4

Learning Design by Making Games

Children's Development of Design Strategies in the Creation of a Complex Computational Artifact

Yasmin B. Kafai

I made a game. It started out very slowly at first. It is very hard to put together your own game. You may think it is easy to do because of all the video games people play. They look so simple but try making your own game and it's a totally different story! Well, I started out with very high expectations thinking that I could make a great game in very short time. It turned out that I'm still not done with it even after about 4 or 5 months. (Rosemary, 10 years, toward the end of the project)

I really expected even though the teacher told me that it would take months and months and months to finish the game, I really did expect to do it, like start it, in one week and finish it up the next week. . . . I just went like: Oh, this will be easy. All it will be, is a little bit of this, a little bit of that, and a little research, and I'll be finished. But it didn't turn out that way because I had to spend a lot of days on research and programming. There were tons of problems, like one time my turtle was messed up. Plus I had to make all the graphics and everything. So it had problems, but it has been fun. (Jeremy, 11 years, at the end of the project)

As these quotes of two young game designers illustrate, there is something to be learned from making games that goes beyond learning programming and specific subject matter. The students' descriptions reflect their early expectations about making games, which were probably influenced by their experiences with regular classroom work and home assignments. Conventional school assignments rarely give students the opportunity to spend 6 months on a complex project such as making a game. Hence, most students have little experience in design (i.e., planning, problem solving, researching, dealing with time constraints, modifying expectations, and bringing everything together into one project). Previous research even suggests that young students may not be able to accomplish such projects because children have limited abilities in planning and dealing with

complex tasks (e.g., Brown & DeLoache, 1978; Friedman, Scholnick, & Cocking, 1987).

Recent efforts in the educational research community, however, have stressed the importance of self-directed, personally meaningful, and cognitively complex projects for students' learning success (Blumenfeld et al., 1991; Collins, Brown & Newman, 1990; Harel & Papert, 1990). Design activities are one example of such project-based activities in which students are engaged in designing complex, interactive pieces of software to be used by other students for learning about a particular subject area (Harel, 1991). A variety of design activities have been proposed that range from creating interactive presentations to instructional software, simulations, and educational games (Carver, 1991; Guzdial, 1993; Kafai, 1993; Lehrer, 1991). A prominent feature of these activities is that students learn about design by managing the projects while, through design, they learn about academic subjects such as programming, history, mathematics, and physics.

One objective, then, of the research presented here was to examine the ways in which children approach a complex design task and identify the kinds of obstacles they face and overcome. A class of fourth-grade students became game designers and worked daily toward the goal of making the learning of fractions fun and easy for younger students. In this process they discussed issues related to fractions, teaching, programming, and games, meeting once a month with their prospective users. Students' design progress was observed and analyzed from the beginning of the product to its completion 6 months later. The following sections provide the theoretical background and describe in detail the nature of the design task and the methods used for gathering and analyzing the data. Selected results from analyzing the whole class of game designers and individual case studies have been included to provide support for reoccurring themes and trends.

REVIEW OF RESEARCH

In this research, design was an object of study as well as a context for a study of learning. Learning through design (Harel, 1988, 1991) is based on a constructionist theory that sees learners as builders of their own knowledge—a process that happens best when students are building external and shareable artifacts such as computer programs, machines, or games (Papert, 1980; 1993). When learners are asked to design something for the use of others, their learning becomes instrumental to a larger intellectual and social goal. In this context, students are in a continuous dialogue with their own ideas and with the ideas of intended users and co-designers. Student-designers assume control of their learning by asking questions, gathering information, and putting all this to work in creating an educational game.

Although learning and designing are closely intertwined in this process, it is important to point out a distinction between learning through design or through professional design. The learning through design is not exclusively represented in the final product, but also in the process of doing it. "Professional design" focuses on the product as the essential outcome, whereas "learning design" focuses on the students' process as the primary cause for learning (which is only partially reflected in the product). In learning through and about design, the process is more important than the product. This means that even though students may not achieve a well-rounded final product, learning can take place because of the involvement over time. Applied to the problem solving and planning abilities of younger children, this means that the context of design puts them in the mode of thinking like planners, problem solvers, and designers—all together—over and over again.

My particular focus is on how students as designers integrate planning and problem solving while building an artifact. Students act as problem solvers and planners in multiple ways, but in the design process they are the ones who identify the problems and plan how to solve them. Schauble (1991) spoke in this context of the student-designer's task to impose and manage the multiple constraints of complex situations in design. The extended time frame of the research project provides the opportunity for students to be engaged with different facets of the design process. For example, the designer begins by finding a problem, then discovers parts of the solution, tries to make sense out of it, considers how to reframe the situation, and continues with problem solving. In this process, the designer constructs a particular, personal relationship to the artifact that undergoes changes as the process continues. This process seems to stop when an artifact has been created, but, actually, it never ends because existing design solutions are used and reused in new design situations (Schön, 1983).

The traditions of school and academia have favored formal and abstract approaches to problem solving and design. Previous research on software design identified different approaches of handling complex design and used bipolar descriptions, such as "top-down" versus "bottom-up" (Jeffries, Turner, Polson, & Atwood, 1981; Newell & Simon, 1972) or "planning" versus "bricolage" (Turkle & Papert, 1991).

> The bricoleur resembles the painter who stands back between the brush strokes, looks at the canvas, and only after this contemplation, decides what to do next. For planners, mistakes are missteps; for bricoleurs they are the essence of a navigation by mid-course corrections. For planners a program is an instrument of premeditated control; bricoleurs have goals, but set out to realize them in the spirit of a collaborative venture with the machine. For planners, getting a program to work is like "saying one's piece"; for bricoleurs it is more like a conversation than a monologue. In cooking, this would be the style of those who do not follow recipes, but instead make a series of decisions according to taste. While hierarchy and abstraction are

> valued by the programmers' planner's aesthetic, bricoleur programmers prefer negotiation and rearrangement of their materials. (Turkle & Papert, 1991, p. 136)

The view that has been called planning, or top-down, tends to see the process of problem solving as one of breaking down a problem into more meaningful subproblems (Guindon, Krasner, & Curtis, 1987; Jeffries et al., 1981). Here, the designer maps out the context, content, and structure of a design at the beginning. The opposing view, called bricolage or bottom-up, describes problem solving as a conversation with the situation, in which the design of the game emerges in the process of implementing it. These two views suggest that students may approach the design task from different ends, choose to emphasize different aspects of the design, and think about it in different ways depending on their personal preferences. My intention in the game design project was to offer a design activity that provided multiple avenues for students to approach a complex problem, build a game around the idea of fractions, and find personal solutions. As Simon (1969) pointed out, a unique feature of design problems is that they do not have a single right solution; there are always alternatives.

Game design proposes a complex project that engages students over a long period of time. The complexity of the task becomes clearer if one considers that designing software involves more than the mere production of code: structuring the control flow of the program, maintaining the connection between different procedures and pages, designing the graphics, and providing the interface for the prospective user. In addition, students must consider different ideas for fraction representations, how the representations can be implemented in Logo, and pedagogical concerns, among many other issues. The actual challenge for the game designers is to combine and integrate the instructional content and game context. I hypothesize that these activities will place children in a situation that requires them to design, plan, reflect, evaluate, and modify their programs on a constant basis. The expected outcome is that these diverse activities will allow students to acquire more sophisticated programming skills in conjunction with other subject matter. Project management is one of the central issues: How do students deal with the complexities of a design task? What kind of strategies do they use or develop, if any, in the course of a long-term project?

The answer to these questions is of particular relevance because previous research on the development of children's abilities to deal with complex tasks suggests that children know how to plan, but are limited by their knowledge of the situation (Bjorklund, 1991). In order to investigate these issues, I chose a familiar, yet complex task: I investigated children's design approaches to programming a game. Unlike most research from the problem-solving domain, the students were given neither a well-defined nor an ill-defined problem that could be accomplished in a short amount of time (Dreher & Oerter, 1987; Kreitler & Kreitler, 1987; Pea & Hawkins, 1987). Instead, they were given a space/context in which they could construct their own problem—their own tasks, goals, activities, and rules that guided their game design activities. This situational approach created a microworld for children; students were autonomous in following their own paths of learning and activities. In doing so, however, they were also building microworlds of their own, game worlds for others to play and learn with. The goal of this investigation, then, was to analyze whether particular design approaches emerged over time, as has been described in the literature, and to what extent these approaches were consistent.

RESEARCH CONTEXT

In many ways, the project recreated the atmosphere of a professional design studio. A class of 16 fourth-grade students were engaged during an extended period of time (6 months) in the design and production of educational games that teach fractions. The students worked for 1 hour a day on their projects in class. In general, students first spent 5 minutes writing their plans and ideas in notebooks before they went to work on their computers for 45 minutes. As students worked on their games, they were allowed to walk around the room to see and discuss each others' projects. Students then returned to their own classrooms and wrote again about their experiences.

These daily sessions were complemented by several focus group sessions in which students discussed issues related to games, their projects, their ideas, or difficulties about fractions and how to represent them. The group sessions were designed for brainstorming and discussing issues of interest for all students. Every month younger students visited the classroom to evaluate and discuss the older students' game projects. Because the purpose of this project was to create a finished product, students worked concurrently in their art classes on cover designs for the packaging of their games and in their language arts classes on advertising for their products and the documentation for them. During the 6 months of the project, the students spent 92 hours programming and 20 hours on related activities and school materials.

The game design project took place at Project Headlight, located at an inner-city, public elementary school in one of Boston's low socioeconomic neighborhoods. The goal of Project Headlight, which started in 1985, is to explore the use of computers in classrooms as they might exist in the near future. Project Headlight operates as a school-within-a-school; the other two thirds of the school have traditional classroom and computer arrangements. One special feature of this elementary school is its open-architecture structure, which was not fully utilized before the project started. In two open areas, four clusters of computers are arranged, each with 16 computers in a circle, screens facing out. The clusters of computers are called "pods." As students come and go to their classrooms, they can walk among the computer pods and see other students' projects. Another

feature of Project Headlight is that students are rarely using educational software; instead, they are working with LogoWriter to create their own software.

Approximately 250 students participated in this project, from first through fifth grade, including advanced, regular, bilingual, and special education classes. The participants were of diverse ethnic backgrounds, half boys and half girls. Most of the students had only joined the school in the fourth and fifth grades; hence, their beginning programming experience was not very extensive.

METHODS OF DATA COLLECTION AND ANALYSES

The game development of 16 student designers was observed every day over the course of 6 months. A combination of qualitative methods was used to document the students' ideas, thoughts, and progress in game development. Pre- and post-interviews were conducted to gather information on students' knowledge of programming, fractions, video games, and planning experiences. During the project, the students participated in research activities by keeping notebook entries and saving log files to provide additional information for the researcher. The data were examined in the following ways: (a) The students' projections of their game designs in the notebook and interviews were compared with their actual program implementations in Logo, (b) the development of the game theme and features over time; and (c) the development of the program code in regard to the use of modularization, that is, making procedures, which is described in the research literature as one important programming strategy (Carver, 1987; Pea & Kurland, 1984). Particular attention was paid to the beginning phase when students began formulating their first game ideas including expectations about upcoming difficulties, and to the period of first transition, 3 to 4 weeks into the study, when students had accomplished part of their implementations and had already handled design and programming problems.

RESULTS

During the 92 days of thinking, designing, programming, modifying, and playing their computer games, the students carried out many activities and touched on many issues.[1] The extended time frame allowed for a closer investigation of the

[1]Other parts of my investigation addressed the conceptual development of students' dealings with fractions, their understanding of the programming language Logo, their mastery of particular programming concepts, and their development of programming strategies. The results from these different investigations provided evidence that the students in the project were able to handle this complex and challenging task in a significant manner when compared to students instructed by other pedagogical means. I have reported elsewhere more extensively on this aspect of the students' learning experience (Kafai, 1993, 1995).

project trends and development. As the students started implementing their game ideas, they put their programming skills to the test and learned about the feasibility of their game ideas as well as different aspects of the design process. It is worthwhile to investigate transitions or shifts and whether they occurred throughout the class or were only experienced by individual students. Over time, two kinds of developments could be observed: One kind, of incremental nature, was reflected in the daily growth of the game designers' programs in terms of Logo code; the other kind represented shifts in the nature of the games as well as the approach chosen by the designers. Looking at the development of the students' games as a whole from beginning to end, seeing how students outlined features of their games and also set out to implement them, provides a first view on distinct design approaches.

In the following sections, design styles adopted by students and different phases of the design process are analyzed in more detail. Particular attention is given to the way students deal with the complex nature of game design by framing the task and imposing their own structures. In addition, students' development of programming strategies is examined using the cases of two young game designers, Amy and Albert.

Design Styles

One objective of this analysis was to investigate whether bipolar descriptions, such as "top-down" or "planner" versus "bottom-up" or "bricoleur," correctly assess children's approaches to a complex task such as designing a game. We could expect that "planners" would carefully lay out what they want to do in advance. We would expect "bricoleurs" to develop their games as they go along. To clarify this point, I chose the games of Barney and Gaby as two extreme examples of game development (see Fig. 4.1).

In her first interview, Gaby described the different parts that she would implement for her spider web game. She had already segmented her descriptions of the different game components into the blocks, the web, and the fractions. When Gaby started implementing her game, she focused first on the spider web, then the blocks on the screen, then colored blocks, before she started working on the fraction questions, one after the other. In her programming, each unit was assigned a procedure. For example, the five colored blocks all have the name BLOCK, but each color has a different prefix (LBLOCK, MBLOCK, and so on). One could say that from day one Gaby laid out the implementation of her game and followed through, even though she did not implement some aspects of her initial plans. In that respect, Gaby falls most clearly into the category of the planner. In the second part of the project (which is not represented in the diagram), Gaby wrote the introduction and directions, then programmed the spider's movements as it followed the fly.

In contrast, Barney described the idea of an adventure story in the interview and his notebook designs. He initially provided only one screen display: that of a

Gaby : I am going to make a **spider web**, right. I need to make someblocksup here, but I don't know how to make them . . . You see a square, like here, and it has a little space so the fly can get over here and then, you see, the spider, the web is going to be right over here and if it gets closer to the web, the spider is sitting there and it is going it eat. But if not, this is going to be the safe place and if it gets the right answer it's going to get a little bit closer and closer until it gets to the safe place and then like a smiling face.

Barney : Yeah I am just going to do like . . . you have to go on an adventure or something like this and people have to ask you all these fractions questions and if you get them right, you go on and if you get them wrong you . . . see it's going to be hard, if you get it wrong,once, then you are out . . . The questions will be pretty easy but it's just if you get it wrong once, you are out.

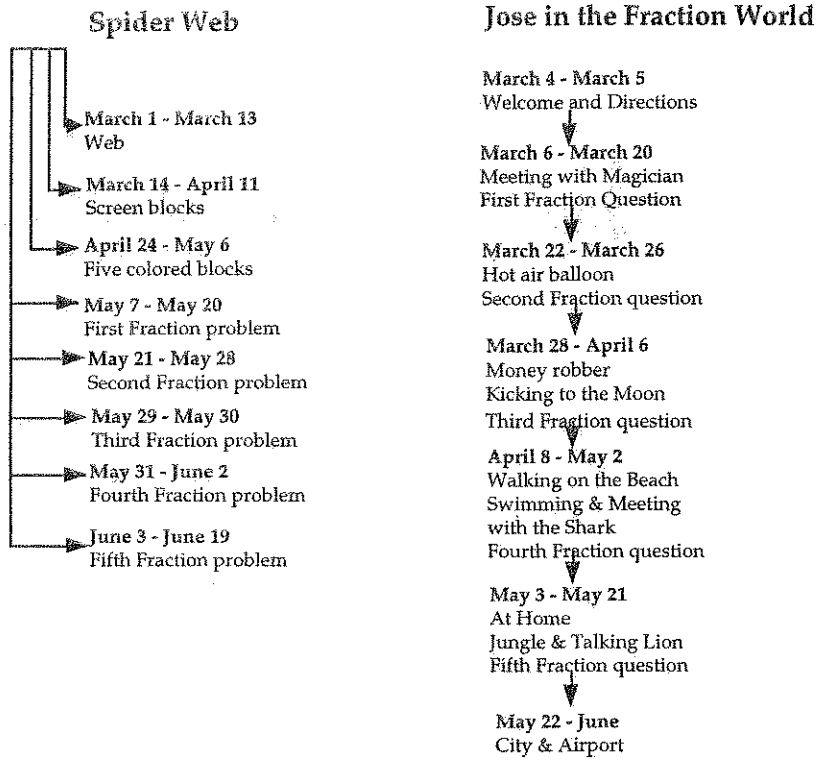


FIG. 4.1. Comparison between the design approaches of Gaby and Barney from March to June.

fish speaking. As Barney set out to implement his game, he started with a welcome screen, introduction, and directions. From then on, he developed scene after scene. Even though most of his scene units, such as the magician or the money robber, are centered around a fraction quiz, they are composed of a series of connected animations that lead to the fraction question and then away from it

to the next adventure. His game is a series of connected adventures created one after the other. In that sense, Barney is very much like a bricoleur.

I would categorize both Barney and Gaby as extreme cases for each approach. Other students (such as Darvin, Albert, Miriam, Trevor, Sina, Shanice, Amy, and Jero) also exhibited planning behavior, but to a lesser extent. Their planning seems to be more localized and deals with particular sequences. In particular, I refer the reader to Amy's blueprint for the instructional quiz that she clearly planned in advance (see the section, "Developing Programming Strategies"), or to Miriam's design of the ski slope, which was implemented to the smallest detail from her notebook design. Other students could also be described as working without a preformed plan. Sid, Juan, Tyree, and Rosy were engaged in a continuous dialogue with their game as they designed various scenes or fraction questions.

A combination of both planning and bricolage seems to be a more accurate way of describing the typical student's game design process. The results of my analyses indicate that most students chose to walk a middle line between bricolage and planning, and their approaches changed over time. Only 4 out of 16 students could be clearly classified as either planners or bricoleurs. At specific stages in the design process (for example, when programming the first fraction question), students tinkered around with different aspects: They designed different shapes and dialogues before deciding on a solution. In later stages, they switched over to reusing, in a purposeful way, program segments that they had written before (such as for the instructional dialogue). The clear distinction between planners and bricoleurs seems to fall apart during observations over a long period of time and over the construction of a complex product. This result speaks against bipolar descriptions. There is further evidence in the game design process that students did not adopt one particular strategy but rather negotiated the situation by imposing and coordinating multiple constraints with their personal interests in game design (Schauble, 1991). To provide better support for this statement, it is important to examine the beginning phase and the way the game design task was construed by students over time.

Defining Constraints of the Design Task

In the first days of the project, students formulated their game ideas and began to implement them. In the game programming process, children exhibited different design approaches. Many changes (game title, new introduction, main game idea) in the students' game design occurred either in the first days of the project or after 3 to 4 weeks into the project.

Game Design Approaches. My observations of the students' work in this beginning phase captured a variety of activities, but two different approaches stood out. In the beginning phase, some students did not start working and implementing their game ideas right away; others started with one idea, aban-

doned it quickly, and moved on to a new one. Those who did not start programming immediately played with or explored areas seemingly unrelated to their assignment. For example, both Albert and Shanice experimented with some new Logo commands to which they had been introduced a few days earlier. Barney refused to start the project at all because, as he told me, he did not have any game ideas for fractions. Shanice, Sina, and Rosy wrote for several days in their notebooks, "I am going to start my fractions project."

A different approach was taken by several other students. They started with one idea, worked on it for a few days, and then moved on to a new game idea. In this category, we find Amy with her "fraction thing" that she later turned into a map design; Miriam with her "Mr. Fraction" that turned into the skiing game; Jero's magician that turned into a map with different levels; and Tyree's fraction screen that turned into a space game. In this approach, students outlined a few features of their first games in their notebooks or programmed a few lines before they came up with a new idea. This change in the beginning phase was rather abrupt, as most students did not indicate a lack of confidence in their first choices. Instead, their games seemed to take a new turn from one day to the next.

The students used both approaches to the design task to enter the new domain of game design. I do not interpret either approach as unproductive or inactive. Either one provided time for students to grapple with the issue of bringing together the two domains of fractions and game in one design. In Harel's (1991) instructional software design project (in which students designed educational software and not educational games), Debbie actually took several days before she started with the design of her instructional screens. In very much the same way that professional designers would start out, the students in the game design project either tried alternative ideas and designs or decided to postpone creating their game until they had formulated a more coherent game idea and knew what they wanted to do. A further look at the content of the students' first designs repeats this impression. Some students, such as Barney, Tyree, Shaun, Juan, and Rosy, started with their welcoming screen or introductions (the first screens usually seen by the player when starting the game). Other students immediately started working on their first game scene, designing shapes or the environment, and did not implement the welcoming screen and the introduction until midway through or toward the end of the project. The diversity of project themes, working styles, and entry paths in design activities has been described and discussed by Resnick (1991).

Definitions of the Game Design Task. Very early in the project students developed game ideas that integrated fractions in different ways. Two distinct game formats were adopted by most students: extrinsic or intrinsic integration of fractions into the game (called Game Worlds or Fraction Worlds, respectively). The former is exemplified most simply in games or software where the player has to answer a question in order to proceed in the game. In contrast, intrinsic

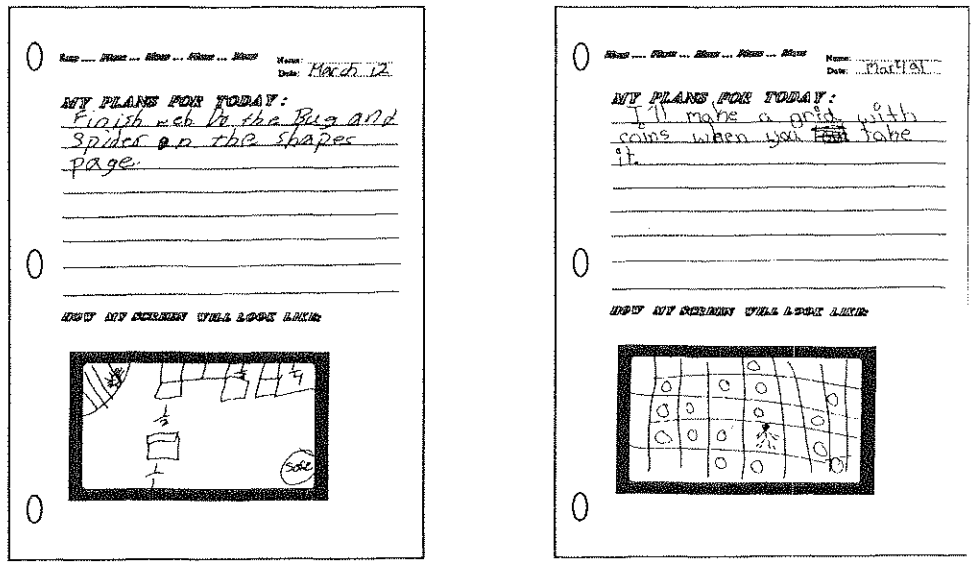


FIG. 4.2a. Examples of game worlds—Gaby's and Trevor's game designs. Gaby's game describes a spider web in which the player moves around as a fly, away from the spider, and turns on fraction blocks where questions are posed. Trevor shows the coin grid and a figure that represents the player.

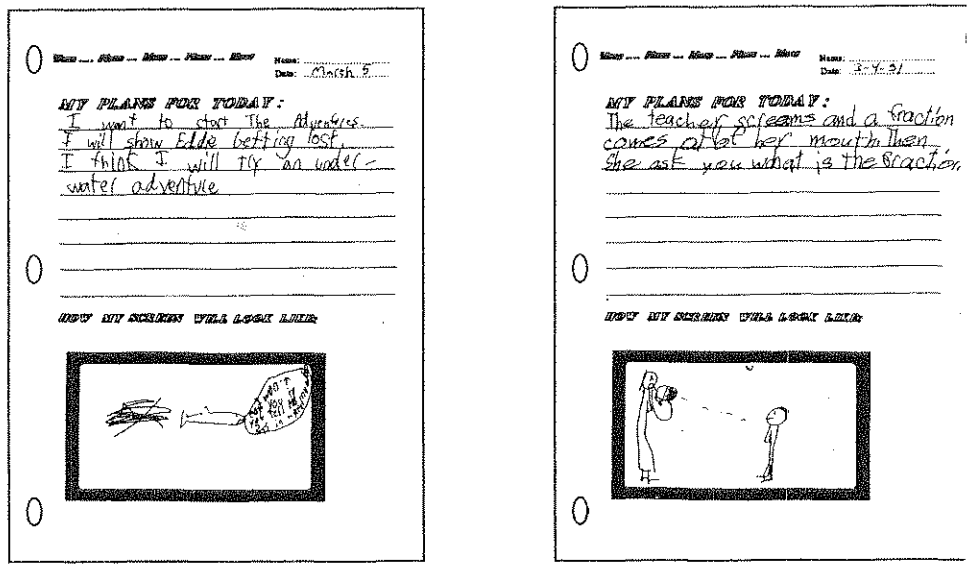


FIG. 4.2b. Examples of game worlds—Barney and Sina's game designs. Barney draws the underwater scene. Sina shows the two protagonists of the game, the teacher and the player.

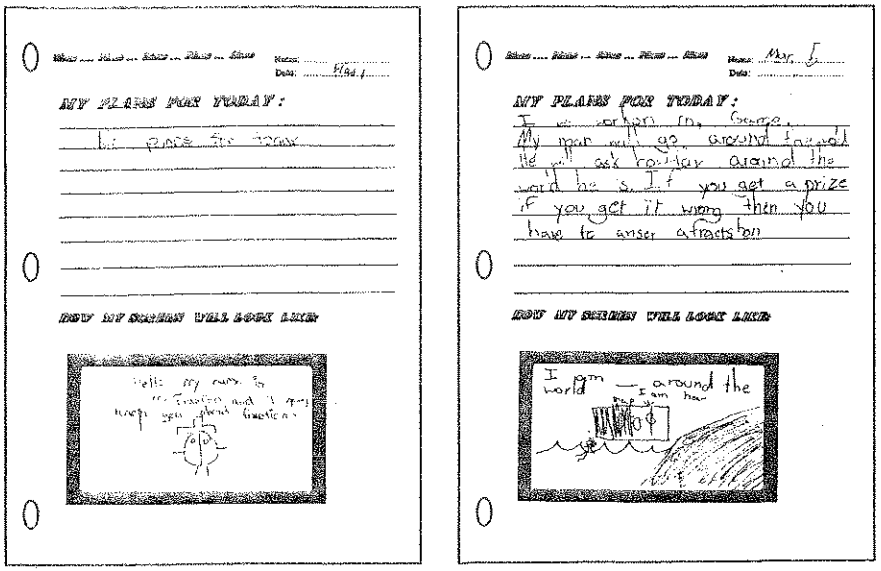


FIG. 4.3. Miriam and Rosy's fraction worlds. Miriam's text to the graphic of Mr. Fractions says: *Hello, my name is Mr. Fraction and I am going to teach you about fractions.* Rosy described her first game idea as: *I will work on my Game. My man will go around the world. He will ask how far around the world he is. If you get a prize if you get it wrong then you have to answer a fraction.*

integration is exemplified in a game where the designer integrates the subject matter with the game idea.

In Game Worlds, students developed different worlds in which the player interacts with fractions. What pertains to all the game worlds is that the rules are bound to the fraction questions, and the player's success depends on figuring out the correct answer in order to continue or finish the game. Many students' games make reference to those commercially available such as Nintendo. For example, Jero's different warp zones are reminiscent of Mario Brothers' tunnel system. Oscar made explicit reference to the Pacman game, but chose eating fractions instead of points. Sid's basketball game is available in various video game versions. Gaby took an educational game she had used in her previous school, the spider web, and adapted it to fractions. Some of the students used an educational situation when choosing a game theme. For example, Sina's outline of her teacher game shows a teacher asking, or rather, "screaming" a fraction. Gloria came up with a similar idea (see Figs. 4.2a and 4.2b).

Fraction Worlds, by contrast, described games in which the idea of fractions is inherent in the game concept. There were a few games in which the idea of fractions was central, such as Amy's "fraction thing," Miriam's "Mr. Fraction," or Rosy's "world map" (see Fig. 4.3).

Games with intrinsic integration could also be considered a form of microworlds in which fractions are situated. Papert has defined the microworld as "a computer-based interactive learning environment in which the prerequisites are built into the system and where learners can become the active, constructing architects of their own learning" (1980, p. 122). Initially, the term *microworld* was reserved for the creation of software tools that allow learners to explore their knowledge. Here the students became the designers of software tools for other children. They used Logo to design their games as microworlds for others to play and learn about fractions.

One issue of immediate concern is why so few students considered the design of fraction worlds. I have three possible explanations.

One explanation addresses the very nature of educational games: They lie somewhere between learning and playing. They are a specific breed because the rules necessary to play the educational games demand the use of valuable educational skills. The students obviously felt the tension between the demands of playing and those of learning. Darvin saw the two poles clearly, as he expressed in his notebook entries on March 1: "I want to make it [the game] as fun and educational as possible but especially fun."

My second explanation comes from observations of educational games designed by researchers and educators. For a long time, teachers have used educational games on and off the computer to keep students motivated (Avedon & Sutton-Smith, 1966; Block & King, 1987; Lepper & Malone, 1987). Teacher-designed games have shared an aspect common to most students' games: The game idea, content, or form was external to the game (Bride & Lamb, 1991; Fennell, Houser, McPartland, & Parker, 1984; Priester, 1984). In all cases, the game idea and format could also be used for other subject areas. By this I mean that the content—fractions—did not matter. The game context was used to keep the students' attention alive and to keep them motivated in accomplishing the tasks. Many of the student-designed games shared this quality.

Yet another explanation of why so few game designers made the ideas of fractions central to their games and why some of the good ideas were quickly abandoned is that fraction worlds or microworlds are harder to invent. A situation where answering fraction problems allows you to advance is conceptually simpler: The fraction challenges are fully factored out of the game context. This is a simpler, logical structure. Microworlds may also be harder to implement after one has the basic idea. The intertwining of fractions and the game context creates interactions between the two that may pose complex programming challenges. One can conclude that students defined the game design task in this fashion because it facilitated their entry into the design task, even though it happened to the detriment of fractions. They limited the range of options to consider. As in commercially made games, the greater number chose the easiest extrinsic integration by stopping the action at key places to ask the player a question. It is a strength and a weakness of the extrinsic integration that domains of knowledge become almost interchangeable. It is a strength because the integration is rela-

tively easy: When answering a question correctly is what allows the next move in a game, the question can be on any topic. This is also a weakness, however, because it causes the designer to lose the incentive to think deeply about the particular piece of knowledge.

These observations of approach and content in the game design process indicate that there was no uniform way in which students handled the early days of creating their games. These observations, furthermore, reinforce the impression that there is no one "right way" to start a design task, and that many of the students' choices in approach and content were related to their personal preferences. The entrance point of forging a relationship with the task was not necessarily the same for everyone. A planner might consider it a bad strategy to start the game with the most important scene instead of designing the first screen. Yet for a bricoleur, this might be the only possible and reasonable way.

Negotiating Constraints of the Design Task

A further observation points out changes in students' own perceptions and expectations in the course of the game design process. In the beginning, students had varied expectations about upcoming difficulties. Some students were not concerned, whereas others considered the feasibility of their game ideas in the light of implementation difficulties.

Expected Difficulties. In interviews during the first days of the project (March 5), neither Amy nor Sid were concerned about any problems:

Interviewer: What do you think is the biggest problem right now?
Sid: I don't know. Actually I don't have a problem right now.
Interviewer: What do you think could become a problem?
Sid: Nothing.

Interviewer: Let me ask you something. What do you think right now will be your biggest problem with this project?
Amy: I have no idea right now [waves her hand], beats me.

In contrast, other students thought of some problems in relation to their game ideas or upcoming implementations:

Interviewer: That sounds like a neat idea. The pacman who is eating fractions?
Oscar: That's what I wanted to do.
Interviewer: So work on that.
Oscar: It's a little bit too hard.
Interviewer: So can you think of a simpler version?
Oscar: I am trying . . . but I am going to think of something different.

Interviewer: If you think about your project now, what do you think might be the biggest challenge for you?
Albert: Ahm, . . . right now, . . . I think the biggest challenge that I am gonna make here is making the robbers come at you or something. When you open a door, something, the guys come at you and making all that happen.
Interviewer: So, it has to do with Logo programming?
Albert: [nods yes] I can also have, once you type in you want to open the door, it shows you and the guy will step out with the knife and the guy will go like this [he indicates a cutting movement with his arms and he is laughing].
Interviewer: Are there any challenges related to fractions for you?
Albert: I don't think I will have any problems, but I might come across some of them that might be kind of difficult.

Oscar was concerned about the difficulties in his idea of adapting a Pacman game to eat fractions instead of points. Albert thinks about the difficulties related to programming future animations. In the following interview, Gaby discusses the number of different parts in her game, Spider Web, that she will have to implement:

Interviewer: OK, so you are going to make a spider web; there is going to be a safe place and some fractions? These are many things you are thinking about for your game. So what do you want to start with now?
Gaby: Well, I am starting a web because I am thinking it is going to be a little bit hard plus making the spider, and the fly, and the blocks and the fractions, and things and the smiling face, would be hard for me, you know, to get it all started. It will take me more than . . . it wouldn't take me less than a week. But it would take me more than a week trying to plan all these things. We have a short amount of time.

Reduced Expectations. Students' reduced their expectations of how much they could accomplish in the project. One example of this is the number of fraction problems students intended to design. In the first days, Amy projected doing 21 fraction questions (taken from the parts of her map). Jero planned to do 20 questions on nine different levels, and Shaun was thinking about 100 fraction problems. The students' projections were probably influenced by worksheets, textbooks, or educational software that contain a large number of problems. These models might have influenced the students in their initial projections of how many questions they could or should incorporate in their games. This also replicates the experience of Debbie in the instructional software design project (Harel, 1988, p. 192), who wanted to show *all* the fractions in the world. In the

course of the project, all of the students reduced their expectations, and the average number of instructional situations was approximately five.

The variations in expectations might be related to the students' lack of experience in working on such a complex and time-consuming project. The students' own evaluations written at the end of the project are a good indicator of this. For example, Gaby wrote: "I disliked it because sometimes I kept working on the same thing over and over again. I liked it because I finally finished it and I made everything move." Amy wrote: "I disliked working on the fractions software project because sometimes it got boring because you had to work on it day after day, week after week, month after month, but at the end I felt like I had accomplished something." In the past, students usually had worked 3 to 4 weeks on one assignment and then moved on to the next one. Their planning skills and estimations regarding the game design were based on their previous experiences. One should not forget that these issues are hard to express as students begin to formulate what their games will be about. Even if we expected students from the beginning to map out their plans and ideas for implementations, it would be unrealistic to think that they could take into consideration all of the aspects involved in planning and designing a game. Related to this, few students likely realized the complexities of the programming behind commercially available video and computer games. Many students might feel comfortable with the implementations of the graphical aspects, but the animations and interactions required a level of programming sophistication that most students had not achieved.

Dealing With Change. To gain a better understanding of the dynamics in students' game development, I compiled information about major changes that students made during the project (see Fig. 4.4). In this figure, I have documented the changes of game ideas and titles or students' thoughts about starting a new game. The information is based on evidence from their notebook writings, program implementations, or interviews. Some students, such as Darvin, Gaby, Gloria, Miriam, Shanice, or Sina, implemented their projects and pursued their beginning ideas the whole way through. Other students changed some features of the game (what I called "surface changes"), such as the title or the introduction and directions, but did not require any new programming. This happened mostly in April, when many students changed the plot of their story. Amy, Albert, Juan, Rosy, Shaun, and Tyree fall into this category. A few students, such as Jero, Sid, and Trevor, started new games in the middle of the project. Most of the changes occurred either in the first days of the project or after 3 to 4 weeks into the project.

I see these changes (the real as well as the intended ones) as indicators of a critical phase in the project, even though they occurred at different times in the project for each student, depending on the individual's game design development. In this time period, the students were in close dialogue with the situation

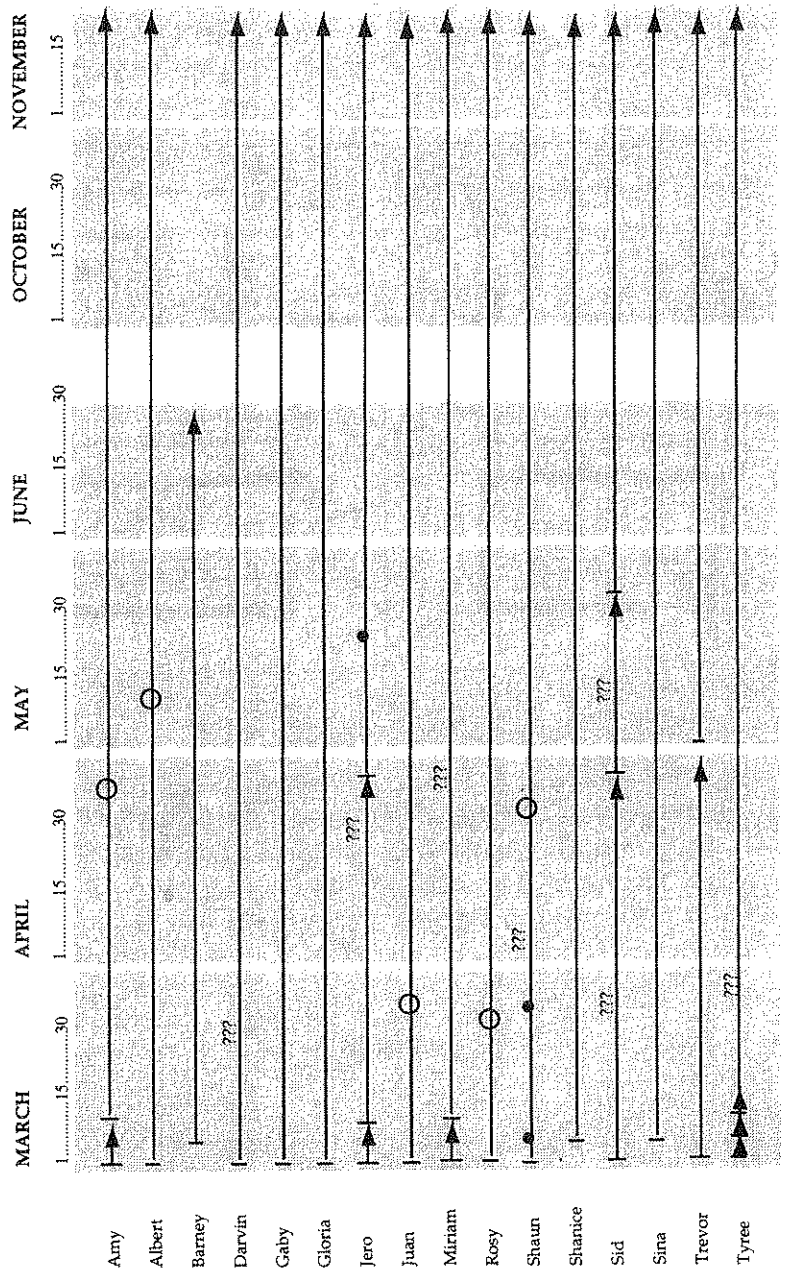
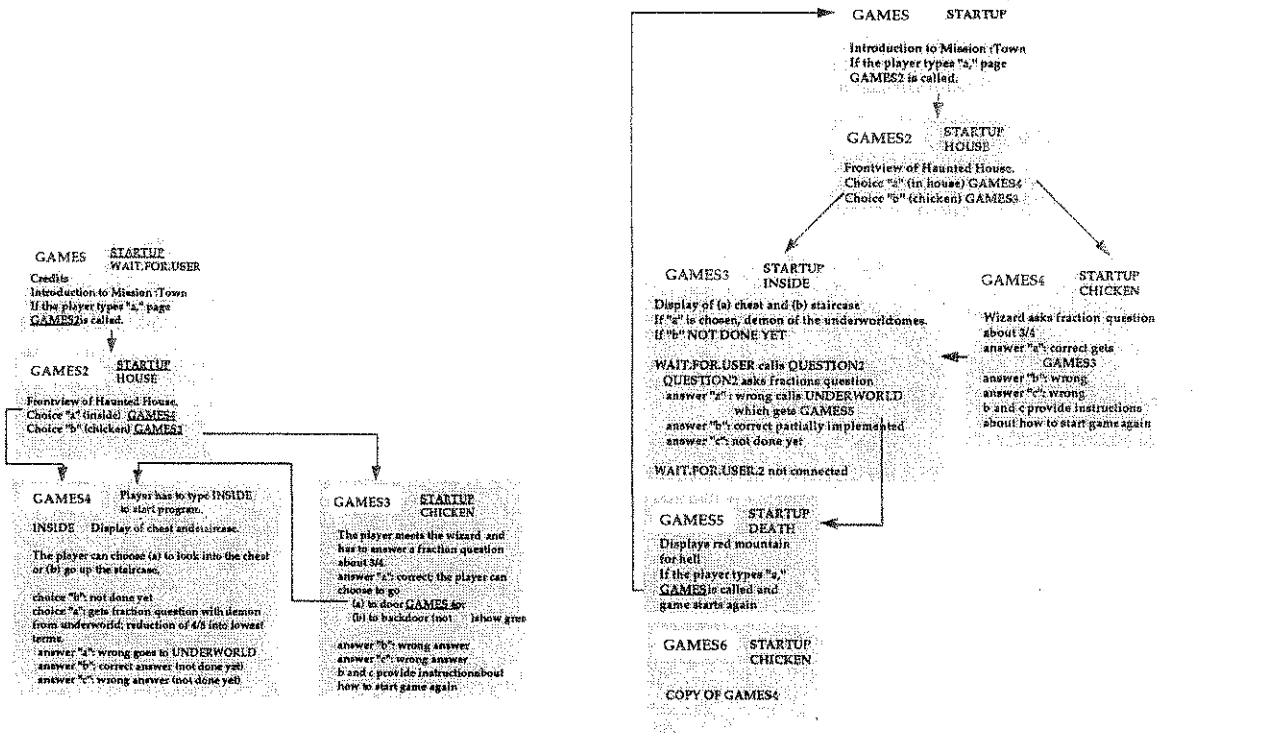
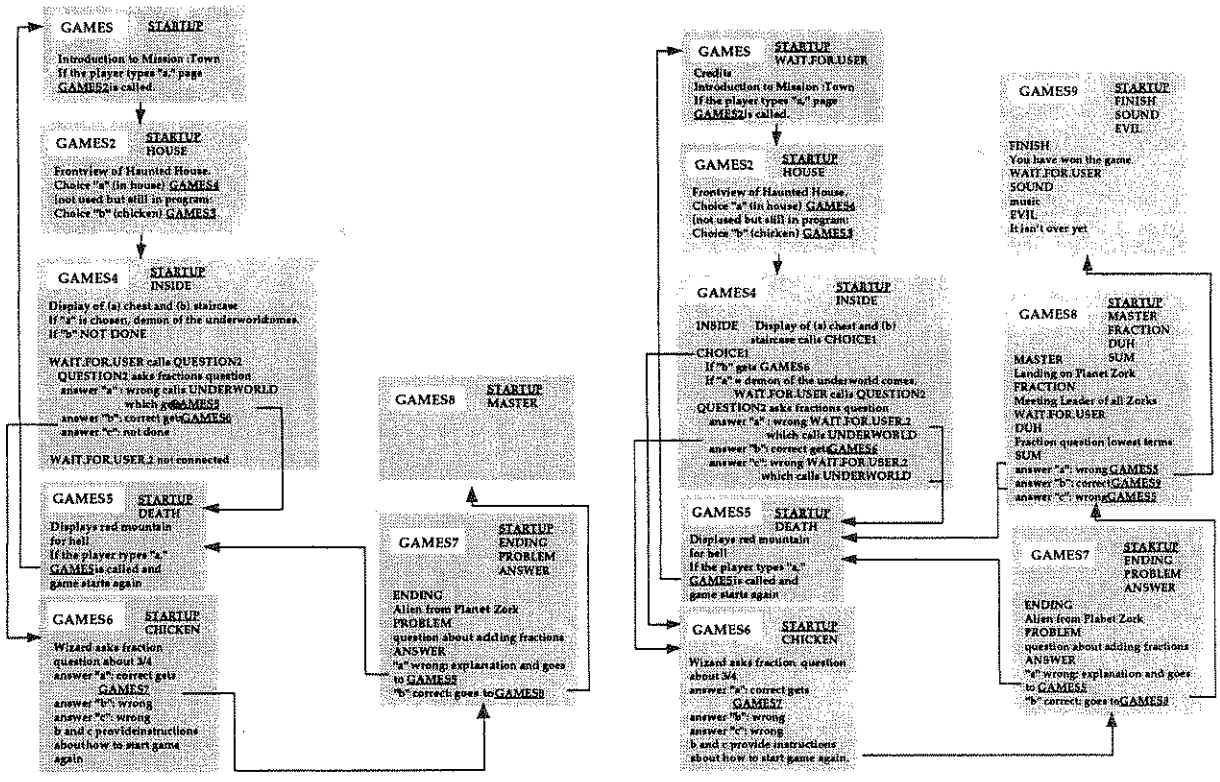


FIG. 4.4. Overview of changes anticipated and implemented by game designers.



June 19 ----------> September 25 ---------->



October 11----------> November 15

FIG. 4.5 Overview of Albert's program and procedure structure from June until November.

ɔn, 1983), or investigating whether their design was a structure adapted to ame's purpose of being fun and educational (Perkins, 1986) or whether their ideas and implementations resonated with their personal interests (Papert,). When students considered about these changes, they thought about what roject meant to them and whether or not they liked it.

loping Programming Strategies

e context of the game design process, it became apparent that students oped a wide range of strategies to deal with the complexities of the game n task. Most of these strategies developed in a later phase of the project. strategy was the modularization of program code, in which students orga- their game according to game scenes. To provide a better sense of the opment, I use two cases, Albert and Amy, as examples.

bert's use of procedures to control the growing complexity of his game am changed only in the last third of the project. All students had been luced to the concept of procedures and super procedures before the project d. Yet in the first phase of the project, Albert preferred to place one big dure on each page without segmenting the different parts of his code into idual procedures (June 19). When Albert came back after the summer and d working on his game again, his initial "one big procedure on one page" amming style changed into "several procedures organized in one super dure on one page," with each page serving as the repository for an event sting of several scenes (September 25 and October 11). In the same context, t also started using LogoWriter pages in a different way: to organize and ure his game, which, in the meantime, had grown to be dozens of different s distributed over eight pages (November 15). He used and manipulated the Writer pages as procedures (see Fig. 4.5).

bert's use of LogoWriter pages is quite unique, but it points out a general n observed in other students. All students used procedures in later parts of roject to organize their programs. This observation indicates that young nts develop modularization strategies on their own, yet they need the de- ls of a complex programming project where it makes sense to do so.

rthermore, students developed "blueprints" to deal with repetitive patterns program code. Amy's example is a good one for blueprint development. f Amy's procedures for the fraction quizzes in her game have the same s; different problems are referenced through different numbers (see .6).

ny chose to create her modules around a set of procedures and to modify accordingly. She recognized procedures as "entities, as things one could , manipulate, or change" (Papert, 1980, p. 153). From a software design of view, Amy adopted a structure that she knew to be efficient and bug-free. euse of procedures allowed Amy to be efficient in her programming produc-

PLAY	PLAY2	PLAY3	PLAY4 ....
MOVING	MOVING2	MOVING3	....
MOVE	MOVE2	MOVE3	....
PROBLEM	PROBLEM2	PROBLEM3	....
ANSWER	ANSWER2	ANSWER3	....

FIG. 4.6. Overview of Amy's procedure structure.

tion. Amy finished her first fraction problem on March 24, her second on April 3, and her third on May 6. Through this approach, she produced and covered a large part of her game structure in a short amount of time. Amy was not the only one "reusing" already written procedure parts. Other classmates did the same for their fraction problems.

DISCUSSION

The investigation confirmed the diversity within students' approaches on many levels. From the very beginning, students developed different strategies for dealing with the demands of the design situation. Some students started quickly with one idea, then abandoned it and came up with a new one. Others preferred to explore new programming commands before beginning their games. In the course of the project, students learned about design principles, such as how to adjust their design expectations about what they wanted to accomplish in relation to their programming skills and time constraints. Students dealt with this particular situation in different ways: Some considered starting a new game but continued with their originals; some implemented several superficial changes, whereas others went about making a new game. Some students, like Gaby, approached the task of designing a computer game by deciding the content of their games at the beginning and implementing one feature after the other. Others, like Barney, designed their games as they went along. Based on these results, issues concerning the prevalence of design styles, the development of programming strategies, and the provision of design support are discussed in the following sections.

In my analysis of the students' games, I looked closely at their individual approaches in dealing with the game design task, addressing both programming and design issues. My starting assumption was that people tend to organize their work in different ways, labeled as *planning* and *bricolage* (Turkle & Papert, 1991). Analysis of the students' game development seems to confirm this distinction. There were clear differences, for example, in the way Gaby approached the design task compared to Barney's approach, to name the two extreme cases. Gaby, however, could not straightforwardly implement all the features she had planned. On many occasions she had to consider the situation in relation to her programming skills, game concept, and personal aesthetics to decide on the next step. Most students, in fact, choose to walk a middle line between bricolage and

planning. At specific stages in their game design process, for example, when programming the first fraction questions, they tinkered around with different aspects: They designed different shapes and dialogues before deciding on one solution. In later stages, they switched over to reusing, in a purposeful way, program segments that they had written before (such as for the instructional dialogue). The point of this discussion is that *the clear distinction between planners and bricoleurs seems to fall apart when looking at the construction of a complex product over a long period of time.*

It is meaningful to ask in this context whether it would make sense for the students to outline all the features of their task. Several studies have asked programmers of varying degrees, from novices to experts, to solve a well-defined problem in a limited amount of time (several hours) (Guindon, Krasner, & Curtis, 1987; Jeffries et al., 1981). A general software design strategy used by most experts is to decompose a problem into smaller units. One result of these studies indicates that novices or inexperienced programmers do not have a model that guides their problem-solving process. One could then argue that the design and programming style chosen by the students reflected their lack of experience. In this particular task, however, the students' model or image of their game guided their programming. In Barney's case, it was a unspecified adventure scheme that was defined through the characteristics of the narrative. In Gaby's case, it was the model of a computer game that she had played previously and then planned to adapt to its new teaching purpose—fractions.

A further point could be made in regard to the students' use of modularization. Usually, the degree to which different scenes are decomposed is reflected in the use of procedures and super procedures. A planning style would make more use of procedures. However, analysis of the game designers' programs indicates that both planners and bricoleurs used super procedures or no procedures according to their personal preferences. The main problem encountered by the designers was the scarcity of their own design and programming experience. For example, when Sid was interviewed in October, he reflected on his own approaches to dealing with problems:

Interviewer: You got a good idea. What is the first thing you do?
Sid: First, I start from the hard thing; then I just go through the small ones so I can work my way down. Like if I want to have a major thing, I would do that right away, so I can get it out of my face. You know, just keep on doing it. So I feel that the game should end.
Interviewer: Also, if you have a good idea, you work on it immediately?
Sid: Also, I have different ideas. So I have to think for a while, then make my decision which I am going to do.
Interviewer: Where do you get most your idea?
Sid: I get my ideas like . . . first, when I started the game, well I am just typing that stuff. I get ideas. So I just keep on doing.

Sid made two points in his answers. The first point concerns his strategy for dealing with the different design components. He said about himself, "I start from the hard thing; then I just go through the small ones so I can work my way down." This more accurately describes the implementations of Sid's first two games in which he start programming the interactions and animations first. For his third and final game, however, Sid changed his strategy and designed his games as he went along. He said, "I get my ideas like . . . first, when I started the game, well I am just typing that stuff. I get ideas. So I just keep on doing." Sid's case emphasizes how both planning and bricolage can coexist in one person.

The size and complexity of the students' games by the end of the project raised the question of what kind of support for the process we can offer students. Albert's case provided one compelling example of how the growing complexity of his game created the need for structure in the form of LogoWriter page organization. A number of research efforts have been dedicated to helping software designers, especially beginners, deal with the complexity of their programs and to supporting them in their learning of proven design strategies (Guzdial, 1993; Soloway, 1988). What is pertinent to most of these approaches is that they include features in their programming environments that have been deduced from observing expert programmers at work. The reasoning behind this approach is that beginning programmers need to learn these strategies in order to become experts. Analysis of experts' work habits and processes has shown that they make extensive use of these tools to organize their work more efficiently. Although many experts share common strategies and knowledge, they do not necessarily achieve their expertise in the same fashion. This raises the substantial issue that the support structures or "training wheels" are built into a design environment (e.g., libraries, prompts, or multiple linked representations) might not have their complement in the designer's learning process.

The important point about Albert's case was that he created the organization of LogoWriter pages himself. If we had provided him with this tool in advance, I do not think that he would have taken advantage of it. This structure developed in Albert's understanding through his involvement in the long-term programming process. His recognition of its usefulness represented one of the crucial learning moments in his learning history. Now, after the Game Design Project, Albert might be ready to work with different structures to help him organize his next project. Of further importance is the observation that even beginning programmers use a variety of approaches in dealing with their problems. Hence, I leave the issue of design support structure open, but it is important that the designers of future environments take the various approaches into consideration.

CONCLUSION

To place students in the role of game designers is to confront them with a complex task. One of the insights gained from this analysis is that there are

multiple ways of designing a game; there was no one "right way" to start, continue, and accomplish a design task. Students used approaches of both planning and bricolage in order to solve the task in a successful manner. Most importantly, students learned not only through design, but also about design, and reached a level of reflection that went beyond traditional school thinking and learning. They were able to see their design experience within a larger context. As Rosemary concluded at the end of her game description: "Truthfully, I hope next time you play a video or computer game, you think about its maker."

ACKNOWLEDGMENTS

This chapter is based on a presentation called "Children's Design Styles" given at the annual meeting of the American Educational Research Association in New Orleans in April 1994. The results presented here are based on my thesis research. I wish to thank my thesis committee members, David Perkins, Seymour Papert, Idit Harel, and Terry Tivnan, for their help and insightful comments. I also thank Joanne Ronkin and her students for their collaboration and great contribution to this work. Without them, this research would not have been possible. The research reported here was conducted at Project Headlight's Model School of the Future and was supported by the IBM Corporation (Grant OSP95952), the National Science Foundation (Grant 851031–0195), the MacArthur Foundation (Grant 874304), the LEGO Group, Fukutake, and Apple Computer, Inc. The preparation of this chapter was supported by the National Science Foundation (Grant 8751190-MDR) and Nintendo Inc. The ideas expressed here do not necessarily reflect the positions of the supporting agencies.

REFERENCES

Avedon, E. M., & Sutton-Smith, B. (Eds.). (1966). *The study of games*. New York: Wiley.

Bjorklund, D. F. (Ed.). (1991). *Children's strategies: Contemporary views of cognitive development*. Hillsdale, NJ: Lawrence Erlbaum Associates.

Block, J. H., & King, N. R. (Eds.). (1987). *School play*. New York: Garland.

Blumenfeld, P. C., Soloway, E., Marx, R. W., Krajcik, J. S., Guzdial, M., & Palincsar, A. (1991). Motivating project-based learning: Sustaining the doing, supporting the learning. *Educational Psychologist, 26*(3 & 4), 369–398.

Bride, J. W., & Lamb, C. E. (1991). Using commercial games to design teacher-made games for the mathematics classroom. *Arithmetic Teacher, 38*, 14–22.

Brown, A. L., & DeLoache, J. S. (1978). Skills, plans and self-regulations. In R. S. Siegler (Ed.), *Children's thinking: What develops?* (pp. 3–35). Hillsdale, NJ: Lawrence Erlbaum Associates.

Carver, S. (1991, April). *Interdisciplinary problem solving*. Paper presented at the American Educational Research Association, Chicago, IL.

Carver, S. M. (1987). *Transfer of LOGO debugging skills: Analysis, instruction and assessment*. Unpublished doctoral thesis. Pittsburgh, PA: Carnegie-Mellon University, Department of Psychology.

Collins, A. S., Brown, J. S., & Newman, S. (1990). Cognitive apprenticeship: Teaching the craft of reading, writing and mathematics. In L. B. Resnick (Ed.), *Cognition and instruction: Issues and agendas* (pp. 453–434). Hillsdale, NJ: Lawrence Erlbaum Associates.

Dreher, M., & Oerter, R. (1987). Action planning competencies during adolescence and early adulthood. In S. L. Friedman, E. K. Scholnick, & R. R. Cocking (Eds.), *Blueprints for thinking* (pp. 321–355). Cambridge, MA: Cambridge University Press.

Fennell, F., Houser, L. L., McPartland, D., & Parker, S. (1984, February). Ideas. *Arithmetic Teacher, 31*, 27–33.

Friedman, S., L., Scholnick, E. K., & Cocking, R. R. (Eds.). (1987). *Blueprints for thinking*. Cambridge: Cambridge University Press.

Guindon, R., Krasner, H., & Curtis, B. (1987). Breakdowns and processes during the early activities of software design by professionals. In G. M. Ohlson, S. Sheppard, & E. Soloway (Eds.), *Empirical studies of programmers: Second workshop* (pp. 65–82). Norwood, NJ: Ablex.

Guzdial, M. (1993). *Emile: Software-realized scaffolding for science learners programming in mixed media*. Unpublished doctoral dissertation, Ann Arbor, MI: University of Michigan.

Harel, I. (1988). *Software design for learning: Children's construction of meaning for fractions and logo programming*. Unpublished doctoral dissertation, Media Laboratory, Cambridge, MA: Massachusetts Institute of Technology.

Harel, I. (1991). *Children designers*. Norwood, NJ: Ablex.

Harel, I., & Papert, S. (1990). Software design as a learning environment. *Interactive Learning Environment, 1*(1), 1–32.

Jeffries, R., Turner, A. A., Polson, P. G., & Atwood, M. E. (1981). The processes involved in designing software. In J. R. Anderson (Ed.), *Cognitive skills and their acquistion* (pp. 225–283). Hillsdale NJ: Lawrence Erlbaum Associates.

Kafai, Y. B. (1993). *Minds in play: Computer game design as a context for children's learning*. Unpublished doctoral dissertation, Cambridge, MA: Harvard Graduate School of Education.

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

Kreitler, S., & Kreitler, H. (1987). Conceptions and processes of planning: the developmental perspective. In S. L. Friedman, E. K. Scholnick, & R. R. Cocking (Eds.), *Blueprints for thinking* (pp. 205–272). Cambridge: Cambridge University Press.

Lehrer, R. (1991). *Knowledge as design*. Paper presented at the American Educational Research Association, Chicago, IL.

Lepper, M. R., & Malone, T. W. (1987). Intrinsic Motivation and Instructional Effectiveness in Computer-Based Education. In R. E. Snow & M. J. Farr (Eds.), *Aptitude, learning and instruction. Volume 3: Conative and affective process analyses* (pp. 255–285). Hillsdale, NJ: Lawrence Erlbaum Associates.

Newell, A. & Simon, H. A. (1972). *Human problem solving*. Englewood Cliffs, NJ: Prentice-Hall.

Papert, S. (1993). *The children's machine*. New York: Basic Books.

Papert, S. (1980). *Mindstorms*. New York: Basic Books.

Pea, R., & Hawkins, J. (1987). Planning in a chore-scheduling task. In S. L. Friedman, E. K. Scholnick, & R. R. Cocking (Eds.), *Blueprints for thinking* (pp. 273–302). Cambridge: Cambridge University Press.

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

Perkins, D. N. (1986). *Knowledge as design*. Hillsdale, NJ: Lawrence Erlbaum Associates.

Priester, S. (1984, March). SUM 9.9: A game for decimals. *Arithmetic Teacher, 31*, 46–47.

Resnick, M. (1991). Xylophones, hamsters, and fireworks: The role of diversity in constructionist activities. In I. Harel & S. Papert (Eds.), *Constructionism*. Norwood, NJ: Ablex.

Schauble, L. (1991, April). *A developmental psychology of design: Generating and coordinating*

constraints. A discussion for the symposium "Kids as Designers" at the American Educational Research Association, Chicago.
Schön, D. A. (1983). *The reflective practitioner*. New York: Basic Books.
Simon, H. A. (1969). *The sciences of the artificial*. Cambridge, MA: MIT Press.
Soloway, E. (1988). It's 2020: Do you know what your children are learning in programming class? In R. S. Nickerson & P. P. Zodhiates (Eds.), *Technology in education: Looking toward 2020* (pp. 121–130). Hillsdale, NJ: Lawrence Erlbaum Associates.
Turkle, S., & Papert, S. (1991). Epistemological pluralism: Styles and voices within the computer culture. In I. Harel & S. Papert (Eds.), *Constructionism* (pp. 161–193). Norwood, NJ: Ablex.

5

Electronic Play Worlds
Gender Differences in Children's Construction of Video Games

Yasmin B. Kafai

Children's culture of the late 20th century—their toys, games and activities—has been marked by the advent of information technologies. Video games, more than any other medium, have brought interactive technologies into children's homes and hearts, and they have been received enthusiastically. In contrast to most adults, children usually do not feel threatened by computational media and other programmable devices. They seem to embrace the new media readily. With good reason, children have been dubbed media enthusiasts (Papert, 1991).

This increasing presence of computer and video games in homes has initiated many discussions in the media and educational circles about their value and influence on children's affective, social, and cognitive well-being (Baughman & Clagett, 1983; Provenzo, 1991). Most research efforts have focused on studying the effects on social behavior and cognitive skills of children playing video games (Greenfield & Cocking, 1994; Loftus & Loftus, 1983; Selnow, 1984). Discussions in cultural studies have centered around the issues of which particular messages are promoted in video games and in what ways they are received by children (Gailey, 1992; Kinder, 1991). As important as these findings are, researchers always look at children as consumers of games and try to deduce from children's game-playing interests and behaviors what impact and attractions video games hold.

In the present study, I address some of these issues from a different perspective by placing children in the role of producers rather than consumers of video games. In a 6-month-long project, called the Game Design Project (Kafai, 1993, 1995), sixteen 10-year-old children were in charge of creating imaginary worlds, characters, and stories in the context of video game design. My intention was to explore the extent to which the activity of making games revealed something