J. EDUCATIONAL COMPUTING RESEARCH, Vol. 37(2) 131-150, 2007

COMPUTER PROGRAMMING IN MIDDLE SCHOOL: HOW PAIRS RESPOND TO CHALLENGES*

JILL DENNER
Education, Training, Research Associates

LINDA WERNER
University of California, Santa Cruz

ABSTRACT

Many believe that girls lack the confidence and motivation to persist with computers when they face a challenge. In order to increase the number of girls and women in information technology careers, we need a better understanding of how they think about and solve problems while working on the computer. In this article, we describe a qualitative study of 126 middle school girls who designed and programmed computer games in an after-school and summer program. Using data from electronic notebooks, we describe how girls thought about the problems they had while programming their games and the strategies they used in their efforts to solve them. Audiotape transcripts were also coded to show how girls talk about challenges and the steps they take to address them when programming a game on the computer. The findings are interpreted in terms of how to promote information technology fluency starting in middle school.

The Committee on Equal Opportunities in Science and Engineering (2004) cites the importance of not just attracting diverse students, but also of retaining them in courses and increasing their persistence in information technology (IT) fields. The percent of bachelor's degrees in computer science awarded to women fell

*This material is based on work supported by the National Science Foundation under grant no. 0217221. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

from 37% in 1985 to 25% in 2004 (National Science Foundation, 2006) and only 27% of workers in the areas of computer and mathematical operations are female (U.S. Department of Labor, 2005). The gender gap remains, in part, because it is not entirely clear why girls and women drop out of computer science courses and IT careers. Although there are good descriptions of how social and attitudinal factors create barriers to girls' and women's retention (Creamer, Burger, & Meszaros, 2004; Margolis & Fisher, 2002), little is known about what happens when girls are actually working on the computer. In particular, we know little about how girls experience and respond when facing challenges while using the computer. To address this gap, this study draws on data from an after-school and summer program in which middle-school girls design and program a computer game with a peer.

The basis of this study is that in order to increase the number of girls and women who produce and persist with information technology, we need to increase their willingness and ability to identify and deal with challenges and setbacks. To date, research has led many people to conclude that girls lack the confidence, motivation, and support to persist when faced with challenges at the computer (Margolis & Fisher, 2002). In fact, girls' and women's interactions with computers are usually described as filled with anxiety or fear that they will fail or break the machine (Cooper & Weaver, 2003). Instead of being motivated by a challenge, research suggests that when girls are interested in technology, they are motivated by social interaction or helping others (Creamer et al., 2004). But among those who dominate the IT workforce, the ability to work well with other people is seen as inconsistent with the ability to problem solve (von Hellens, Nielsen, & Beekhuyzen, 2004).

Based on this research, many educational practitioners and researchers perpetuate the myth that girls are unable to or are uninterested in persisting in the face of problems or challenges. However, studies suggest that some adult women are drawn to computers because they like to explore and solve problems (Tillberg & Cohoon, 2005; Yates & Littleton, 2001). In fact, many women who persist in college computer science courses have a way of thinking about problems and problem solving that gives them more confidence to overcome challenges (Margolis & Fisher, 2002). Some view computer programming as the process of creating solutions to problems, and that is itself a motivation (Bruce et al., 2004). In a study using the same data as the present one, Turner and Denner (under review) found that when unable to solve a problem, pairs of girls were not significantly more likely to blame themselves than to blame another person or the nature of the task. This lends hope to the idea that girls do not automatically blame themselves when problems arise, and may be more likely than previously thought to take credit for their successes. Thus, in order to develop strategies to increase the numbers of girls and women in the field of computer science, we need to understand how they think about and respond to challenges while working at the computer.

RESEARCH ON PROBLEM SOLVING AT THE COMPUTER

Although there is extensive research on collaborative, computer-based problem solving of discrete tasks with a correct answer (Barron, 2000; Parnafes & DiSessa, 2004), little is known about how students experience and respond to open-ended problems. Open-ended problems are those more typically encountered by users doing design or programming tasks because there is more than one way to approach or solve them. A focus on open-ended problem solving is consistent with a recent shift in focus from computer literacy to fluency with information technology. Some key aspects of fluency are the ability to identify specific problems, to try different solutions over a period of time, and to manage problems in faulty situations (National Research Council, 1999). Our focus on open-ended problems instead of closed-ended problems reflects one difference between real-world versus contrived problems in information technology and gives us an opportunity to look at problem solving over longer periods of time.

Research in computer science suggests that one's problem-solving approach is influenced by the type of problem. For example, Brooks' (1987) distinction between accidental and essential problems has been influential in computer programming. Accidental difficulties occur when the tools for creating software are imperfect, which leads to problems that could be avoided with better tools. Essential difficulties are those that require human thought in order to solve them, and will not go away even if the computer software and hardware are improved. A person's perception of what kind of difficulty they are facing has strong implications for whether and how they think it can be fixed. Yet we know little about how children think about or respond to challenges they face on the computer.

Previous research provides some indication of how students might respond to challenges while computer programming. Turkle and Papert (1992) identified two distinct ways that people approach open-ended problems. They describe the bricoleur as someone who makes mid-course changes as he or she works and considers mistakes to be an opportunity to test creative ideas. They contrast this style with the planner who programs using a top-down approach, which involves outlining and following specific steps with a clear endpoint in mind. Research in the last decade has often assumed that gender plays a role in how a person learns to program. However, McKenna (2001, 2004) found no significant difference between females and males learning to program in regards to the use of prepackaged routines (based on survey data) with the implication that there is not a gendered approach toward abstract versus concrete learning-to-program strategies. In addition, a study of fourth grade students found that most used a mixture of planning and bricolage—only four of the 16 students used primarily one approach to designing and programming games (Kafai,

1995). This suggests that the ability to move between being a top-down planner and a creative bricoleur will enable a range of students to persist in the different cultures of computing.

Rather than a programming style, McKenna (2001) suggests we focus on the strategies people use when they learn to program. For example, he describes the strategy of black boxing and abstracting detail, which allows the user to set aside information that they do not need to know in order to solve the problem. Another strategy involves reduction, where the problem is compared to one for which they already know the solution (Armoni, Gal-Ezer, & Tirosh, 2005). Strategies will vary depending on how people think about problem solving and the motivation that drives their work on the computer. For example, people who experience learning to program a computer as a series of puzzles that need to be solved are motivated by engaging in the process of responding to challenges (Bruce et al., 2004). Others want to get the answer as quickly as possible, while still others are less concerned about solving the problem than they are with understanding the big picture or being seen as a programmer by others (Bruce et al., 2004).

The research is not clear on whether there are gender differences with regard to style. A small study with middle-school students found that girls focus more on the functions of the technology needed to complete a task as efficiently as possible, while boys were more interested in exploring the features (Hou et al., 2006). Similarly, von Hellens, Nielsen, and Beekhyzen (2004) find that female IT professionals identified dualisms in the IT industry that make it hard for women to be seen as both good problem solvers and good at talking with other people. However, Beckwith et al. (2006) find that women were as likely as men to tinker with new features when there was sufficient explanation of what was happening and what could be done next ("high support") embedded in the software. The current study extends this research on mostly adult-computer interaction to examine how middle school students engage in open-ended problem solving.

The current study also extends previous research because most studies of problem solving at the computer focus on individuals. However, collaboration can increase individual problem solving (Barron, 2000), probably because of the need to articulate, negotiate, and monitor their strategies. Studies of children working at the computer (but not programming) suggest that effective collaboration helps them perform better, build confidence, and increase motivation to continue playing (Inkpen, Booth, Klawe, & Upitis, 1995; Littleton & Light, 1999). In pair programming (PP), two students share one computer (Barker & Cohoon, 2006; Williams, Kessler, Cunningham, & Jeffries, 2000), and there are clear roles: one is the driver who works the keyboard and mouse while the other navigates. Pair programming promotes persistence among university computer science students (McDowell, Werner, Bullock, & Fernald, 2003; Werner, Hanks, McDowell, Bullock, & Fernald, 2005). However, there are no known studies of PP in middle school.

In summary, in order for educators to support girls to persist in the field of computer science, research needs to provide a better understanding of the kinds of challenges they face while working on the computer, and how they respond. The current study is designed to fill a gap in studies of collaborative problem solving in real-world settings.

CURRENT STUDY

An after-school and summer program that offers computer game design with a peer provides the setting for this study of how students think about and respond to challenges on the computer. This approach builds on Kafai's (1995) study of game design by individual students in school. This study also builds on sociocultural theory which describes thinking as a social process in which students learn by interacting with other people (Vygotsky, 1978). Like others, we aim to understand how children learn by focusing on their social interaction (Perret-Clermont, Perret, & Bell, 1991) with the goal of understanding the different ways in which girls experience and respond to challenges on the computer. Although the data do not allow us to make generalizations to how individuals interact with technology, they do allow us to hear how girls are thinking about and negotiating problems while making a computer game with a peer.

METHOD

Participants

Data were collected from 126 girls who were voluntary participants in an after-school and summer program called Girls Creating Games (GCG). They all live in a small urban community in Central California, and range in age from 10-14 years (M = 11.75, SD = 1.0). The girls' self-reported ethnicity was mostly white (58%) and Hispanic/Latina (31%); 35% reported that they speak a language other than English at home at least some of the time.

Program Description

The goal of the program is to increase middle-school girls' interest, confidence, and skills to pursue courses and careers in information technology. The approach was based in the constructionist theory called "learning by design" in which students are actively involved in building knowledge by creating products like computer games (Harel, 1991). Four program strategies were implemented over 23 sessions: learning by design; identity formation; collaborative learning; and teaching as assisted performance. These approaches are described in detail in another publication (Denner, Werner, Bean, & Campe, 2005). All participants completed a game that used a "choose your own adventure" format in which there

is a narrative, and the player must choose what happens at key points in the story. In this article, we focus on girls' activities while at the computer, while they designed and programmed a narrative game using the software Flash MX from Macromedia.

Procedure

Each pair of girls had an electronic notebook. During program meetings, girls typed their answers to question prompts; their responses were password protected, and no names were attached. Program leaders encouraged the girls to be thoughtful and honest in their responses. Some girls chose to write as a pair, while others took turns writing in the first person. The program was implemented six separate times. During the last semester of implementation, 10 pairs of girls were audiotaped for 30-minutes during a session about halfway through the program while they worked at the computer on their games.

Instruments

Data from students' electronic notebooks and from audiotape transcripts of pairs were used to describe how the girls thought about problems and the strategies they used to solve them. Data from the electronic notebooks includes girls' written responses to the following questions: "Describe at least one time you got stuck in Flash today. How did you figure out what to do?" and "What problems do you have with using Flash today? How have you solved some of the problems?"

Audiotape transcripts reveal what pairs talked about when faced with a challenge, if and how they tried to figure it out themselves, and what led up to them asking for help or using other resources. The transcripts are conversations between two girls working together, with one student designated as "A" and the other as "B." In some of the transcripts, students refer to instructional materials provided by the GCG program, such as flowcharts of the steps needed to perform a task in Flash.

Data Analysis

The data were coded in a multi-step process. Following guidelines for analyzing qualitative data (Auerbach & Silverstein, 2003; Miles & Huberman, 1994) we identified our research questions and expected themes, and then two researchers separately read through the transcripts to identify sections where students were talking about a challenge. In subsequent readings, the researchers sorted excerpts of the text into key themes and categories and entered these codes into an Excel spreadsheet. Each code was reviewed by both researchers until 100% agreement was achieved.

To examine how problems were defined in the girls' electronic notebooks, we first coded whether each entry did or did not identify a challenge. Then the type of

challenge was coded inductively, as described above, and resulted in eight specific categories (e.g., problems with the timeline). Girls' descriptions of how they tried to solve the challenge or figure out what to do were also coded inductively. Four categories resulted: 1) asked for help from an adult only; 2) asked for help plus tried their own strategy; 3) tried to solve their problem on their own; and 4) asked for help from their partner or another peer. Finally, each entry was coded for whether or not the challenge had been resolved.

Data analysis of the audiotapes focused on the overall structure and process of the dialogue, and their conceptual understanding of problems and how to solve them. Transcripts were coded to identify the problem solving strategies that previous research (Armoni et al., 2005; Beckwith et al., 2006; McKenna, 2001) suggests are used by adults learning to program computers. There are five strategies: 1) identify a problem; 2) use documentation to determine if you are following the instructions correctly; 3) compare the non-working instance to a working instance to determine if there are differences that can shed light on the source and cause of the problem; 4) tinker; and 5) ask an expert.

Following other studies that use peer interactions to understand problem solving (Barron, 2000; Goos, Galbraith, & Renshaw, 2002), we identified episodes that represent different kinds of problem-solving behaviors. The analysis of these transcripts pays close attention to how they identify that something is wrong, how they respond individually and as a pair to the challenge, and the role of their interactions in moving toward a solution.

RESULTS

The findings are organized into three sections. The first section provides a summary of how girls experience challenges on the computer, focusing on how they describe their difficulties. The next section contains a description of the strategies that the girls report they use when faced with a challenge. Finally, the audiotapes are used to describe how students interact when they encounter a challenge while working at the computer.

Difficulties with the Flash Software

There were 231 notebook entries in response to the questions "Describe at least one time that you got stuck in Flash today. How did you figure out what to do?" and "What problems do you have with using Flash today? How have you solved some of the problems?" There were 136 entries (59%) that stated they had a problem or got stuck. Of these, 82 (60%) stated they solved it and 119 (88%) described the type of problem or how they got stuck. The different wording allowed us to examine whether responses varied depending on how the question was asked. There were 139 unique responses to the question about getting stuck and 92 responses to the question about having a problem. As shown in Table 1, responses were different, depending on how the question was asked. Students

Table 1. Rates of Problems or Getting Stuck

	Had a problem or got stuck	Described the type of problem or how they got stuck	Solved it
Both questions combined	59%	87%	60%
Describe at least one time you got stuck in Flash	76%	87%	61%
What problems do you have with using Flash?	34%	87%	55%

were more likely to say they got stuck (76%) than to say that they had a problem (34%). The extent to which they figured it out or solved it was similar for both questions, but students were more likely to describe how they got stuck than to describe the nature of their problem.

The vast majority of problems that students described appeared to be accidental difficulties (according to the definition by Brooks, 1987) because they dealt with the complexities of the Flash software, rather than the inherent complexities of their game narratives and graphics. In other words, their responses imply that the problems could have been prevented if the software or hardware had been easier to understand and use. Although quite a bit was done by the instructors to reduce the complexity of the software by requiring students to make a specific type of game and by only teaching those constructs that were necessary to make that type of game, some problems remained. These included problems with the timeline, the Actionscript programming language, and using graphics tools. Table 2 provides a summary of the types of challenges that students reported.

Below are examples of notebook entries that serve as examples of what the most common difficulties entailed. For example, timeline or keyframe difficulties had mostly to do with animation, sounds, and the layering system in Flash. One entry read: "In our playfile we had problems trying to make our snowman turn a degree. We also wanted to make the timing slower so you could see the snowman moving." Problems with Actionscript occurred primarily with the "go to" command in Flash, which allows the player to move to different scenes in the game: "We got stuck when we changed the buttons that knew where to go, and as soon as it was changed it lost the 'where-to-go' info." Entries that described problems with understanding symbols had to do with making buttons and creating sound effects: "We got stuck with buttons because they weren't supposed to be buttons they were just meant for decoration." When students had difficulties with using Flash tools, it was mostly with the drawing tools that can be used to

Table 2. Types of Challenges and Whether They Were Solved

	Number of entries	Percent solved
Timeline or keyframe	39	74%
Actionscript	28	64%
Using Flash tools	15	73%
Understanding symbols	14	78%
File management	7	57%
Computer malfunction	6	33%
Pair programming	5	40%
Internet access	5	40%

make colors or pictures, or to manipulate text. For example, "One time that we got stuck today was when we wanted to know how to have a rainbow background."

There were 34 entries where students stated that they had no problem; in seven of these they also described a problem. In these cases, the problem that was described was coded, but these entries seemed to indicate that some girls defined a "problem" as something that was never solved. The following notebook entry provides an example: "Hello! We didn't really have any problems today, we really understood everything that we had to do. Whenever I didn't really understand something I looked at my flow chart." This entry also suggests that some girls saw problems as something to avoid, rather than an inherent part of programming a game.

The examples above are all accidental difficulties because they involve problems with the complexities of the Flash software. The following quote is an example of an essential difficulty, where the students struggled with the complexities of the game design rather than the software: "I got stuck when my partner and I did a test movie and we pressed the 'credits' button and it went to Scene A and we didn't understand what happened." In this example, the students had difficulty keeping track of the multiple scenes in their game and how they should be connected. However, as described above, most of the girls' responses can be framed in terms of the complexity of the software (as accidental difficulties). In the next section, we summarize the strategies that students reported using in response to their difficulties.

Reported Problem-Solving Strategies

Of the 136 notebook entries where students said that they had a problem or got stuck, in 99 (71%) the students identified at least one way they tried to solve or

figure it out; 40 (29%) identified no strategy. Of those 99, the strategies varied, as shown in Table 3.

The strategies were similar for responses to both questions (being stuck or having a problem) except 10% reported help from a partner in response to being stuck, while only 1% reported help from a partner in response to having a problem. One possible reason that there were so few who described getting help from a partner was that most faced challenges and wrote responses as a pair. Examples of the different strategies are below.

> Our computer kept not working by not going to flash then it wouldn't import an image that we wanted it to. *We solved it* by choosing another image that worked and we signed on to flash by going to MMC 110 or whatever to get to flash another way.
>
> One time we got stuck today was when we had an overlapping sound effects layer. We *asked for some help by the teacher* and got it set up right.
>
> Our music would not play. *We tried to figure it out* but could not. *We asked for help* and figured it out (on our own!) cause we are on fire! Dynamite!
>
> I knew what kind of button I wanted to make, but I didn't know how to draw it the way I wanted. *I asked my partner for help* and she helped me.
>
> The problem we had today was that we couldn't find the sounds we needed. We solved it by *asking for help from another group*.

In summary, data from the electronic notebooks provides information about the frequency and types of challenges that students faced, whether they were solved, and the strategies used to solve them. Most of the problems were described as "accidental," and 50% of all strategies involved asking an adult for help. In the next section, data from the audiotapes of pairs are used to describe what problem solving actually looked like.

Table 3. Strategies in Response to Challenges

Strategy	Percent
Tried to figure it out on their own (only)	35%
Got help from an adult only	28%
Got help from an adult plus tried something on their own	22%
Got help from their partner	9%
Got help from peers only	3%

Interactions while Problem Solving

Audiotape recordings during game production provide more information about how girls think about and respond to challenges while working on a computer game design task with a peer. Excerpts of three transcripts are included to illustrate common patterns in girls' interactions as they addressed challenges at the computer. Partner A is the navigator and Partner B is the driver of the pair, and each partner's conversational turn is numbered as a move.

In the first excerpt, the students are trying to figure out how to add music to their game. They have put the music on a separate layer, but it does not play. Layering is used in Flash to deal with complexity. Different pieces of the graphics of a scene can be separated by putting each piece on a different layer, and when changes are needed, only the associated layer needs to be modified. In move 1, they identify the problem. In the next move, Partner A reads out loud the information from the flowchart about how to add music to their game. She continues to read directions, even when Partner B repeats that it is not working. In move 10, Partner A disagrees that it did not work. Next, Partner B uses a reduction strategy when she suggests they look at another instance of the use of music and see the difference between the two, drawing on previous experience to solve their problem. However, before they can debug, Partner B must both convince herself and her partner that there is a problem. They discover that the two uses of music in their game appear different (see moves 11 and 13). Verbalizing this process is a way for Partner B to teach her partner about the debugging process. The two partners try to remember how they successfully did an earlier task (move 15), but neither can remember. After a brief exchange that does not involve tinkering, they agree to ask a teacher for help (move 18).

1. B: That's weird it doesn't work. I mean maybe.
2. A: Okay choose a music clip from the library. [reading flowchart]. Is that where we're at? Where we're at? Okay so go to Common Libraries. Window, Common Libraries.
3. B: Hold on, this isn't working.
4. A: OK. Choose the music list from Common Libraries. So go to Window, Common Libraries. [reading flowchart]
5. B: This isn't working.
6. A: And then, click.
7. B: Where is the music?
8. A: click and drag, sound, click and drag sound to stage.
9. B: Okay. That's weird it didn't work.
10. A: Yeah it did. It might of.
11. B: See on the other one? That's what it did. OK let's find the music. See look at the music. It's different.
12. A: Oh.
13. B: See there's a line above it.
14. A: Oh.

15. B: I don't know what we did for that? What did you do?
16. A: I don't know. When did we do that?
17. B: 'Cause it works. And that way it doesn't work.
18. A: Let's ask for Cathy's help. Um we can't get the music.
19. B: Yeah, we still can't get the music on.

In the following transcript, the students are using the "paste in place" command to copy and paste a hand-drawn image of a button that says "Go to Air Show" from one part of their game into another scene. In move 1, the navigator identifies the problem and immediately suggests deleting it and starting over. In move 3, she suggests looking at documentation to solve the problem, and then suggests a different strategy that involves getting some clip art instead. Both of these suggestions are ignored as the girls become playful and focus on the tape recorder (the "it" in move 5). The next suggestion (move 8) is to ask an adult but they continue to playfully work toward a solution together. Ultimately, an adult helps them figure out how to paste the button in place on a new layer.

1. A: Omigod how come we can't do paste in place? How come we can't do paste in place? OK why don't you just delete that whole thing?
2. B: Wait, why isn't it doing it?
3. A: Because you are doing it wrong. I don't know how to do it either. Let's look on the flow chart! If they have one . . . um no. No. No. No. Hey Nancy wanna use some clip art or no?
4. B: I'm not! Lila!
5. A: I told you it heard us.
6. B: OK how do we do this Lila!
7. A: Hey!
8. B: Amy! I mean Cathy!
9. A: Hey yanna, Ho yahh, ho . . . ahh! Excuse me we need our privacy for one moment please. No I don't want to see it again.
10. B: No I don't know where it is!
11. A: I don't want to see it again.
12. B: Yeah, yea all you people.
13. A: Hey you are right it does. . . . Welcome, I said Welcome to the YMCA! I saw you bonding with the computer!
14. B: Oh yeah?
15. A: They are kissing! To the YMCA. . . . We need to put that into, into . . .
16. B: That's the third time that's happened to you.
17. A: We need to put this, the "go-to air show" in a new layer. But it's in this layer.
18. B: OK.

In the following excerpt, the partners are discussing which colors to use and whether certain kinds of colors can even be used. They run into a problem when they can not display the chosen color (move 2). They go straight to tinkering, but disagree on how to go about it. To ultimately solve the problem, the navigator ends up taking the mouse from the driver and making the changes needed to

display the chosen color. In move 27, she walks her partner through the steps to fix the problem.

1. B: I tell you what, we'll just try it right here, then we can move it off it.
2. A: No. Let's not. Omigosh. Not again. Oh, oh.
3. B: Why is it doing that. It's not letting me.
4. B: It doesn't have to be on any layers.
5. A: So add layers. Here. So go in once. Yeah. And now…
6. B: Oh! This is not good.
7. A: So go just pick another color.
8. B: Well it's not responding. I like this color that just kind of jumps out at ya?
9. A: OK fine.
10. B: Blue or Red?
11. A: Blue.
12. B: Blue, blue. Omigod . . .
13. A: OK here just try another color now.
14. B: There's something seriously wrong . . .
15. A: Try another color.
16. B: I'll have to try everything. We'll try it like . . .
17. A: Here try, oooh that purple is really pretty! Let's try it! Click. Click on there.
18. B: Woah. Huh. No we do grey, look.
19. A: No do it.
20. B: Oh.
21. A: OK just do that purple that was pretty.
22. B: No I really like this one.
23. A: No keep writing like that.
24. B: Have chosen . . . needs. . . . But one second. Dangit that's white.
25. A: Oh well.
26. B: There. Woah. And we're just leaving it, I know how…there. This is screwed up.
27. A: OK here, can I see the mouse for a sec 'cause I know how to do this. Change your color, now right like that, see now, now it's good. And I delete that, and then . . .

The transcripts above provide examples of how the girls talk about challenges, how they react as individuals, and the way that their interactions shape those responses. The transcripts also show how different kinds of interactions can lead them closer to or further from a solution.

DISCUSSION

In this era of increased computer access and declining enrollment in computer science, there is a great need for current research on how students think about, respond to, and solve computer problems. Efforts to understand and mediate the under-representation of women in the IT workforce have largely ignored the ways

that girls learn, interact with, and produce technology. In an attempt to fill this gap, the current findings provide some indication of how pairs of middle-school girls think about challenges while working on the computer as well as the strategies they use to address those challenges. They offer insight into how to promote the kind of problem solving that will increase the likelihood that girls will pursue and persist in information technology courses and careers.

How Challenges Were Described

Unlike most studies of problem solving in which tasks are generated for the purposes of research, this one was conducted in an after-school setting where students worked on an authentic, open-ended task. Thus, the types of challenges that girls faced are more typical of what they would encounter during "real world" interactions with the computer. We have chosen to use the word "challenge" because Schoenfeld (1985) states that the definition of "problem" is relative. In other words, if someone already has a schema for solving a task, then it is more of an exercise than a problem. The girls appeared to interpret the word "problem" in a similar way to Schoenfeld. The data also suggest that the extent to which girls reported facing any challenge depended on how the question was asked. They were much more likely to agree they got stuck in Flash than to acknowledge that they had a "problem," which has implications for how educators support students and how future research questions are framed.

Of those who did report a challenge, most were able to describe the type of challenge they faced and more than half said that it was solved. The challenges primarily involved aspects of the Flash software, which is consistent with how the questions were asked. The most common difficulties were in making images move, which required them to use the timeline aspect of the software correctly, a problem that is similar to those faced by others using digital media for movie or animation creation. Students also reported difficulties with using the Actionscript programming language within Flash to create interactivity. These types of accidental difficulties are common in programming. To solve, they require algorithmic thinking, which is one of the concepts of IT fluency (National Research Council, 1999). Specifically, girls in this study reported problems with using "go to" to create interactivity in their games. Concerns about the "go to" statement were raised decades ago in a classic article by Dijkstra (1968), who argued for removing it from all high-level programming languages.

It is not surprising that students were more likely to describe accidental difficulties (a function of limitations in the software) rather than essential difficulties. The program activities allowed students to work through some of the essential difficulties involved with creating software (such as overall design) by paper prototyping before translating their design into instructions understandable by a computer. In addition, the girls were asked directly about when they got stuck with Flash—not when they simply got stuck. It may also be a function of the girls'

interpretation of their difficulties in terms of the software rather than in terms of their approach to it. This question could only be answered with additional data. It would also be useful to collect data during the paper prototyping phase of creation in addition to the programming phase, for a more complete picture of the kinds of essential difficulties they faced. In addition, more data on essential difficulties is needed to determine whether students' strategies for solving problems vary depending on whether they are perceived as accidental or essential.

Notably, many students did not describe a problem because they did not consider problems that were solved to be real problems. This is similar to what Kafai (1995) found in her study of elementary school students—some students did not identify any problem and others could identify a problem but not a solution. As stated earlier, the capability to identify and deal with unexpected problems when they arise prepares students to become fluent with information technology (National Research Council, 1999). The data suggest that educators need to reinforce to students that problems are an inherent and potentially fun part of programming. In addition, educators should provide the guidelines for helping students identify the source of their problems.

Problem-Solving Strategies

The data suggest that the girls in this study were able to monitor and assess their attempts at problem solving. The most common strategies they described involved asking for help and working to figure it out own their own. By helping each other learn the debugging process, many of these pairs of middle-school girls employed the problem-solving steps that will prepare them for computer science courses and IT fluency. Debugging is one the capabilities of the National Research Council's (1999) list for IT fluency—manage problems in a faulty use of IT. These steps include identifying that a problem exists, isolating the source and cause of the problem, determining and performing the work necessary to solve the problem, and then testing the solution.

When working with a partner, the debugging process requires the negotiation of interpersonal differences. As Barron (2000) suggests, students must create representations of their problem as they attempt to solve it. The transcripts showed how the first step for collaborative problem solving is coming to a shared understanding that there is a problem and defining what it is, and that this can create difficulties in moving toward a solution (Roschelle & Teasley, 1995). The process of negotiating this shared understanding is shown in the transcripts we presented. The transcripts also provide a window into how students respond to challenges when creating a computer game with a peer and their dialogue illustrates how they construct meanings about how they should interact with the computer. Contrary to previous research (Hou et al., 2006) the girls did not try to complete the task as efficiently as possible. Instead, like the adults in the study by Beckwith et al. (2006), the girls were interested in exploring the features of the software.

What the pairs seemed particularly effective at was reminding each other to use resources or to compare the non-working instance to one that they already solved. The transcripts also show how some students embed playful interaction when they do not know how to do something. As Schoenfeld (1985) has reported, sometimes the "acknowledgment of mutual ignorance" serves as a pressure release valve which lightens the burden of the problem-solving task. Because the audiotapes were collected at only one point in time, it was not possible to get an accurate measure of how problem solving changed over time; however, the transcripts show the process through which different types of talk might support or undermine problem solving at the computer with a peer.

The girls' strategies are consistent with the instructional model that was used in the program. Similar to other programs (Edwards, 2002), the teachers worked to scaffold students' learning whenever possible, rather than to solve their problems for them, building on Vygotsky's (1978) theory that scaffolding can help students solve more complex problems than when they work alone. As Schoenfeld (1985) states: "It seems reasonable that involvement in cooperative problem solving—where one is forced to examine one's ideas when challenged by others and in turn to keep an eye out for possible mistakes that are made by one's collaborators—is an environment in which one could begin to develop the skills that, internalized, are the essence of good control" (p. 143). By control, Schoenfeld refers to "selecting and pursuing the right approaches, recovering from inappropriate choices, and in general monitoring and overseeing the entire problem-solving process." The current study suggests that working in pairs might allow for the development of effective control strategies for solving information technology problems.

Although most of the girls did describe a strategy to solve their problem, 29% of the entries did not include a strategy. For students to persist in computer science, they must understand that even for experts, working with computers requires developing strategies for problem solving. It is also important to know that problem solving when debugging software does not necessarily mean that there will always be clear steps to a solution. The problem-solving process often involves tinkering, which may involve making and testing hypotheses about the cause of the problem, or may involve more random exploration (Papert, 1993).

Motivating Girls to Persist in the Face of Challenges

As education moves toward emphasizing computer fluency (rather than literacy), students' ability to identify and respond to challenges may be more important than learning to use specific software programs. The current findings build on other studies that show which types of interactions support learning and problem solving. They also suggest that in some settings girls do persist in the face of challenges and provide information about how social processes can extend and inhibit their strategies.

The findings have implications for how teachers and other educators can create a learning environment in which girls develop a way of thinking and problem solving that will make it easier and more likely for them to pursue courses and careers in technology. As previous studies suggest, in order for girls to persist in computer science courses and careers, they need to be resilient in the face of challenges. In successful collaborations, conversational turns build on each other and the content moves the pair closer to solving the problem (Roschelle & Teasley, 1995; Schegloff, 1991). One example is that a student might start an idea while her partner finishes it. If conflicts occur, as they often do, the pair can attempt to work with the conflict by either finding a compromise, using persuasion to convince a partner of the preferred approach, or working around the disagreement by doing something different.

It is important to inform students of the range of legitimate techniques for addressing the inevitable challenges that they will face while working both alone and with others on the computer. The findings suggest that girls are using a range of techniques when faced with challenges while working with a partner on the computer. Although we cannot say which techniques are most effective with our current data, we can use these findings to identify specific practices. The next step would be to determine which were most effective for solving the problem, and which are most likely to prepare students to persist in computer science courses and careers.

In summary, the current study contributes to our understanding of how girls think about and solve challenges in pairs while engaged in the act of producing (not just using) software. By focusing on open-ended problems, the findings have relevance to efforts to teach girls programming, as well as to efforts to engage them in greater numbers to become producers of technology.

ACKNOWLEDGMENTS

We would like to thank all the participants, program leaders, teaching assistants, and community and school partners who made this program possible. We also recognize the work of Steve Bean, Shannon Campe, Cathy Tyner, and Gail Levine who helped to develop and implement the program.

REFERENCES

Armoni, M., Gal-Ezer, J., & Tirosh, D. (2005). Solving problems reductively. *Journal of Educational Computing Research, 32*, 113-129.

Auerbach, C. F., & Silverstein, L. B. (2003). *An introduction to coding and analyzing qualitative data*. New York: New York University Press.

Barker, L., & Cohoon, J. (2006) *Promising practices: Collaborative learning environments and pair programming.* National Center for Women and Information Technology (NCWIT). Retrieved April 7, 2006, from www.ncwit.org/pdf/COLLABORATIVE_ED_practice.pdf

Barron, B. (2000). Problem solving in video-based microworlds: Collaborative and individual outcomes of high-achieving sixth-grade students. *Journal of Educational Psychology, 92*, 391-398.

Beckwith, L., Kissinger, C., Burnett, M., Wiedenbeck, S., Lawrance, J., Blackwell, A., & Cook, C. (2006). Tinkering and gender in end-user programmer's debugging. *Computer Human Interaction, ACM Proceedings*.

Brooks, F. P., Jr. (1987). No silver bullet: Essence and accidents of software engineering. *Computer Magazine, 4*, 10-19.

Bruce, C., Buckingham, L., Hynd, J., McMahon, C., Roggenkamp, M., & Stoodley, I. (2004). Ways of experiencing the act of learning to program: A phenomenographic study of introductory programming students at university. *Journal of Information Technology Education, 3*, 143-160.

Committee on Equal Opportunities in Science and Engineering. (2004). *Broadening participation in America's science and engineering workforce*. Arlington, VA: National Science Foundation.

Cooper, J., & Weaver, K. D. (2003). *Gender and computers: Understanding the digital divide*. Mahwah, NJ: Erlbaum.

Creamer, E. G., Burger, C. J., & Meszaros, P. S. (2004). Characteristics of high school and college women interested in information technology. *Journal of Women and Minorities in Science and Engineering, 10*, 67-78.

Denner, J., Werner, L., Bean, S., & Campe, S. (2005). The Girls Creating Games program: Strategies for engaging middle school girls in information technology. *Frontiers: A Journal of Women Studies. Special issue on gender and IT, 26*(1) 90-98.

Dijkstra, E. (1968). Go to statement considered harmful. *Communications of the ACM* (Vol. 11, pp. 147-148. Retrieved March 28, 2006, from http://www.acm.org/classics/oct95/

Edwards, L. D. (2002). Learning by design: Environments that support girls' learning with technology. In N. Yelland & A. Rubin (Eds.), *Ghosts in the machine: Women's voices in research with technology* (pp. 119-137). New York: Lang.

Goos, M., Galbraith, P, & Renshaw, P. (2002). Socially mediated metacognition: Creating collaborative zones of proximal development in small group problem solving. *Educational Studies in Mathematics, 49,* 193-223.

Harel, I. (1991). *Children designers: Interdisciplinary constructions for learning and knowing mathematics in a computer-rich school*. Norwood, NJ: Ablex Publishing Corporation.

Hou, W., Kaur, M., Komlodi, A., Lutters, W. G., Boot, L., Cotten, S. R., Morrell, C., Ozok, A. A., & Tufekci, Z. (2006). "Girls don't waste time": Pre-adolescent attitudes toward ICT. *Computer Human Interaction, ACM Proceedings*.

Inkpen, K., Booth, K. S., Klawe, M., & Upitis, R. (1995). Playing together beats playing apart, especially for girls. *Proceedings of Computer Supported Collaborative Learning (CSCL)* (pp. 177-181). Hillsdale, NJ: Lawrence Erlbaum Associates.

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

Littleton, K., & Light, P. (Eds.). (1999). *Learning with computers: Analyzing productive interaction*. London: Routledge.

Margolis, J., & Fisher, A. (2002). *Unlocking the clubhouse: Women in computing*. Cambridge, MA: MIT Press.

McDowell, C., Werner, L., Bullock, H., & Fernald, J. (2003). The impact of pair programming on student performance, perception and persistence. Proceedings of the *25th International Conference on Software Engineering*, pp. 602-607.

McKenna, P. (2004). Gender and black boxes in the programming curriculum [Electronic version]. *ACM Journal on Educational Resources in Computing (JERIC), 4*(1).

McKenna, P. (2001). Programmers: Concrete women and abstract men? *Journal of Computer Assisted Learning, 17*, 386-395.

Miles, M. B., & Huberman, A. M. (1994). *Qualitative data analysis: A sourcebook of new methods* (2nd ed.). Thousand Oaks, CA: Sage.

National Research Council. (1999). *Being fluent with information technology*. Washington, DC: National Academy Press.

National Science Foundation, Division of Science Resources Statistics. (2006). Special tabulations of U.S. Department of Education, National Center for Education Statistics, Integrated Postsecondary Education Data System, Completions Survey, 1985-2004. Retrieved July 28, 2006, from http://www.nsf.gov/statistics/wmpd/figc-2.htm

Papert, S. (1993). *Mindstorms: Children, computers, and powerful ideas*. Cambridge, MA: Perseus Publishing.

Parnafes, O., & DiSessa, A. (2004). Relations between types of reasoning and computational representations. *International Journal of Computers for Mathematical Learning, 9,* 251-280.

Perret-Clermont, A., Perret, J., & Bell, N. (1991). The social construction of meaning and cognitive activity in elementary school children. In L. B. Resnick, J. M. Levine, & S. D. Teasley (Eds.), *Perspectives on socially shared cognition* (pp. 41-62). Washington, DC: American Psychological Association.

Roschelle, J., & Teasley, S. (1995). The construction of shared knowledge in collaborative problem solving. In C. E. O'Malley (Ed.), *Computer supported collaborative learning*. Heidelburg.

Schegloff, E. A. (1991). Conversation analysis and socially shared cognition. In L. Resnick, J. M. Levine, & S. D. Teasley (Eds.), *Perspectives on socially shared cognition* (pp. 150-170). Washington, DC: American Psychological Association.

Schoenfeld, A. (1985). *Mathematical problem solving.* Orlando, FL: Academic Press.

Tillberg, H. K., & Cohoon, J. M. (2005). Attracting women to the CS major. *Frontiers: A Journal of Women Studies, 26*, 126-140.

Turkle, S., & Papert, S. (1992). Epistemological pluralism and the revaluation of the concrete. In I. Harel & S. Papert (Eds.), *Constructionism* (pp. 161-191). Norwood, NJ: Ablex.

Turner, E., & Denner, J. (under review). Girls' attributions for success or failure regarding challenges on the computer.

U.S. Department of Labor. (2005). *Women in the labor force: A databook. Employed persons by detailed occupation and sex, 2004.* Bureau of Labor Statistics. Retrieved August 7, 2006, from http://www.bls.gov/cps/wlf-table11-2005.pdf

von Hellens, L., Nielsen, S. H., & Beekhyzen, J. (2004). An exploration of dualisms in female perceptions of IT work. *Journal of Information Technology Education, 3,* 103-116.

Vygotsky, L. (1978). *Mind in society: The development of higher psychological processes*. Cambridge, MA: Harvard University Press.

Werner, L., Hanks, B., McDowell. C., Bullock, H., & Fernald, J. (2005).Want to increase retention of your female students? *Computing Research News, 17*, 2.

Williams, L. L., Kessler, R. R., Cunningham, W., & Jeffries, R. (2000). Strengthening the case for pair programming. *IEEE Software, 17*(4), 19-25.

Yates, S. J., & Littleton, K. (2001). Understanding computer game cultures: A situated approach. In E. Green & A. Adam (Eds.), *Virtual gender: Technology, consumption, and identity* (pp. 103-123). London: Routledge.

Direct reprint requests to:

Dr. Jill Denner
Education, Training, Research Associates
4 Carbonero Way
Scotts Valley, CA 95066
e-mail: jilld@etr.org