CHAPTER 3

Constructionism

Yasmin B. Kafai

Learning sciences researchers are unified by their deep commitment to radically transform learning – away from the transmission and acquisition style associated with lectures and quizzes, to a more active, participatory learning style. Perhaps the first scholar to realize that computers provided schools with an opportunity to do so was Seymour Papert, who created the now-famous Logo programming language. Papert, who received two doctorates in mathematics, expanded his career by studying cognitive development with Jean Piaget, the founder of constructivism – one of the theoretical foundations of today's learning sciences. After leaving Piaget's lab in Switzerland, Papert took a faculty position at the Massachusetts Institute of Technology, where he cofounded the Artificial Intelligence Laboratory with Marvin Minsky. In the 1970s, Papert began to expand the psychological insights of Piaget's constructivism into pedagogical principles, providing a template that later influenced many learning sciences researchers.

When Papert's book *Mindstorms* was published in 1980 the term constructionism had yet to be coined. In this book and subsequent publications he advanced a theory of learning, teaching, and design. Many took the notion of "children, computers and powerful ideas" (to quote the subtitle of his book) as a rather simplistic version of Piagetian discovery learning with the Logo programming language when, in fact, the opposite applied. Constructionism is not constructivism, as Piaget never intended his theory of knowledge development to be a theory of learning and teaching; nor is constructionist learning simply discovery learning and thus opposed to any forms of instruction; and last, in constructionism, people and not computers are seen as the driving force for educational change. Papert's constructionism views learning as building relationships between old and new knowledge, in interactions with others, while creating artifacts of social relevance. Thus any chapter on constructionism needs to begin with a clarification of three issues – constructivism, instructionism, and technocentrism – before delving any further into the theoretical and pedagogical particulars.

Constructionism's close resemblance to Piaget's constructivism often leads to confusion, yet there is a clear distinction between the two:

> *[C]onstructionism – the N Word as opposed to the V word – shares constructivism's connotation to learning as building knowledge structures irrespective of the circumstances of learning. It then adds the idea that this happens especially felicitously in a context where the learner is consciously engaged in constructing a public entity whether it's a sand castle on the beach or a theory of the universe. (Papert, 1991, p.1)*

Constructionism always has acknowledged its allegiance to Piagetian theory but it is not identical to it. Where constructivism places a primacy on the development of individual and isolated knowledge structures, constructionism focuses on the connected nature of knowledge with its personal and social dimensions. This combination of individual and social aspects in learning is at the heart of many discussions in the learning sciences.

The opposition of constructionism to instructionism often aligns constructionist learning with discovery learning – as learning without curriculum in which the child discovers principles or ideas by him or herself. A common myth associated with constructionism is the idea that all instruction is bad. A closer reading of Papert's original writings clarifies this issue:

> *. . . but teaching without curriculum does not mean spontaneous, free-form classrooms or simply 'leaving the child alone'. It means supporting children as they build their own intellectual structures with materials drawn from the surrounding culture. In this model, educational intervention means changing the culture, planning new constructive elements in it and eliminating noxious ones. (p. 31, 1980/1993)*

Constructionism has articulated a more distributed view of instruction, one where learning and teaching are constructed in interactions between the teacher and students as they are engaging in design and discussion of learning artifacts. Furthermore, such learning interactions are not limited to schools alone but extend into community centers and families. How to design learning environments that facilitate collaboration and idea sharing is a key focus of many efforts in the learning sciences.

The Logo programming language has always been closely associated with constructionism, and this has led many to believe that constructionism sees technology as the driving force for how we teach and learn. Yet, as Papert argued, this type of technocentric thinking assigns more importance than is appropriate to technology as an agent of change:

> *Does wood produce good houses? If I built a house out of wood and it fell down, would this show that wood does not produce good houses? Do hammers and saws produce good furniture? These questions betray themselves as technocentric questions by ignoring people and elements that only people can introduce: skill, design, and aesthetics. (p. 24, 1987)*

Constructionism challenges us to reconsider our notions of learning and teaching. Programming with Logo provided a testing bed for engaging students in problem solving and learning to learn. Moreover, programming in Logo also illustrated conceptually different ways of learning mathematics and science with computers. Many of these challenges to learning and teaching continue to be relevant in the learning sciences, whether or not computers are involved.

The goal of this chapter is, then, to articulate more clearly a constructionist perspective on the nature of knowing, teaching, and learning. In the first part, I review the historical roots of constructionism using Logo as an example and move on to discuss key constructionist ideas around knowledge construction, learning cultures, and the application of knowledge construction to the design of microworlds and construction kits. I then present a case study of software learning through design activities that illustrate the implementation of core constructionist ideas. In conclusion, I address outstanding issues and challenges in constructionism and in the field of the learning sciences.

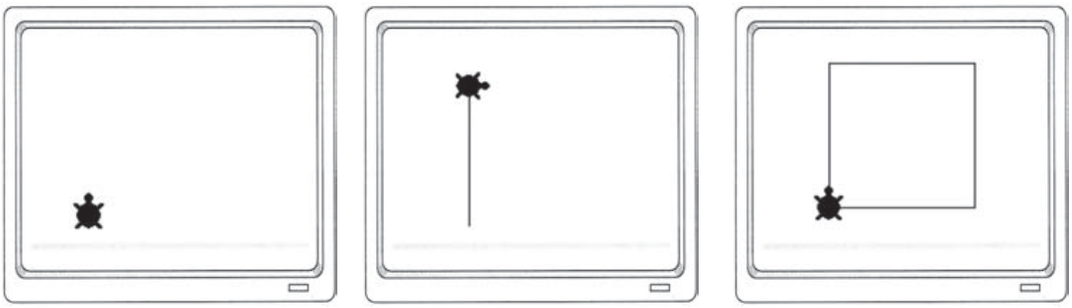

Figure 3.1. The screen shot to the left shows a turtle in starting position ready to be programmed. In the middle screen, the turtle has been programmed to execute the following steps: PEN DOWN FORWARD 10 RIGHT TURN 90. The right screen shows the drawing of a square by having the turtle repeat the commands four times or REPEAT 4 [FORWARD 10 RIGHT TURN 90].

The Historical Roots

In any historical account of constructionism the programming language Logo is the "evocative object" – to use a phrase coined by Sherry Turkle (Turkle, 1995) – for it embodies the issues that fueled many debates about computers in schools in the early 1980s. At that time, the computer was ready to move out of the university laboratories into the world, but computer work was seen as the exclusive domain of adults. Logo was not the first programming language used by children. Basic was prominent in many schools and indeed there was considerable debate about which programming language would be best for schools. But in contrast to Basic, learning with Logo promised to offer more than just learning to program: it included learning about your own thinking and learning, and learning mathematics and science in conceptually new ways. These additions made Logo unlike any other programming language.

The first feature to note about Logo is how learners interacted with the computer: children were writing commands to move a graphical object – called the turtle – on the screen, rather than to manipulate array of numbers or symbols (see Figure 3.1, left). Programming the computer meant programming the turtle. A programmer would give commands for the turtle such as "move forward ten steps and then turn 90 degrees to the right" which in Logo would be "FORWARD 10 RIGHT 90" (see Figure 3.1, middle). The turtle would then move on the screen and thus provide visual feedback on whether or not the program was correct. In addition, the turtle carried a pen with which it could draw, leaving a trace of its steps. The commands "PEN DOWN FORWARD 10 RIGHT 90," executed four times, would result in the drawing of a square on the computer screen (see Figure 3.1, right).

The second feature to note is that the Logo turtle served as a first representative of formal mathematics for children because they could bring to bear their body knowledge on how to move the screen turtle. Consider the following commands, in which the turtle moves one step forward, then moves one degree to the right, and then repeats this procedure 360 times: REPEAT 360 [FORWARD 1 RIGHT 1]. With the pen down, these commands draw a circle on the screen. A child, using his own body, pretending to be the turtle, could execute every single one of these steps. Papert attributed great importance to this feature, which he called *syntonic* learning, because it allowed children to identify with the computational object in multiple ways:

> *For example, the Turtle circle is* body syntonic *in that the circle is firmly related to children's sense and knowledge about their own bodies. Or it is* ego syntonic *in that it is coherent with children's sense of themselves as people with intentions, goals, desires, likes and dislikes. . . . One can also*

> *see it as* cultural syntonic *in that when drawing the circle, the turtle connects the idea of an angle to the idea of navigation which is closely rooted in children's extracurricular experiences. (1980/1993, pp. 63–68)*

The Logo turtle allowed children to manipulate objects on the screen as they would manipulate them in the physical world. Thus, turtle geometry provided a concrete entrance into the formal world of mathematics and allowed learners to connect their personal experiences to mathematical concepts and operations.

A third and equally important feature of Logo programming is the idea of children learning about their own thinking and learning, called reflection or metacognition. Papert claimed that in learning programming, children learn to articulate procedures, recognize repetition, and "debug" their own thinking when programs don't run as expected: "But thinking about learning by analogy with developing a program is a powerful and accessible way to get started on becoming more articulate about one's debugging strategies and more deliberate about improving them" (p. 23, 1980/1993). Computer programs can become "objects-to-think-with" that help children reflect on their performance in ways similar to experienced learners.

Learning Logo thus combined multiple purposes: learning to program, learning mathematics, and learning to learn. These claims did not remain uncontested. So much has been written about the success or failure of Logo in schools that it's worth providing some background to the debate. In an excellent analysis of the historical context in the United States and in Europe, Richard Noss and Celia Hoyles (1996) identify some of the larger cultural forces at play that led critics to ask certain questions about Logo and not others. In many schools, questions about Logo's learning benefits focused exclusively on the transfer of problem solving skills and less on the benefits of learning mathematics and pedagogical reform ideas. A series of smaller studies conducted by Roy Pea and Midian Kurland (1984) is often referenced as the sole evidence that learning Logo programming did not produce any transferable effects. These studies had several methodological issues; for example, neglecting to consider the length of time spent learning programming and the type of programs created. These features have now been recognized as instrumental in designing successful programming instruction (Palumbo, 1990) and I discuss them in more detail in a later section on software design for learning. A further problem has been that teachers often adopt Logo but not the pedagogical innovations for learning mathematics and science; and even if they did, many did not receive widespread institutional support in their schools for doing so (Papert, 1991, 1997). Many institutional forces shaped the use of Logo in schools, and the result was often that its constructionist ideas about learning and teaching were the least acknowledged.

Key Ideas in Constructionism

The name "constructionism" brings to mind the metaphor of learning by constructing one own's knowledge, and is often contrasted to the more traditional "instructionism," which favors the metaphor of learning by transmission of knowledge. Although these two metaphors offer a versatile summary, it is worthwhile to unpack the constructionist idea of knowledge construction and examine its individual and social dynamics. We will then move to the notion of learning cultures and address which features of a learning environment promote successful knowledge construction.

Knowledge Construction

The idea of constructing one's own knowledge draws heavily from Piaget's theory of knowledge development and his instrumental insight that children understand the world in fundamentally different ways than adults. He identified two mechanisms, assimilation and accommodation, that explained how children made sense of

the world they interacted with and how they integrated these experiences into their understanding. Constructionism builds on these mechanisms, and focuses on the processes that help learners make connections with what they already know. A key aspect in knowledge construction is *appropriation* – how learners make knowledge their own and begin to identify with it. These appropriations go beyond the intellectual and include emotional values.

According to Papert, physical objects play a central role in this knowledge construction process. He coined the term "objects-to-think-with" as an illustration of how objects in the physical and digital world (such as programs, robots, and games) can become objects in the mind that help to construct, examine, and revise connections between old and new knowledge. "Objects-to-think-with" such as the Logo turtle are particularly effective at supporting appropriation, because they facilitate the child's identification with the object, or syntonic learning.

Constructionism further differs from the Piagetian model in equally valuing the concrete and the abstract. In Piaget's stage theory, formal abstraction is seen as the ultimate goal of all knowledge construction, with concrete thinking always associated with younger, less advanced children. Turkle and Papert (1990) instead argue that concrete thought could be just as advanced as abstract thought. The sciences in general, but the computer culture in particular, have tended to value abstract thinking. But in studying programmers, Turkle and Papert discovered that the officially promoted top-down or planning approach was not always superior to a more improvised, more bricoleur-like approach. The bricoleur style is not a stepping stone towards more advanced forms of knowledge construction, but rather is a qualitatively different way of organizing one's planning and problem solving.

In sum, knowledge construction is "the deliberate part of learning [which] consists of making connections between mental entities that already exist; new mental entities seem to come into existence in more subtle ways that escape conscious control. . . . This suggests a strategy to facilitate learning by improving the connectivity in the learning environment, by actions on cultures rather than on individuals" (p. 105, Papert, 1993).

Learning Cultures

The importance of learning cultures was informed by Papert's observations of children's difficulties in understanding and learning mathematics. Piagetian studies indicated that all young children develop their first fundamental mathematical concepts, but many struggle in later school years. In his book *Mindstorms*, Papert offered Brazilian samba schools as one possible image for a learning culture:

> *[t]hese are not schools as we know them; they are social clubs with memberships that may range from a few hundred to many thousands. Each club owns a building, a place for dancing and getting together. Members of a samba school go there most weekend evenings to dance, to drink, and to meet their friends. During the year each samba school chooses its theme for the next carnival, the stars are selected, the lyrics are written and rewritten, and the dance is choreographed and practiced. 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. (1980/1993, p. 178)*

Papert's idea of a learning culture has been developed in several directions, from neighborhood centers to virtual worlds. The Computer Clubhouse (Resnick, Rusk, & Cooke, 1998), for instance, is an outside of school learning culture that is located in after school programs and community centers. In these clubhouses, youth convene at their own volition and learn to work with creative software applications to produce digital graphics, music, and videos. Unlike schools, the activities in the clubhouse do not follow a set curriculum, and members are responsible for introducing each other to new activities, with the support of coordinators and mentors. Other examples include multi-user

online environments in which community members contribute to the design of various elements of the online world by populating it with objects and houses, such as Bruckman's MOOSE Crossing (this volume).

What stands out in these examples of the samba school, Computer Clubhouse, and MOOSE Crossing is the rich set of interactions between different community members. These instructional interactions are not formulated in the one-directional pathway of traditional classrooms; rather, they draw on apprenticeship models (see also Collins, this volume) in which all members of the community of practice contribute to the larger enterprise (Lave & Wenger, 1991). Although sociocultural researchers emphasize the social dynamics of learning cultures, constructionists focus on how the social context provides opportunities for making connections to what is being learned.

Logo Microworlds and Construction Kits

The programming language Logo provided a programmable object, the turtle, to facilitate learners' constructing relationships with mathematical concepts and their own thinking in the context of programming. *Microworlds* and *construction kits* have expanded on different aspects of Logo to promote learning in mathematics and science. These applications illustrate how the design of computer applications can be driven by constructionist theory.

Microworlds

Microworlds have been described as "a computer-based interactive learning environment where the prerequisites are built into the system and where learners can become the active, constructing architects of their own learning" (Papert, 1980/1993, p. 122). A classic example is the Dynaturtle, a physics environment in which learners can experience Newtonian physics, and also historically important alternatives like Aristotelian physics. Movements and states of turtles can be preprogrammed to respond to certain laws of motion that can be manipulated by the learner. No explicit instruction about the laws is provided in microworlds, unlike in computer-based tutorials or computer-assisted instruction. Learners induce these laws by interacting with a turtle preprogrammed to behave as an object in a frictionless universe.

Further developments have expanded Logo into massively parallel microworlds on the computer; instead of one turtle, now hundreds or even thousands can interact. In StarLogo (Resnick, 1991), a circle would no longer be drawn by one turtle, but instead by dozens of turtles following two simple rules: (1) to keep a specific distance from each other and (2) to repel the group as a whole and move away from other turtles (see Figure 3.2).

This version of Logo connects to another emergent discipline, that of complex system design, which is interested in how complex behavior patterns emerge from interactions between many simple objects. Many natural and human phenomena can be described this way, as Mitchel Resnick argues in his book *Turtles, Termites, and Traffic Jams* (1994). Working with StarLogo offers learners the opportunity to explore the probabilistic patterns in complex interactions in the same way as the turtle in Logo offers learners an opportunity to connect to formal mathematical objects in new ways (Resnick & Wilensky, 1998). StarLogo can provide accessible objects-to-think-with for people to examine emergence in complex systems.

Microworlds are the prototypical constructionist learning environment for the following reasons. First, scientific and mathematical microworlds offer access to ideas and phenomena – such as the frictionless world – that students may not easily encounter in their regular textbooks or classroom lessons. Second, they provide environments that challenge naïve understandings by providing the learner with feedback on their interactions and manipulations. Third, these interactions with the microworld allow the learner to develop personal knowledge that can provide the foundation for more formalized interactions. Last, microworlds create a type

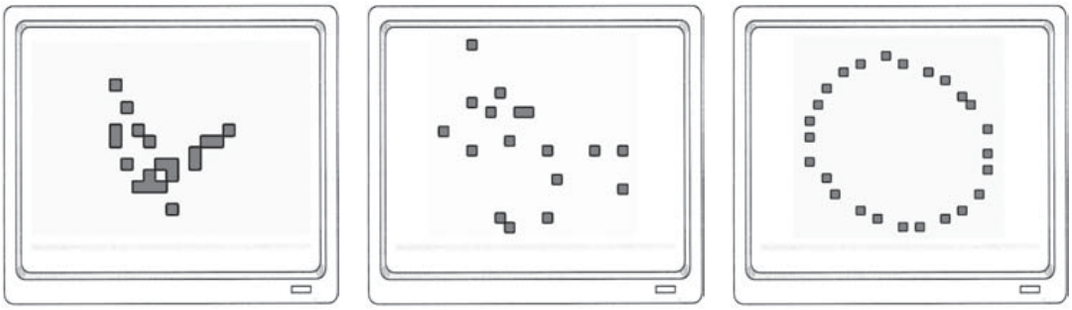

Figure 3.2. These screen shots show a new way to create a circle with StarLogo. No single turtle draws the circle. Rather, the turtles arrange themselves into a circle, based on their interactions with one another. Each turtle follows two simple rules: (1) it tries to keep a certain distance from each of its two "neighbors," and (2) it gently "repels" the group as a whole, trying to move away from the other turtles. With these two rules, the turtles arrange themselves into a circle. (Adapted with permission from the StarLogo Web site at http://education.mit.edu/starlogo/.)

of learning environment in which talking about mathematics (or science) is part of the classroom peer culture. The turtle world in Logo is "a 'place', 'a province of Mathland' where certain kinds of mathematical thinking could hatch and grow with ease. The microworld was an incubator" (Papert, 1980/1993, p. 125). A wide range of microworlds in mathematics and science has been developed since then, and not all of them are Logo-based environments (for more examples of microworlds and further developments see diSessa, 2000; Edwards, 1998; Noss & Hoyles, 1996, this volume).

Construction Kits

LEGO™ building blocks and the programming language Logo were combined to create computationally enhanced construction kits that allow children to explore engineering and architectural design. For example, LEGO™ bricks have been equipped with motors and sensors and a control language to combine the physical and the digital worlds (Resnick & Ocko, 1991). LEGO™/Logo draws on the constructionist tradition of using materials and activities that are already part of children's experiences, but enriches them with computational elements and brings engineering and robotics activities into the classroom and home.

The LEGO™/Logo computational bricks have been employed in a variety of educational contexts, ranging from homes to college classrooms. The Mindstorms robotics competitions, now a part of many high school and college classrooms, specify a goal for the robot, and then give teams a limited amount of time and resources to build a robot out of LEGO™ bricks with motors and sensors; at the end of the time period, the teams test their designs in a competition. One study that followed the students through this process found that this type of combined engineering and programming activity provided the students with hands-on and team experience (Martin, 1996, 2001). In traditional classes college students often have difficulty translating and applying their textbook knowledge into actual robotic design and learning to make distinctions between ideal and real systems.

A further development moves the LEGO™/Logo brick concept into new territory by taking traditional toys such as balls and beads and providing them with computational enhancements (Resnick, 1998). Adding sensors and feedback mechanism to these toys provides them with different interaction possibilities. Providing programmable interfaces lets children not just play with but also design these toys. Another direction is the computational crafts development, which takes traditional craft activities such as origami and uses the computer for the design of materials that can be printed on paper and used for play (Eisenberg, 2003). The design and creation of such polygonic objects gives

children experience with different geometry concepts.

The research group around Seymour Papert at the MIT Media Lab, along with many others, has developed, implemented, and examined different examples of constructionist learning cultures and technologies. A more extensive collection of examples and further theoretical papers can be found in two books published by members of the group, *Constructionism* (Harel & Papert, 1991) and *Constructionism in Practice* (Kafai & Resnick, 1996).

Software Design for Learning – A Constructionist Learning Environment

Design activities play a central role in constructionism. They can facilitate knowledge construction, reformulation, and expression in the process of building shareable artifacts such as robots, software, and games. The *Instructional Software Design Project* (hereafter: ISDP; Harel, 1990; Harel & Papert, 1991) provides an example of how students can engage in these processes as they design instructional software to teach fractions to younger students in their schools – a topic that they were learning about in mathematics class. Classroom practices included students writing in notebooks about their software and instructional designs, discussing fraction representations in class, inviting prospective users for feedback sessions, and conducting software review sessions guided by the teacher.

The curricular model of ISDP responded to several of the criticisms of previous Logo research (Palumbo, 1990). First, it situated the daily programming activities in the classroom rather than in a distant computer laboratory visited only once a week. Second, it integrated the learning of programming with other subject matter such as the learning of fractions, rather than keeping programming isolated from the rest of curriculum. Finally, students were asked to create a meaningful artifact, such as an instructional piece of software to teach younger students in their school, rather than to produce small pieces of program code with no authentic purpose.

The analysis of the ISDP outcomes provided compelling evidence of the benefits for both learning fractions and learning programming, especially when compared to two other classes at the same school that either had programming only once a week or daily but without the focus of creating instructional fraction software. In all these comparisons, ISDP students improved significantly – not only in their programming skills but also in their conceptual and procedural understanding of fractions. The instructional software designed by students illustrated personal choices in representations that students created for their learners. Individual interviews also revealed increased metacognitive competence in juggling the multiple demands of learning by design (see Kolodner, this volume). In addition, ISDP students showed increased persistence in debugging Logo programming problems and in their ability to manage the multiple learning demands of software design. The importance of having students create shareable artifacts such as instructional software is one example of the increased attention that the learning sciences are paying to providing authentic learning activities, products, and tools in learning environments. In addition, the provision of an audience other than the teacher for the learning products is another important feature of many learning sciences projects.

Building on the successful design of the first ISDP version, a subsequent version added an apprenticeship component that illustrates how connections across grade levels can help to create a learning culture. Collaborative interactions are a key component of many environments and curricula in the learning sciences. By setting up software design teams rather than individual designers, and composing teams of students with and without prior software design experience, we showed that young designers can bring to bear their previous software design experiences in multiple ways: by initiating and expanding science conversations in groups (Kafai & Ching, 2001), by helping younger inexperienced team members with planning their instructional designs (Marshall, 2000), and by providing

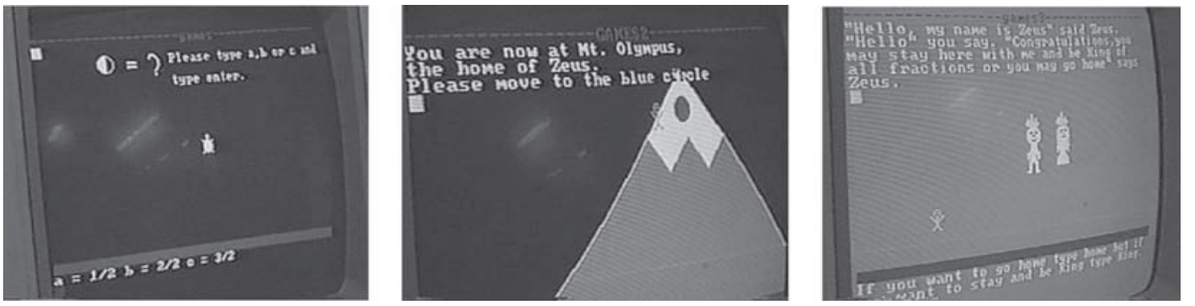

Figure 3.3. Fraction game designs by a girl. The three screen shots showcase Amy's Greek Myths fraction game in which a player has to assemble a map ripped into pieces to finally meet the gods and goddesses of fractions at Mount Olympus. The left screen displays a fraction problem whereas the middle and right screens show game components of Mount Olympus and a meeting with gods and goddesses of fractions.

programming assistance when needed (Ching, 2000).

In a comparison study (Ching, 2000) with software design teams of only inexperienced older and younger students, we found that the quality of collaborative helping interactions also shifted dramatically: teams with experienced software designers provided more collaborative assistance rather than taking over programming tasks, provided room for making mistakes while still monitoring programming activities, and provided more access to computer resources when needed for less experienced members. Our analyses indicated that experience and not age was a decisive factor in how student designers handled programming and collaborative interactions. Apprenticeship, as a model for collaborative interactions, was key in distributing responsibilities in teams and in the classroom with the teacher.

But the most important finding resulted from the analysis of apprenticeship interactions in teams when comparing both setups. Students working with experienced team members were provided with more flexible and collaborative work arrangements. In contrast, students working with older inexperienced students were often put in more supervised activities and not involved in programming activities; their opportunities to develop independent programming skills were largely reduced because older team members directed all their activities, trying to prevent mistakes. These different teams also resulted in different understandings of roles in the projects. Experienced software designers addressed a much richer set of roles involving planning, helping, teaching, and understanding younger students' concerns and anxieties. These perceptions also change over time, as found in one of the few longitudinal examinations of long-term programming learning (Kafai & Roberts, 2002).

Although instructional software design is safely grounded in school culture, whether it's purchased in the form of commercial software or designed by the students themselves, entertainment media like video games often do not enter the classroom. In a continuation of the ISDP project, a class of ten-year-old children was asked to design and program their own video games. The children met every day over a period of six months to design games by creating their own characters, story lines, game themes, and interactions. Here again, we found benefits in learning programming when compared to other classes not engaged in extended programming (Kafai, 1995).

It is worthwhile to take a closer look at the games created by the students and examine them as microworlds from an instructional and game perspective (Kafai, 1996). The game design activity offered a microworld in which both girls and boys could situate their preferred ideas and fantasies. The most distinctive feature, however, is the degree to which gender differences permeate nearly all aspects of game design. Nearly all the games created by boys featured fantasy contexts, with many characters and violent feedback, when compared to the games designed by girls (see Figures 3.3 and 3.4).

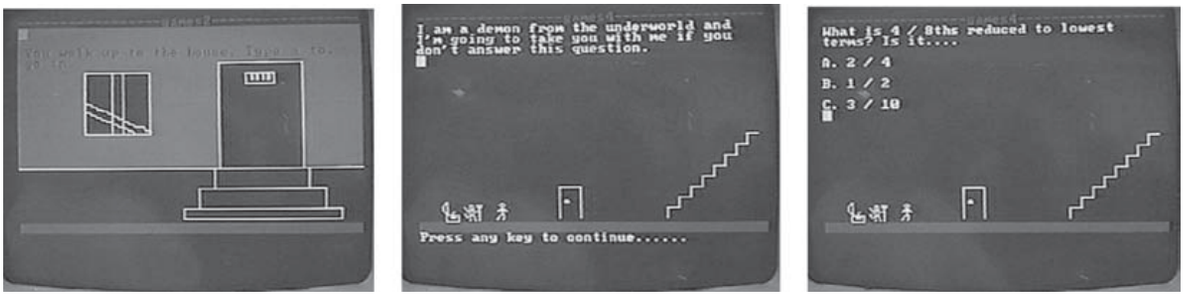

Figure 3.4. Fraction game designs by a boy. Albert designed a haunted house fraction game in which a player explored the different rooms of a house such as the one displayed in the right screen shot. When he opens the treasure chest, a demon jumps out and asks a fraction question (middle screen). A wrong answer choice sends the player to the underworld, eating fried food for the remainder of his life.

Almost all of the boys, for example, created adventure hunts and explorations, whereas the girls' games were more evenly divided among adventure, skill/sport, or teaching. In their choices of game themes and their programming of animation and interactions, the students offered a glimpse into what they found appealing and unappealing in the games and stories they experience through other media. Making a game and its rules allowed the game designers to be in charge and to determine the player's place and role in a virtual world, with all the consequences.

Unlike the microworlds discussed in previous sections, the nature of pedagogical interactions favored by most student designers was that of drill-and-practice. Nearly all games featured multiple-choice questions and expected the prospective learner to provide the answers. There were few constructive elements for learners in these games. We found in follow-up research that children do have models for constructive game activities, but often assume that teaching is about asking questions and learning is about giving answers (Kafai, Franke, Ching, & Shih, 1998).

Under Construction

Constructionism presents a particular combination of individualistic, cognitive processes – with its emphasis on personal appropriation and knowledge construction – and of more social, cultural processes – with its focus on the design of and participation in learning cultures. Microworlds and construction kits have become constructionism's most popular educational software because they combine both cultural and personal aspects. They are cultural because they embody particular disciplinary ideas, and they are personal because they allow for individualized expression of these ideas.

One aspect that deserves further treatment concerns the development of knowledge and its personal connections. I have discussed the importance of connectedness in the process of constructing knowledge, of connecting new ideas to old existing ones, and of facilitating the building of personal relationships with knowledge. Yet in the learning sciences, this aspect is still lacking attention. Learning is often portrayed as a matter of developing disciplinary understanding and practices in the sciences. The learning sciences often build on motivation in their project-based approaches (see Blumenfeld, Kemplar, & Krajcik, this volume) but they do not address "knowledge as desire" as Hans Furth (1987) once articulated in his provocative essay integrating Piagetian and Freudian perspectives to combine cognitive and emotional aspects of learning. Future research needs to expand these combinations of disciplinary practices and interest to develop a better understanding of how learning can tie into the socioemotional personal lives of learners.

The concept of learning cultures has provided a helpful metaphor in designing

successful learning environments. Many learning scientists model their learning environments on studies of how professional practice takes place. But in doing this, they necessarily choose one particular model of professional practice, while in most professions, there are multiple approaches in use. For example, many professional programmers use a top-down, more abstract style, and in fact this style is taught in engineering schools. But other successful programmers use a more concrete, bricoleur style. If learning scientists design classrooms based on only one model of professional practice, they risk misrepresenting the full range of successful practices. This has a particular impact on the issue of gender representation in science, mathematics, and engineering, because females often prefer approaches that do not correspond to the officially valued practices in the profession. Research in the sciences has provided ample evidence of nonvalued but equally successful practices as in the case of the Nobel Prize–winner Barbara McClintock (Keller, 1983). The issue at hand is then what kind of images of a learning culture and of practices do we follow and where might we create new ones – especially if we are interested in having learners not just follow the beaten path but create new venues. This is a challenge for the next generation of tools and environments in the learning sciences.

To conclude, in this chapter I examined key aspects of how we can design constructionist learning environments, technologies, and activities that create supportive learning cultures. Microworlds and construction kits illustrated that the design of learning technologies comes wrapped in a theory of mind coupled with disciplinary understanding. Many developments in the learning sciences are influenced by this premise and continue to develop variations on microworlds and construction kits with additional scaffolds to support learners' inquiry, collaboration, and reflection processes (see the chapters in this volume by Edelson & Reiser; Noss & Hoyles; Pea & Maldonado; Stahl, Koschmann, & Suthers). Software design for learning illustrated that technologies need to be integrated within a larger learning culture. Many curricular efforts in the learning sciences have adopted project-based learning approaches to create motivating and authentic contexts for learners to develop and practice their skills (e.g., Krajcik & Blumenfeld, this volume; Linn; Scardamalia & Bereiter, this volume; Songer, this volume). Constructionist theory challenges us to consider individual and sociocultural aspects in the design and investigation of the learning sciences.

Acknowledgments

The author's work described in the chapter was supported by an EARLY CAREER grant of the National Science Foundation (NSF-9632695). The writing of this chapter was supported in part by a grant of the National Science Foundation (NSF-0325828). The views expressed are those of the author and do not necessarily represent the views of the supporting funding agency or the University of California, Los Angeles.

References

Ching, C. C. (2000). *Apprenticeship, learning and technology: Children as oldtimers and newcomers in the culture of learning through design*. Unpublished doctoral dissertation. University of California, Los Angeles.

diSessa, A. (2000). *Changing minds: Computers, Learning and Literacy*. Cambridge, MA: MIT Press.

Edwards, L. (1998). Embodying mathematics and science: Microworlds as representations. *Journal of Mathematical Behavior*, *17*(1), 53–78.

Eisenberg, M. (2003). Mindstuff: Educational technology beyond the computer. *Convergence*, *4*, 45–76.

Furth, H. G. (1987). *Knowledge as desire: An essay on Freud and Piaget*. New York: Columbia University Press.

Harel, I. (1990). *Children designers*. Norwood, NJ: Ablex.

Harel, I., & Papert, S. (1991). Software design as a learning environment. *Interactive Learning Environments*, *1*(1), 1–30.

Kafai, Y. B. (1995). *Minds in play: Computer game design as a context for children's learning*. Hillsdale, NJ: Lawrence Erlbaum Associates.

Kafai, Y. B. (1996). Gender differences in children's constructions of video games. In Patricia M. Greenfield & Rodney R. Cocking (Eds.), *Interacting with video* (pp. 39–66). Norwood, NJ: Ablex Publishing Corporation.

Kafai, Y. B., & Ching, C. C. (2001). Affordances of collaborative software design planning for elementary students' science talk. *The Journal of the Learning Sciences*, *10*(3), 323–363.

Kafai, Y. B., Franke, M., Ching, C., & Shih, J. (1998). Games as interactive learning environments fostering teachers' and students' mathematical thinking. *International Journal of Computers for Mathematical Learning*, *3*(2), 149–193.

Kafai, Y. B., & Resnick, M. (1996). *Constructionism in practice*. Mahwah, NJ: Lawrence Erlbaum Associates.

Kafai, Y. B., & Roberts, M. (2002). On becoming junior software designers. In R. Stevens & P. Bell (Eds.), *Proceedings of the Fifth International Conference on the Learning Sciences* (pp. 191–198). Mahwah, NJ: Erlbaum.

Keller, E. F. (1983). *A feeling for the organism: The life and work of Barbara McClintock*. San Francisco: W. H. Freeman.

Lave, J., & Wenger, E. (1991). *Situated learning: Legitimate peripheral participation*. London: Cambridge University Press.

Marshall, S. (2000). *Planning in context: A situated view of children's management of science projects*. Unpublished doctoral dissertation. University of California, Los Angeles.

Martin, F. (1996). Ideal and real systems: A study of notions of control in undergraduates who design robots. In Y. Kafai & M. Resnick (Eds.), *Constructionism in practice* (pp. 255–268). Mahwah, NJ: Lawrence Erlbaum Associates.

Martin, F. (2001). *Robotic Explorations: A Hands-on introduction to engineering*. New York: Prentice Hall.

Noss, R., & Hoyles, C. (1996). *Windows on mathematical meanings: Learning cultures and computers*. Dordrecht: Kuwer Academic Publishers.

Palumbo, D. (1990). Programming language/problem-solving research: A review of relevant issues. *Review of Educational Research*, *45*, 65–89.

Papert, S. (1980/1993). *Mindstorms* (2nd ed.). New York: Basic Books.

Papert, S. (1987). Computer criticism versus technocentric thinking. *Educational Researcher*, *16*(1), 24–28.

Papert, S. (1991). Situating constructionism. In I. Harel & S. Papert (Eds.), *Constructionism* (pp. 1–14). Hillsdale, NJ: Lawrence Erlbaum Associates.

Papert, S. (1993). *The children's machine: Rethinking school in the age of the computer*. New York: Basic Books.

Papert, S. (1997). Tinkering towards utopia: A century of public school reform. *Journal of the Learning Sciences*, *6*(4), 417–427.

Pea, R., & Kurland, M. (1984). On the cognitive effects of learning computer programming. *New Ideas in Psychology*, *2*(2), 137–168.

Resnick, M. (1991). New paradigms for computing, new paradigms for thinking. In Y. Kafai & M. Resnick (Eds.), *Constructionism in Practice* (pp. 255–268). Mahwah, NJ: Lawrence Erlbaum Associates.

Resnick, M. (1994). *Turtles, Termites, and Traffic Jams*. Cambridge, MA: MIT Press.

Resnick, M. (1998). Technologies for life long learning. In *Educational Technology, Research & Development*, *46*(4), 43–55.

Resnick, M., & Ocko, S. (1991). LEGO/Logo: Learning through and about design. In I. Harel & S. Papert (Eds.), *Constructionism* (pp. 141–150). Hillsdale, NJ: Lawrence Erlbaum Associates.

Resnick, M., Rusk, N., & Cooke, S. (1998). The computer clubhouse: Technological fluency in the inner-city. In D. Schon, B. Sanyal, & W. Mitchell (Eds.), *High Technology and Low Income Communities* (pp. 266–286). Cambridge, MA: MIT Press.

Resnick, M., & Wilensky, U. (1998). Diving into complexity: Developing probabilistic decentralized thinking through role-playing activities. *Journal of the Learning Sciences*, *7*(2), 153–172.

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

Turkle, S., & Papert, S. (1990). Epistemological pluralism and the reevaluation of the concrete. *Signs*, *16*(1), 128–157.