studies are fundamentally flawed, because the complexity of human experience can not be reduced to pre and post tests (Papert, 1987). The counter-argument is that researchers arguing that technology has a transformative power need to back up their claims with evidence of some form, whether quantitative or qualitative (Pea, 1987; Walker, 1987). More recently, the debate has shifted to the topic of technological fluency. As technology increasingly surrounds our everyday lives, the ability to use it effectively as a tool becomes important for children's success in school and later in the workplace (Resnick & Rusk, 1996).

In the late 1960s, Feurzeig and colleagues (1996) at BBN invented Logo, the first programming language for kids. Papert extended Logo to include "turtle graphics," in which kids learn geometric concepts by moving a 'turtle' around the screen (Papert, 1980). A variety of programming languages for kids have been developed over subsequent years, including Starlogo (Resnick, 1994), Boxer (diSessa & Abelson, 1986), Stagecast (Cypher & Smith, 1995), Agentsheets (Repenning & Fahlen, 1993), MOOSE (Bruckman, 1997), and Squeak (Guzdial & Rose, 2001). Lego Mindstorms (originally "Lego/Logo") is a programmable construction kit with physical as well as software components (Martin & Resnick, 1993). Another programmable tool bridging the gap between physical constructions and representations on the screen is Hypergami, a computer-aided design tool for origami developed by Michael Eisenberg and Ann Nishioka Eisenberg at the University of Colorado at Boulder. Students working with Hypergami learn about both geometry and art (Eisenberg, Nishioka, & Schreiner, 1997).

Figure 2: Penguins created using Hypergami.

In most design tools, the goal is to facilitate the creation of a product. In educational construction kits, the goal instead is what is learned through the process of creation. So what makes a good construction kit? In a 1996 *Interactions* article entitled "Pianos, Not Stereos: Creating Computational Construction Kits," Resnick, Bruckman, and Martin discuss the art of designing construction kits for learning ("constructional design"):

> "The concept of learning-by-doing has been around for a long time. But the literature on the subject tends to describe specific activities and gives little attention to the general principles governing what kinds of "doing" are most conducive to learning. From our experiences, we have developed two general principles to guide the design of new construction kits and activities. These constructional-design principles involve two different types of "connections":
>
> • Personal connections. Construction kits and activities should connect to users' interests, passions, and experiences. The point is not simply to make the activities more "motivating" (though that, of course, is important). When activities involve objects and actions that are familiar, users can leverage their previous knowledge, connecting new ideas to their pre-existing intuitions.
>
> • Epistemological connections. Construction kits and activities should connect to important domains of knowledge—more significantly, encourage new ways of thinking (and even new ways of thinking about thinking). A well-designed construction kit makes certain ideas and ways of thinking particularly salient, so that users are likely to connect with those ideas in a very natural way, in the process of designing and creating." (Resnick, Bruckman, & Martin, 1996)

Bruckman adds a third design principle:

> • "Situated support. Support for learning should be from a source (either human or computational) with whom the learner has a positive personal relationship, ubiquitously available, richly connected to other sources of support, and richly connected to every-day activities." (Bruckman, 2000)

4.3.4 Computer-Supported Collaborative Learning (CSCL)

Most tools for learning have traditionally been designed for one child working at the computer alone. However, learning is generally recognized to be a social process (Newman et al., 1989). With the advent of the Internet come new opportunities for children to learn from one another and from knowledgeable members of the adult community. This field is called "Computer-Supported Collaborative Learning (CSCL) (Koschmann, 1996).

CSCL research can be divided into four categories:

- Distance education

Attempts to move something like a traditional classroom online.

- Information retrieval

Research projects in which students use the Internet to find information.

- Information sharing

Students debate issues with one another. One of the first such tools was the Computer-Supported Intentional Learning Environment (CSILE), a networked discussion tool designed to help students engage in thoughtful debate as a community of scientists does (Scardamalia & Bereiter, 1994). They may also collect scientific data and share it with others online. In the "One Sky, Many Voices" project, students learn about extreme weather phenomena by sharing meteorological data they collect with other kids from around the world, and also by talking online with adult meteorologists (Songer,). In the Palaver Tree Online project, kids learn about history by talking online with older adults who lived through that period of history (Ellis & Bruckman,

2001). A key challenge in the design of information sharing environments is how to promote serious reflection on the part of students (Guzdial, 1994; Kolodner & Guzdial, 1996).

- Technological samba schools

In *Mindstorms*, Seymour Papert has a vision of a "technological samba school." At samba schools in Brazil, a community of people of all ages gather together to prepare a presentation for carnival. "Members of the school range in age from children to grandparents and in ability from novice to professional. But they dance together and as they dance everyone is learning and teaching as well as dancing. Even the stars are there to learn their difficult parts" (Papert, 1980). People go to samba schools not just to work on their presentations, but also to socialize and be with one another. Learning is spontaneous, self-motivated, and richly connected to popular culture. Papert imagines a kind of technological samba school where people of all ages gather together to work on creative projects using computers. The Computer Clubhouse is an example of such a school in a face to face setting (Resnick & Rusk, 1996). MOOSE Crossing is an Internet-based example (Bruckman, 1998). A key challenge in the design of such environments is how to grapple with the problem of uneven achievement among participants. When kids are allowed to work or not work in a self-motivated fashion, typically some excel while others do little. (Elliott, Bruckman, Edwards, & Jensen, 2000) (For more on the design of online communities for kids, see the chapter by Jennifer Preece in this volume.)

4.4 Child Safety Online

One challenge in the design of Internet-based environments for kids is the question of safety. The Internet does contain information that is sexually explicit, violent, and racist. Typically, such information does not appear unless one is looking for it; however, it is unusual but possible to stumble across it accidentally. Filtering software blocks access to useful information as well as harmful (Schneider, 1997). Furthermore, companies that make filtering software often

fail to adequately describe how they determine what to block, and they may have unacknowledged political agendas that not all parents will agree with. Resolving this issue requires a delicate balance of the rights of parents, teachers, school districts, and children (Electronic Privacy Information Center, 2001). Another danger for kids online is the presence of sexual predators and others who wish to harm children. While such incidents are rare, it is important to teach kids not to give out personal information online such as their last name, address, or phone number. Kids who wish to meet an online friend to friend face to face should do so by each bringing a parent and meeting in a well-populated public place like a fast-food restaurant. A useful practical guide "Child Safety on the Information Superhighway" is available from the Center for Missing and Exploited Children (http://www.missingkids.org). Educating kids, parents, and teachers about online safety issues is an important part of the design of any online software for kids.

5.0 Conclusion

To design for kids, we must have a model of what kids are and what we would like them to become. Adults were once kids. Many are parents. Some are teachers. We tend to think that we know kids--who they are, what they are interested in, what they like. However, we do not have as much access to our former selves as many would like to believe. Furthermore, it's worth noting that our fundamental notions of childhood are in fact culturally constructed and change over time. Karin Calvert writes about the changing notion of childhood in America, and the impact it has had on artifacts designed for children and child-rearing:

> "In the two centuries following European settlement, the common perception in America of children changed profoundly, having first held to an exaggerated fear of their inborn deficiencies, then expecting considerable self-sufficiency, and then, after 1830, endowing young people with an almost celestial goodness. In each era, children's artifacts mediated

> between social expectations concerning the nature of childhood and the realities of child-rearing: before 1730, they pushed children rapidly beyond the perceived perils of infancy, and by the nineteenth century they protected and prolonged the perceived joys and innocence of childhood." (Calvert, 1992), p.8)

While Calvert was reflecting on the design of swaddling clothes and walking stools, the same role is played by new technologies for kids like programmable Legos and drill and practice arithmetic programs: these artifacts mediate between our social expectations of children and the reality of their lives. If you believe that children are unruly and benefit from strong discipline, then you are likely to design computer-aided instruction. If you believe that children are creative and shouldn't be stifled by adult discipline, then you might design an open-ended construction kit like Logo or Squeak. In designing for kids, it is crucial to become aware of one's own assumptions about the nature of childhood. Designers should be able to articulate their assumptions, and be ready to revise them based on empirical evidence.

References

Alborzi, H., Druin, A., Montemayor, J., Platner, M., Porteous, J., Sherman, L., Boltman, A., Taxén, G., Best, J., Hammer, J., Kruskal, A., Lal, A., Schwenn, T. P., Sumida, L., Wagner, R., & Hendler, J. (2000). *Designing StoryRooms: Interactive Storytelling Spaces for Children.* Paper presented at the Proceedings of the Symposium on Designing interactive systems: processes, practices, methods, and techniques, Brooklyn, NY.

BBC. (1997). *Teletubbies Press Release*, [Website]. BBC Education. Available: http://www.bbc.co.uk/education/teletubbies/information/pressrelease/.

Bederson, B., Hollan, J., Druin, A., Stewart, J., Rogers, D., & Proft, D. (1996). *Local Tools: An Alternative to Tool Palettes.* Paper presented at the Proceedings of the ACM Symposium on User Interface Software and Technology, Seattle, WA.

Berkovitz, J. (1994). *Graphical Interfaces for Young Children in a Software-based Mathematics Curriculum.* Paper presented at the Proceedings of the ACM Conference on Human Factors in Computings Systems: Celebrating Interdependence, Boston, MA.

Bernard, M., Mills, M., Frank, T., & McKnown, J. (2001). *Which Fonts Do Children Prefer to Read Online?* (Winter 2001), [Web Newsletter]. Software Usability Research Laboratory (SURL). Available: http://wsupsy.psy.twsu.edu/surl/usability_news.html.

Bruckman, A. (1997). *MOOSE Crossing: Construction, Community, and Learning in a Networked Virtual World for Kids.* Unpublished PhD, MIT.

Bruckman, A. (1998). Community Support for Constructionist Learning. *Computer Supported Cooperative Work, 7*, 47-86.

Bruckman, A. (2000). Situated Support for Learning: Storm's Weekend with Rachael. *Journal of the Learning Sciences, 9*(3), 329-372.

Byrne, M. D., Anderson, J. R., Douglass, S., & Matessa, M. (1999). *Eye tracking the visual search of click-down menus.* Paper presented at the Proceedings of the ACM Conference on Human Factors in Computings Systems: the CHI is the limit, Pittsburgh, PA.

Calvert, K. (1992). *Children in the House: The Material Culture of Early Childhood, 1600-1900*. Boston: Northeastern University Press.

Clements, D. H. (1986). Effects of Logo and CAI Environments on Cognition and Creativity. *Journal of Educational Psychology, 78*(4), 309-318.

Clements, D. H., & Gullo, D. F. (1984). Effects of Computer Programming on Young Children's Cognition. *Journal of Educational Psychology, 76*(6), 1051-1058.

Cypher, A., & Smith, D. C. (1995). *End User Programming of Simulations.* Paper presented at the Proceedings of the ACM Conference on Human Factors in Computing Systems, Denver, CO.

Davenport, A., & Wood, A. (1997). *TeleTubbies FAQ*, [Website]. BBC Education. Available: http://www.bbc.co.uk/education/teletubbies/information/faq/q18.shtml.

diSessa, A. A., & Abelson, H. (1986). Boxer: A Reconstructible Computational Medium. *Communications of the ACM, 29*(9), 859-868.

Druin, A. (1999). *Cooperative Inquiry: Developing New Technologies for Children with Children.* Paper presented at the Proceedings of the ACM Conference on Human Factors in Computings Systems: the CHI is the limit, Pittsburgh, PA.

Druin, A., Bederson, B., Hourcade, J. P., Sherman, L., Revelle, G., Platner, M., & Weng, S. (2001). *Designing A Digital Library for Young Children: An Intergenerational Partnership.* Paper presented at the Proceedings of the Joint Conference on Digital Libraries, Roanoke, VA.

Eisenberg, M., Nishioka, A., & Schreiner, M. E. (1997, Jan. 6-9). *Helping Users Think in Three Dimensions: Steps Toward Incorporating Spatial Cognition in User Modelling.* Paper presented at the Proceedings of the International Conference on Intelligent User Interfaces, Orlando, FL.

Electronic Privacy Information Center. (2001). *Filters and Freedom 2.0: Free Speech Perspectives on Internet Content Control*. Washington DC: Electronic Privacy Information Center.

Elliott, J., Bruckman, A., Edwards, E., & Jensen, C. (2000, June). *Uneven Achievement in a Constructionist Learning Environment.* Paper presented at the Proceedings of the International Conference on the Learning Sciences, Ann Arbor, MI.

Ellis, J. B., & Bruckman, A. S. (2001). *Designing palaver tree online: supporting social roles in a community of oral history.* Paper presented at the Proceedings of the SIG-CHI on Human factors in computing systems, Seattle, WA.

Erickson, T. (1990). Working With Interface Metaphors. In B. Laurel (Ed.), *The Art of Human-Computer Interface Design* (pp. 65-73). Reading, MA: Addison Wesley Publishing Company Inc.

Feurzeig, W. (1996). Personal communication .

Gay, L. R., & Airasian, P. (2000). *Education Research: Competencies for Analysis and Application*. (6 ed.). Upper Saddle River, NJ: Merrill.

Goldman-Segall, R. (1996). Looking Through Layers: Reflecting upon Digital Video Ethnography. *JCT: An Interdisciplinary Journal For Curriculum Studies, 13*(1).

Guzdial, M. (1994). Software-Realized Scaffolding to Facilitate Programming for Science Learning. *Interactive Learning Environments, 4*(1), 1-44.

Guzdial, M., & Rose, K. (Eds.). (2001). *Squeak: Open Personal Computing and Multimedia*: Prentice Hall.

Halgren, S., Fernandes, T., & Thomas, D. (1995). *Amazing AnimationTM: Movie Making for Kids Design Briefing.* Paper presented at the Proceedings of the ACM Conference on Human Factors in Computings Systems, Denver, CO.

Hanna, L., Risden, K., & Alexander, K. (1997). Guidelines for Usability Testing with Children. *Interactions, 4*(5), 9-14.

Hickey, D. T., Kindfield, A. C. H., Horwitz, P., & Christie, M. A. (2000). *Integrating Instruction, Assessment, and Evaluation in a Technology-Based Genetics Environment: The GenScope Follow-up Study.* Paper presented at the Proceedings of the International Conference of the Learning Sciences, Ann Arbor, MI.

Inkpen, K. (1997). *Three Important Research Agendas for Educational Multimedia: Learning, Children and Gender.* Paper presented at the Proceedings of Graphics Interface, Calgary, AB.

Inkpen, K. (2001). Drag-and-Drop versus Point-and-Click: Mouse Interaction Styles for Children. *ACM Transactions Computer-Human Interaction, 8*(1), 1-33.

Inkpen, K., Gribble, S., Booth, K. S., & Klawe, M. (1995). *Give and Take: Children Collaborating on One Computer.* Paper presented at the Proceedings of the ACM Conference on Human Factors in Computings Systems, Denver, CO.

Joiner, R., Messer, D., Light, P. and Littleton, K. (1998). It Is Best to Point for Young Children: A Comparison of Children's Pointing and Dragging. *Computers in Human Behavior, 14*(3), 513-529.

Jones, T. (1992). Recognition of animated icons by elementary-aged children. *Association for Learning Technology Journal, 1*(1), 40-46.

Kail, R. (1991). Developmental Changes in speed of processing during childhood and adolescence. *Psychological Bulletin, 109*, 490-501.

Kolodner, J., & Guzdial, M. (1996). Effects With and Of CSCL: Tracking Learning in a New Paradigm. In T. Koschmann (Ed.), *CSCL: Theory and Practice* . Mahwah, NJ: Lawrence Erlbaum Associates.

Koschmann, T. (Ed.). (1996). *CSCL: Theory and Practice*. Mahwah, NJ: Lawrence Erlbaum Associates.

Martin, F., & Resnick, M. (1993). LEGO/Logo and Electronic Bricks: Creating a Scienceland for Children. In D. L. Ferguson (Ed.), *Advanced Educational Technologies for Mathematics and Science* (pp. 61-90). Berlin Heidelberg: Springer-Verlag.

Miller, C. (1998). *Designing for Kids: Infusions of Life, Kisses of Death.* Paper presented at the Proceedings of the Game Developers Conference, Longbeach, CA.

Miller, L. T., & Vernon, P. A. (1997). Developmental Changes in speed of information processing in young children. *Developmental Psychology, 33*(4), 549-554.

Newman, D., Griffin, P., & Cole, M. (1989). *The Construction Zone: Working for Cognitive Change in School*. Cambridge, England: Cambridge University Press.

Nix, D., Fairweather, P., & Adams, B. (1998). *Speech recognition, children, and reading.* Paper presented at the Proceedings of the ACM Conference on Human Factors in Computings Systems, Los Angeles, CA.

O'Hare, E. A., & McTear, M. F. (1999). Speech recognition in the secondary school classroom: an exploratory study. *Computers & Education, 3*(8), 27-45.

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

Papert, S. (1987,). Computer Criticism vs. Technocentric Thinking. *Educational Researcher, Jan-Feb,* 22-30.

Papert, S. (1992). *The Children's Machine*: BasicBooks.

Pea, R. (1984). On the Cognitive Effects of Learning Computer Programming. *New Ideas in Psychology, 2*(2), 137-168.

Pea, R. (1987). The Aims of Software Criticism: Reply to Professor Papert. *Educational Researcher, June-July*, 4-8.

Piaget, J. (1970). *Science of Education and the Psychology of the Child*. New York: Orion Press.

Repenning, A., & Fahlen, L. E. (1993). *Agentsheets: A Tool for Building Domain-Oriented Visual Programming Environments.* Paper presented at the Proceedings of the ACM Conference on Human Factors in Computing Systems, Amsterdam, The Netherlands.

Resnick, M. (1994). *Turtles, Termites, and Traffic Jams: Explorations in Massively Parallel Microworlds*. Cambridge, MA: MIT Press.

Resnick, M., Bruckman, A., & Martin, F. (1996). Pianos not stereos: Creating Computational Construction Kits. *Interactions, 3*(5), 40-50.

Resnick, M., & Rusk, N. (1996). The Computer Clubhouse: Preparing for Life in a Digital World. *IBM Systems Journal, 35*(3-4), 431-440.

Rubin, J. (1994). *Handbook of Usability Testing*. New York: John Wiley and Sons, Inc.

Scaife, M., & Rogers, Y. (1999). Kids as Informants: Telling Us What We Didn't Know or Confirming What We Knew Already? In A. Druin (Ed.), *The Design of Children's Technology* (pp. 27-50). San Francisco, CA: Morgan Kaufmann Publishers, Inc.

Scardamalia, M., & Bereiter, C. (1994). Computer Support for Knowledge-Building Communities. *The Journal of the Learning Sciences, 3*(3), 265-283.

Schneider, K. G. (1996). Children and Information Visualization Technologies. *Interactions, 3*(5), 68-73.

Schneider, K. G. (1997). *The Internet Filter Assessment Project (TIFAP)*, [Website]. Available: http://www.bluehighways.com/tifap/learn.htm.

Schuler, D., & Namioka, A. (Eds.). (1993). *Participatory Design: Principles and Practices*. Hillsdale, NJ: Lawrence Erlbaum Associates, Publishers.

Skinner, B. F. (1968). *The Technology of Teaching*. New York: Appleton-Century-Crofts.

Soloway, E., Guzdial, M., & Hay, K. E. (1994). Learner-Centered Design: The Challenge for HCI in the 21st Century. *Interactions, 1*(1), 36-48.

Soloway, E., Jackson, S. L., Klein, J., Quintana, C., Reed, J., Spitulnik, J., Stratford, S. J., Studer, S., Jul, S., Eng, J., & Scala, N. (1996). *Learning Theory in Practice: Case Studies of Learner-Centered Design.* Paper presented at the Proceedings of the ACM Conference on Human Factors in Computings Systems, Vancouver, Canada.

Songer, N. Kids as Global Scientists: http://onesky.engin.umich.edu/.

Stewart, J., Raybourn, E. M., Bederson, B., & Druin, A. (1998). *When two hands are better than one: enhancing collaboration using single display groupware.* Paper presented at the Proceedings of the ACM Conference on Human Factors in Computings Systems, Los Angeles, CA.

Strommen, E. (1994). *Children's use of mouse-based interfaces to control virtual travel.* Paper presented at the Proceedings of the ACM Conference on Human Factors in Computings Systems: Celebrating Interdependence, Boston, MA.

Strommen, E. (1998). *When the Interface is a Talking Dinosaur: Learning Across Media with ActiMates Barney.* Paper presented at the Proceedings of the ACM Conference on Human Factors in Computings Systems, Los Angeles, CA.

Strommen, E., & Alexander, K. (1999). *Emotional Interfaces for Interactive Aardvarks: Designing Affect into Social Interfaces for Children.* Paper presented at the Proceedings of the ACM Conference on Human Factors in Computings Systems: the CHI is the limit, Pittsburgh, PA.

Taylor, R. P. (Ed.). (1980). *The Computer In the School, Tutor, Tool, Tutee*: Teachers College Press.

Thomas, J. R. (1980). Acquisition of motor skills: Information processing differences between children and adults. *Research Quarterly for Exercise and Sport, 51*(1), 158-173.

Tinker, R. (1993). *Thinking About Science* . Concord, MA: The Concord Consortium.

Walker, D. F. (1987). Logo Needs Research: A Response to Professor Papert's Paper. *Educational Researcher*, 9-11.

Winograd, T. (1996). Profile: Kid Pix. In T. Winograd (Ed.), *Bringing Design to Software* (pp. 58-61). New York: ACM Press.