Learning Design by Making Games
Children's Development of Design Strategies in the Creation of a Complex Computational Artifact

Yasmin B. Kafai
University of California, Los Angeles

"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 four or five 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 the above quotes of young game designers illustrate, there is something to be learned from making games that goes beyond learning programming and particular subject matters. The students' descriptions reflect on their early expectations about

making games which were most probably influenced by their experiences with regular classroom work and home assignments. None of these students had ever worked on a complex task and gained experience in design, i.e., planning, problem solving, doing research, dealing with time constraints, modifying expectations, and bringing it all together in one game. It even is not clear whether young students are able to accomplish such projects as previous research suggests that children have limited abilities in planning and dealing with complex tasks (for an overview see: Scholnick, Friedman, & Cocking, 1987; Bjorklund, 1991).

However, recent efforts in the educational research community have stressed the importance of self-directed, personally meaningful and cognitively complex projects for students' learning success (Blumenfeld et al., 1990; Brown, Collins, & Duguid, 1989; Harel & Papert, 1991; Newman, Griffith, Cole, 1989). Design activities are one example of such project-based activities in which students are engaged in designing complex, interactive pieces of software for the use of other students to learn about a particular subject area (Harel, 1991). A variety of design activities have been proposed ranging from creating interactive presentations to instructional software, simulations, and educational games (Carver, 1991; Guzdial, 1993; Kafai, 1993; Lehrer et al., 1991). A prominent feature of these activities is that students learn through design about academic matters such as programming, history, mathematics, and physics and about design by managing the projects.

One of the objectives of the research project presented in this paper was then to examine in which ways children approach such a complex design task. A class of fourth-grade students became game designers and worked daily on designing games with the goal of making the learning of fractions more 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 in their school. Students' design progress was observed and analyzed from the beginning to the completion of the product for a period of six months. The following sections provide the theoretical background, describe in more 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 case studies have been included to provide a better sense of re-occurring themes and trends.

REVIEW OF RESEARCH

The particular background for the research on children's design approaches is based on constructionist theory (Papert, 1980, 1990). This view sees learners as builders of their own mental tools and structures (Piaget, 1968). Learning happens best when students are engaged in building external and shareable objects such as computer programs, machines, or games. Learning through and about design offers one model of constructionist learning activities (Harel, 1988, 1991). Learners are engaged in a continuous dialogue with their own ideas, those of intended users and co-designers and makes the learning of any subject instrumental to a larger intellectual and social goal: to create something for the use of others. Student-designers assume control in their learning through asking questions and gathering information and putting all this to work by creating an educational game. Shannon, an architect, pointed out "[t]he difference between simply doing something and designing something is in the level and quality of commitment and consideration given to the task, and how one feels in accomplishing it. ... Designing on the other hand cannot be automatic. It always forces critical thinking, judgment—personal involvement" (1990, p. 40). Designing, then puts students in charge and engages them in a continuous dialogue with their own ideas and with the ideas of intended users and co-designers. Students assume control in their learning through asking questions, gathering information, and putting all this to work by creating an educational game for younger students to learn about fractions.

In the context of this research, design is an object of study as well as a context for a study. 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. Schaeuble (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 students with the opportunity to be engaged with different facets of the design process. For example, the designer starts finding a problem, and finally continues with parts of the solution, tries to make sense out of it, then considers how to re-frame the situation and continues with problem solving. In this process, the designer constructs a particular, personal relationship to the artifact which undergoes changes as the process continues. This process seems to stop when an artifact has been created but actually never ends since existing design solutions are used and re-used in new design

situations.

It is important to point out a distinction between learning through design and professional design. The learning in design is not exclusively represented in the final product, but also is 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 partially reflected in the product). In design for learning, the process is more important than the product. This means 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.

The traditions of school and academia have favored formal and abstract approaches in 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" (Newell & Simon, 1972, Jeffries et al., 1981) or "planning" versus "bricolage" (Turkle & Papert, 1990):

> 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, 1990, 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 sub problems (Jeffries et al, 1981; Guindon, Krasner, & Curtis, 1987). Here the designer maps out context, content, and structure of a design at the beginning. The opposing view, called bricolage or bottom-up, describes problem solving rather 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 with the game design project was to offer a design activity that had multiple avenues for students to approach a complex problem, build a game around the idea of fractions, and find a personal solution. 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 which 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 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 matters. 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 since previous research on the development of children's abilities to deal with complex tasks suggests that children know how to plan but being limited by their knowledge of the situation (for an overview see: Scholnick, Friedman, & Cocking, 1987; Bjorklund, 1991). In order to investigate these issues, I chose a familiar, yet complex task: I investigated children's design approaches in programming a game. Unlike most research from the problem solving domain, the students were given neither a well-defined nor an ill-defined problem which 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 paving their own paths of learning and activities. But while doing so they also were building microworlds of their own—game worlds for others to play and learn with. The goal of this investigation was then to analyze whether particular design approaches emerged over time, as they have 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 sixteen fourth grade students were engaged during an extended period of time (6 months) in the design and production of educational games to teach fractions. The students worked for one hour a day on their projects in class. In general, students first spent 5 minutes writing in their plans and ideas in notebooks before they went to work on their own computers for 45 minutes. As students were working on their games, they were allowed to walk around and see and discuss each others' projects. Students then returned to their classroom and wrote again about their experiences. (For a more detailed description, I refer the reader to the subsequent paper "Electronic Play Worlds" in this volume.)

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. Since the purpose of this project was to create a finished product, students also worked in parallel in their art classes on cover designs for packaging their games; and in language/arts they worked on advertising their product and the documentation for it. During the six 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 (Papert, 1986; 1987) which is located at an inner-city public elementary school in one of Boston's low socio-economic neighborhoods. The goal of Project Headlight, which started in 1985, is to explore the use of computers in the classrooms as they might exist in the near future. Project Headlight operates as a "school-within-a-school" as the two other thirds of the school

have traditional classroom and computer arrangements. One special feature of this elementary school is its open-architecture structure which was virtually unused by the school before the project started. In the two open areas, four clusters of computers are arranged with sixteen computers each in a circle, screens facing out. The clusters of computers are called "pods." As students come and leave their classrooms, they have a chance to walk along the computer pods and see the projects of other students. 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 participate in this project, ranging from first through fifth grade and including advanced, regular, bilingual and special educational classes. The participants had mixed ethnic background and were half boys and girls. Most of them had joined the school only for the fourth and fifth year, hence their beginning programming experience was not very extensive.

METHODS OF DATA COLLECTION AND ANALYSES

The game development of sixteen game designers was observed every day over the course of six 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 and video games, and their 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 was examined in the following ways: (1) the students' projections of their game design in the notebook and interviews compared to their actual program implementations in Logo. Particular attention was paid to the beginning phase when students began formulating their first game ideas and what expectations they had about upcoming difficulties, and the period of first transition, three to four weeks into the study, when students had accomplished part of their implementations and had already handled design and programming problems; (2) the development of the game theme and features over time; and (3) the development of the program code in regard for the use of modularization, i.e., making procedures. Using procedures has been described in the research literature as one important programming strategy (Pea & Hawkins, 1984; Carver, 1986).

RESULTS

During the 92 days of thinking, designing, programming, modifying, and playing their computer games, the students were involved in many activities and touched upon many issues.[1] The extended time frame allowed for a closer investigation of the project trends and development. As the students started implementing their game ideas, they put to test their programming skills, and they learned about the feasibility of their game ideas and different aspects about the design process. It is worthwhile investigating transitions or shifts and whether they occurred throughout the class or whether they were only experienced by individual students. Over time, two kinds of developments could be observed: one kind was of incremental nature as reflected in the daily growth of the game-designers' programs of a few lines of Logo code; the other kind represented shifts in the nature of the games as well as the approach chosen by the designers. By looking at the development of the students' games as a whole, from the beginning to the end, and how students outlined features of their games and also set out to implement them provides a first take on distinct design approaches.

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

Design Styles

One of the objectives of this analysis was to investigate whether bi-polar descriptions such as "top-down" or "planner" versus "bottom-up" or "bricoleur" assess correctly children's approaches to a complex task such as designing a game. By looking at the development of the students' games as a whole, from the beginning to the end, and how students outlined features of their games and also set out to implement them provides a first take on distinct design approaches. We could expect that "planners" would carefully lay out what they want to do in advance. We would expect "bricoleurs" to develop their game as they go along. To make this point clearer, I chose Barney's and Gaby's games as two extreme examples of game development (see Figure 1).

<Insert Figure 1 about here>

In her first interview, Gaby described the different parts that she will implement for her spider web game. She already segmented her descriptions of the different game components into the blocks, the web, and the fractions. When Gaby started implementing her game, she first did the spider web, then the blocks on the screen, then the colored blocks, before she began 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 different color has a different prefix (LBLOCK, MBLOCK and so on). One could say that Gaby laid out from day one of the project 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, and programmed the spider 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 fish speaking. As Barney set out to implement his game, he started with a welcome screen, introduction and directions. From then on, he develops 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 that are 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 the same planning behavior but to a lesser extent. Their planning seems to be more localized and deal with particular sequences. I refer the reader here in particular to Amy's blueprint for the instructional quiz that she clearly planned ahead (see the section on “Developing Programming Strategies”). Or 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 pre-formed 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 students' game design process. The results of my analyses indicate that most students chose to walk a middle line between bricolage and planning. 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 upon 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 clear distinction between planners and bricoleurs seems to fall apart when observed over a long period of time and the construction of a complex product. This result speaks against bi-polar descriptions. There is further evidence in the game design process that students did not adopt one particular strategy but rather negotiated the situation in imposing and coordinating multiple constraints with their personal interests in game design (Schaeuble, 1991). To provide better support for this statement it is important to examine the beginning phase and how 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. In the beginning phase, some students did not start working and implementing their game ideas right away; others started with one idea, abandoned it quickly and moved on with a new one. Many changes (title of game, new introduction, main game idea) in the students' game design occurred either in the first days of the project or after three to four 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. Many students did not start working and implementing their game ideas right away. whereas others started with one idea, abandoned it quickly and moved on with a new one. Those who did not start programming immediately their games explored or played around with other aspects seemingly unrelated to their assignment. For example, Albert and Shanice

experimented both with some new Logo commands to which they had been introduced a few days before. Barney refused to start the project at all because, as he told me, he did not have any game ideas with 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 over 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 that they were not confident about their first choice. Instead, their games seemed to take a new turn from one day to the next.

The students used both approaches to the design task and to enter the new domain of game design. I do not interpret either approach as unproductive or inactive. In both cases, it provided a time when students grappled with the issue of bringing together the two domains of fractions and games in one design. In Harel's (1991) instructional software design project, Debbie actually took several days before she started with the design of her instructional screens. Very much in the same way that professional designers would start out, the students in the game design project either tried out alternative ideas and designs or decided to postpone the beginning of 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 of diversity. Some students, such as Barney, Tyree, Shaun, Juan, and Rosy, started with their welcoming screen or introductions (the first screens that the player usually sees 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 introductions until midway or towards the end of the project. The diversity of project themes, working styles, and of entry paths in design activities has been described and discussed by Resnick (1991).

Definitions of the Game Design Task. Very early on students developed game ideas that integrated fractions in different ways. Two distinct game formats were adopted by

most students: extrinsic and intrinsic integration of fractions into the game (called Game Worlds and 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 integration is exemplified in a game where the designer takes care of integrating 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 ideas of 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 games here make reference to commercially available games such as Nintendo. For example, Jero's different warp zones are reminiscent of Mario Brother's tunnel system. Oscar made explicit reference to the Pacman game, but 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 better "screaming", a fraction. Gloria came up with a similar idea (see Figures 2a and b).

<Insert Figure 2a about here>

<Insert Figure 2b about here>

Fraction Worlds, in 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 Figure 3).

<Insert Figure 3 about here>

Games with intrinsic integration could also be considered a form of microworlds in which fractions are situated. Papert defines 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 the learner to explore their knowledge. Here the students became the designers of software tools for other children. Students 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 addresses the very nature of educational games. Educational games lie between learning and playing. They are a particular breed of games because the rules necessary to play the game demand the use of valuable educational skills. The students obviously felt the tension between the demands of playing and 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. Educational games on and off the computer have been used for a long time by teachers to keep students motivated (Block & King, 1987; Avedon & Sutton-Smith, 1966; Lepper & Malone, 1987). Teacher-designed games shared an aspects 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 accomplishing the tasks. Many of the students' 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 constrained 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 relatively easy: When answering a question correctly is what allows the next move in a game, the question can be on any topic. But this is also a weakness, because it causes the designer to loose 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 are 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 observations 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), both Amy and Sid were not concerned with any problems.

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

Yasmin: *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.

Yasmin: *That sounds like a neat idea. The pacman who is eating fractions?*

Oscar: *That's what I wanted to do.*

Yasmin: *So work on that.*

Oscar: *It's a little bit too hard.*

Yasmin: *So can you think of a simpler version?*

Oscar: *I am trying ... but I am going to think of something different.*

Yasmin: *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.*

Yasmin: *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].

Yasmin: *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.

Yasmin: *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 the expectations how much they could accomplish in the project. One example is the number of fraction problems that 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 9 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 experiences 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 students reduced their expectations and the average number of instructional situations is 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 for 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." Or Amy who 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 three to four weeks on one assignment and then moved on to the next one. Their planning skills and estimations in regard to the game design were based upon 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 to map out from the beginning their plans and ideas for implementations, it would be unrealistic to think that they can take into consideration all the aspects would be involved in planning and designing a game. Related to this might be that few students 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 Figure 4). In this figure, I documented the changes of game ideas and titles or when students just thought 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 project and pursued the same idea from the beginning. 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 three to four weeks into the project.

<Insert Figure 4 about here>

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 (Schön, 1983) or investigated whether their design was a structure adapted to the game's purpose of being fun and educational (Perkins, 1986) or whether their game ideas and implementations resonated with their personal interests (Papert, 1980). When students thought about these changes, they thought about what this project meant to them and whether they liked it or not.

Developing Programming Strategies

In the context of the game design process, it became apparent that students developed a wide range of strategies to deal with the complexities of the game design task. Most of these strategies developed in the later phase of the project. One strategy was the modularization of program code when students organized their game according to game scenes. To provide a better sense of the development, I use two cases, Albert and

Amy, as examples.

Albert's use of procedures to control the growing complexity of his game program changes only in the last third of the project. All students had been introduced to the concept of procedures and super procedures before the project started. Yet in the first phase of the project, Albert preferred to place one big procedure on each page without segmenting the different parts of his code into individual procedures (June 19). When Albert came back after the summer and started working on his game again, his initial "one big procedure on one page" programming style changed into "several procedures organized in one super procedure on one page" where each page was the repository for an event consisting of several scenes (September 25 and October 11). In the same context, he also started using Logo Writer pages in a different way: to organize and structure his game which in the meantime has grown to be dozens of different scenes distributed over eight pages (November 15). He used and manipulated the LogoWriter pages as procedures (see Figure 5).

<Insert Figure 5 about here>

Albert's use of Logowriter pages is quite unique but points out a general pattern observed in other students. All students used procedures in the later parts of the project to organize their programs. This observation indicates that young students develop modularization strategies on their own; yet they need the demands of a complex programming project where it makes sense to do so.

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

<Insert Figure 6 about here>

Amy chose to create her modules around a set of procedures and to modify them accordingly. Amy recognized procedures as "entities, as things one could name, manipulate or change" (Papert, 1980, p. 153). From a software design point of view, Amy has adopted a structure that she knows is efficient and bug free. The reuse of procedures allows Amy to be efficient in her programming production. 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 is not the only one who is "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, abandoned it and came up with a new one, whereas others preferred to explore new programming commands before beginning their game. 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; whereas others, like Barney, designed their games as they went along. Based on these results, issues around the prevalence of design styles, development of programming strategies, and the provision of design support will be discussed in the following sections.

In the 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 point was that people tend to organize their work in different ways and these have been labeled as planning and bricolage (Turkle & Papert, 1990). 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, to name the two extreme cases. But also Gaby could not implement straightforwardly all the features she had planned. On many occasions she had to converse with the situation in regard to her programming skills, game concept, and personal aesthetics to decide upon 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 upon 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 over a long period of time and the construction of a complex product**.

It is meaningful to ask in this context whether it would make sense for the students to outline all the features of the 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) (Jeffries et al, 1981; Guindon, Krasner, & Curtis, 1987). 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. But in this particular task, 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 that she planned to adapt to its new teaching purpose—fractions.

A further point could be make 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. One example here is Sid; when interviewed in October, he reflected on his own approaches to dealing with problems:

Cora: *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.*

Cora: *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.*

Cora: *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 makes 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 description is more accurate of Sid's implementations of his first two games where 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 co-exist 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 we can offer to students in the process. Albert's case provided one compelling example of how the growing complexity of his game created his 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 et al., 1991; 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 experts make extensive use of these tools to organize their work more efficiently. Although many experts share common strategies and knowledge, they did not necessarily achieve their expertise in the same fashion. This raises the substantial issue that whatever "training wheels" are built into a design environment, such as 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 himself created the organization of LogoWriter pages. I do not think that if we had provided him with this tool in

advance he would have taken advantage of it. This structure developed in Albert's understanding through the long-term programming process in which he was involved. His recognition of its usefulness represented one of his crucial learning moments in his learning history. Maybe 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 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 previous aspects into consideration.

CONCLUSION

To place students in the role of game designers confronts 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 both approaches of 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 paper is based on a presentation called "Children's Design Styles" given at the annual meetings 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 wish to thank Joanne Ronkin and her students for their collaboration and their 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 Company, Fukutake, and the Apple Computer, Inc. The preparation of this paper was supported by the National Science Foundation (Grant # MDR 8751190) and Nintendo Inc., Japan. The ideas expressed here do not necessarily reflect the positions of the supporting agencies.

REFERENCES

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

Carver, S. M. (1987). Transfer of LOGO Debugging Skills: Analysis, Instruction and Assessment. Unpublished Doctoral Thesis. Pittsburgh, PA: Carnegie-Mellon University, Department of Psychology.

Chipman, S. F., Segal, J. W., & Glaser, R. (Eds.) (1985). *Thinking and Learning Skills. Volume 1 and 2*. Hillsdale, NJ: Erlbaum.

Clements, D. H. (1987). Longitudinal Study of the Effects of Logo Programming on Cognitive Abilities and Achievement. *Journal of Educational Computing Research*, 3 (1), 73-94.

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*. Cambridge: Cambridge, University Press, 321-355.

Friedman, S., L., Scholnick, E. K., & Cocking, R. R. (1987). Reflections on reflections: what planning is and how it develops. In S. L. Friedman, E. K. Scholnick, & R. R. Cocking, (Eds.), *Blueprints for thinking*. Cambridge: Cambridge, University Press, 515-534.

Foreman, G. (1985). The value of kinetic print in computer graphics for young children. In E. Klein (ed.), *Children and Computers: New Directions for Child Development*. No.28. San Francisco: Jossey-Bass.

Gilligan, C. (1982). *In a different voice*. Cambridge, MA: Harvard 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*. Norwood, NJ: Ablex.

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

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*. Hillsdale NJ: Erlbaum.

Kafai, Y. B. (1993). *Minds in Play—Computer Game Design as a Context for Children's Learning*. Unpublished Doctoral Dissertation, Harvard Graduate School of Education, Cambridge, MA.

Kafai, Y. B. (1993, April). *Constructing Mathematical Representations: Lessons Learned from two Design Task.*

Paper presented at the Annual Meeting of the American Educational Research Association, Atlanta, GA.

Kafai, Y. B. (1992, April). *Learning through Design and Play: Games as a Context for Children's Explorations of Fractions and Logo.* Paper presented at the Annual Meeting of the American Educational Research Association, San Francisco, CA.

Kinder, M. (1991). *Playing with Power*. Berkeley: University of California Press.

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

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

Palumbo, D. N. (1990). Programming Language/Problem-Solving Research: A Review of Relevant Issues. *Review of Educational Research*, 60 (1), pp. 65- 89.

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

Papert, S. (1993). *The Children's Machine*. 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*. Cambridge: Cambridge, University Press, 273-302.

Perkins, D. N. (1990). The nature and nurture of creativity. In B. F. Jones & L. Idol (Eds.), *Dimensions of thinking and cognitive instructions*. Hillsdale, NJ: Lawrence Erlbaum Associates.

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

Perkins, D. N. (1981). *The mind's best work*. Cambridge, MA: Harvard University Press.

Provenzo, E. F. (1991). *Video Kids: Making Sense of Nintendo*. Cambridge, MA: Harvard University Press.

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.

Turkle, S. & Papert, S. (1991). Epistemological Pluralism: Styles and voices within the computer culture. In. I. Harel & S. Papert, (Eds.), *Constructionism.* Norwood, NJ: Ablex Publishing Corporation.

[1] Other parts of my investigation addressed the conceptual development of students' dealing with fractions, their understanding of the programming language Logo, their mastery of particular programming concepts and 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).