HUMAN-COMPUTER INTERACTION, 1993, Volume 8, pp. 337-352

Is It Easier to Hop or Walk? Development Issues in Interface Design

Erik F. Strommen
Children's Television Workshop

ABSTRACT

Thirty-six 3-year-old children used a Nintendo controller to play a simple video game that required the child to capture both moving and stationary onscreen targets by positioning a Sesame Street character under them, then making the character jump to capture them. Two different forms of character movement were tested: moving in discrete steps ("hopping") and moving in a smooth, continuous motion ("walking"). Targets were the same for both movement types. Results indicated that, although there was no difference between movement types in number of targets successfully captured, continuous movement was significantly more challenging for children, both when positioning the cursor and when trying to capture targets. Results are discussed with reference to possible cognitive factors governing children's game performance, and implications for the design of interactive materials for preschoolers are considered.

Author's present address: Erik F. Strommen, Director of Research for Interactive Technologies, Children's Television Workshop, One Lincoln Plaza, New York, NY 10023.

CONTENTS

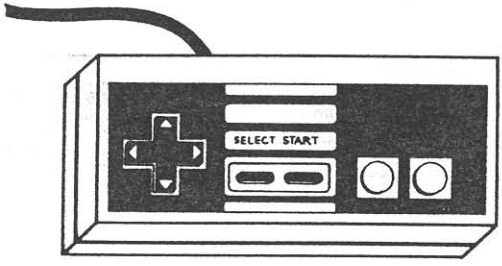
1. INTRODUCTION

For more than a decade, the Interactive Technology Division (ITD) at Children's Television Workshop (CTW) has been producing educational software for small children (Children's Computer Workshop, 1983; CTW, 1987, 1989b; Rice, 1987). Just as Sesame Street's mission is to utilize television as a medium to provide learning opportunities for preschoolers in home settings, the ITD has recently begun to exploit home video-game systems, such as the Nintendo Entertainment System (NES), as platforms for providing educational experiences for young children as well. To date, we have produced four Nintendo cartridges (CTW, 1988, 1989, 1990, 1991) designed for preschool users. The goal of such products is to provide educational activities for children that are designed in a playful, informal manner that capitalize on the appeal of this ubiquitous technology.

A review of the evolution of the design of these products informs the issues addressed in the present study. The first three products (CTW, 1988, 1989, 1990) all resemble simple educational software products. They all present the child with a single screen and use an extremely simple nondirectional interface that cycles a cursor through possible onscreen choices in a fixed order. This interface, designed for use by extremely young children, was a deliberate simplification of the functionality of the standard NES interface, and it was based on informal observations of 3-year-olds unable to use the standard Nintendo interface. As with adult interface design, the goal was to provide children with a very simple way to engage the content of the products and not bog them down with interface conventions that force them to perform special actions or require extra cognitive effort to manipulate the cursor. When we began design work of the fourth Nintendo cartridge (CTW, 1991), however, we wanted to design a product that was more like the games older children use on the NES. We were prompted to conduct studies to answer two questions: Exactly what are the sources of children's difficulty with the NES interface? How similar could an interface for preschoolers be to a standard NES interface?

It is striking to consider that the competence of preschoolers when using the Nintendo interface, a ubiquitious and highly popular technology, has been

Figure 1. **The standard input device for the Nintendo Entertainment System.**



largely neglected as a research topic in both developmental psychology and in human interface studies. The NES is operated by a small, handheld plastic input device measuring 4.75 × 2.0 × ⅝ in. (hereafter called the NES controller). A cross-hair button and two selector buttons are used to move a cursor, usually in the form of a small figure, on the television screen (see Figure 1). The cross-hair button is shaped like a plus sign or cross, and each arm corresponds to one of four directions: up, down, left, and right. The arms are all part of a common button, but they operate in a manner analogous to keyboard arrow keys: Pressing each arm of the button makes the character/cursor move in the direction pressed. The correspondence between directions of movement and the arms of the cross hair mean that, for proper use, the device must always be held lengthwise, so that the arms match their appropriate directions. The two red selector buttons serve various functions, depending on the game being played. Typically, one makes the character jump, and the other registers a game-specific response when needed.

Little is known about young children's competence with the NES controller. Previous research (Grover, 1986; Revelle & Strommen, 1990; Revelle, Strommen, & Offerman, 1990) had indicated that young children were unable to use directional input devices similar to the Nintendo cross-hair button, such as keyboard arrow keys and joysticks, to direct cursors to onscreen targets. It was hypothesized that the difficulty of translating the direction of cursor movement on the screen into the movements required of the user with the input device was the source of children's difficulty, and this hypothesis led us to design our NES game interfaces so that directional control was not required. We tested this hypothesis in the single extant study of young children's ability to use the Nintendo controller (Strommen, Razavi, & Medoff, 1992). Strommen et al. compared the performance of two groups of 3-year-old children with no Nintendo experience using two highly constrained software interfaces on the NES system. The experimental task presented children with the image of a brownstone with four windows (two above, two

below), each of which contained a Sesame Street character. The goal of the task was to use the NES controller to select a given character by moving a cursor, in the form of a small bird, to the window with the correct character in it. Two interfaces were tested. The *directional interface* employed the cross-hair button in the standard manner: Pressing an arm of the button moved the bird to the next window in the direction indicated, and it stopped there. For the *nondirectional interface,* the directional function of the cross-hair button on the Nintendo controller was disabled, just as it was in CTW's previous Nintendo products: Pressing any part of the button simply caused the bird to fly to the next window in a clockwise sequence and stop there. For both conditions, pressing a red button confirmed the window choice. The purpose of the study was to assess which interface resulted in more accurate cursor placement, which was assessed on a variety of levels (e.g., number of incorrect placements, number of "overshoots," or times the cursor passed its intended target).

Two results are relevant to the present study. First, although young 3-year-olds (those younger than 42 months) required more button presses to guide the cursor the target window when using the directional control interface, there was no significant difference between the older and younger 3-year-olds in accuracy of placement. In other words, younger children had more difficulty guiding the cursor to its target in the directional condition, but all eventually did so. However, errors in cursor placement were actually higher when using the nondirectional interface. The reason for this was that children using the nondirectional interface were impatient with having to move the cursor through all the intermediate choices when trying to cycle the cursor to a given target. They quickly adopted a strategy of rapidly pressing the cross-hair button in an attempt to move the cursor swiftly through all the choices and pressing the red button when the cursor appeared to be moving into place. The result was a consistent pattern of selecting the wrong target, because the children's estimations of the cursor's location were frequently in error.

The results of Strommen et al. indicate that the directional control features of the NES interface are not the source of young children's difficulty with the Nintendo controller: The children were more accurate with the experimental directional interface than with the nondirectional design. What other features of the standard NES interface could be the source of young children's problems? Consider the differences between the standard NES interface and the experimental interface used in Strommen et al. (1992). Because the issue being studied was young children's ability to use a directional controller, the interface in that study was radically simplified to remove confounding features. For example, unlike standard NES directional interfaces, the directional condition task did not require children to actually stop a moving cursor on a target. There were only four discrete cursor positions available;

children simply selected the direction of movement, and the cursor automatically stopped on the target. To select the target, they simply had to press one of the red buttons on the controller. This is strikingly different from the typical Nintendo game design, in which the NES controller moves a cursor through an open field with no discrete positions. The cursor, usually in the form of a small human figure, must be positioned carefully in order to select an item. In addition, in most Nintendo applications, the red buttons do not produce error-free selection of an item. Rather, they make the character jump to "catch" onscreen objects, which can be either stationary or moving. So the character can "miss" an object when its jump has been incorrectly executed.

This comparison reveals two additional functional demands of the Nintendo interface: the need to use directional input to accurately position the character/cursor on the screen and the need to execute appropriate jumps by the character in order to capture objects onscreen. These interface features require significantly more cognitive effort than the simplified interfaces described earlier. Are these demands a source of children's difficulty with the NES interface, or are these demands mediated by the type of cursor movement the NES supports? The major difference between the experimental interfaces tested in Strommen et al. (1992) and the standard NES interface is that movement was discrete, or stepped, in both experimental interfaces but is continuous in most NES games. Continuous movement makes extra demands on the user in that it requires constant monitoring of cursor position. The cursor glides smoothly across the screen, and the user must continuously check its position so it can be stopped or its trajectory modified. Discrete movement, in contrast, requires no such effort: Each step of discrete movement creates a time-limited, fixed-length change in cursor position that ends with the cursor being stationary. Thus, the cursor's position can be evaluated while it is not in motion, awaiting the next movement input.

This study investigates young children's ability to use a standard NES interface, when movement is either discrete or continuous. Two interfaces that mimic the standard Nintendo design were created, differing only in the form of cursor motion they supported. In the *continuous movement* condition, the character/cursor moves in a smooth motion in whichever direction the cross-hair button is pressed; in the *discrete movement* condition, movement of the character is stepped so that distinct button presses are required to advance it across the screen. How children performed with these interfaces, and what their performance reveals about their cognitive abilities and limitations, are described later.

2. METHOD

Participants. Thirty-six 3-year-old children, equal numbers of boys and girls, were drawn from local preschools as the sample in the present study

(mean age = 42.83 months, range = 37 to 48 months). Samples were divided by age according to a median split such that there were equal numbers of younger 3-year-olds (37 to 42 months) and older 3-year-olds (43 to 48 months) in each condition. Children were matched for age and sex across the two conditions. Approximately two thirds of the children in each condition had previous experience with either the Nintendo system or with a computer, according to parental reports.

Materials. A standard NES controller was modified to operate with an IBM computer. The right and left legs of the cross-hair button moved the character right or left when pressed; pressing the up or down arrow had no effect. The two smaller red buttons both caused the character to jump into the air when pressed. The software used for testing was a simple mock-up of a Nintendo-like game. In this game, a Sesame Street character, The Count, was moved by the controller. The Count walked left or right along the bottom of the screen, and above the Count's head, in regularly spaced positions, numbers appeared for the Count to "catch" by jumping. The Count signaled that the numbers were within his jumping range by raising his head and looking up at the numbers when he walked beneath them. There were two types of numbers intermixed throughout the game: *stationary* and *moving*. Stationary numbers simply appeared above the Count's head. Moving numbers followed a horizontal left–right path above the Count that was approximately three times the width of the Count, and catching them required coordinating the Count's jump with the number's position during movement. Twenty of each number type were included in the game. The goal of the game was to capture as many numbers as possible by using the cross-hair button to position the Count to stand beneath each one and then using the red buttons to make him jump to capture it.

Children were tested on one of two types of interface using this controller: *continuous* or *discrete* movement. The continuous condition is typical of almost all NES products. Pressing the left or right legs of the cross-hair button caused the Count to walk in the direction indicated as long as the button was pressed. When the button was released, the Count stopped walking. To catch a number by pressing a red button, children had to specifically place the Count where he would look up, signaling he could catch the number. The Count could be positioned anywhere, including out of reach of the numbers. Pressing a red button when the Count was standing under a number caused him to jump and catch it, resulting in a musical payoff. Pressing a red button when the Count was not under a number simply caused him to leap into the air. In the discrete motion condition, pressing the left or right legs of the cross-hair button caused the Count to hop one quarter of the screen length in the direction indicated. The cross-hair button had to be released and then pressed again to make the Count jump to a new location. If a number was

present at the stopping location, the Count always stopped directly beneath stationary numbers or in the middle of the path of moving numbers, and he always looked up to indicate he could catch the number.

Procedure. The testing protocol was identical for both conditions. Children played the game individually, in a room near their classrooms. The children in the discrete movement condition were shown how to use the controller, with the following instructions: "See that little man on the TV? This button makes him move. Watch. When I press this button [right arrow] he hops toward me! When I press this button [left arrow] he hops toward you! You try it!" The children were then shown how the Count was to catch numbers: "See how the Count is standing under that number, and looking up at it? He wants to catch it! To catch the number, you press a red button when the Count is looking up at a number. Try it!"

The children in the continuous condition were instructed as follows: "See that little man on the TV? This button makes him walk. When I press this button, he walks toward me! When I press this button he walks toward you. See how he keeps walking, as long as I press the button? To make him stop, I have to let go of the button! Watch! Now you try it!" In describing how to catch numbers, the children were told, "See how the Count is looking up at that number? He wants to catch it! You have to stop him when he is looking up at the number, and then press a red button to make him jump to catch it!" The red button was explained in the same manner as before.

Regardless of the condition, children had to demonstrate the ability to move the Count left and right on demand and to make him jump in order to actually play the game. The children then caught as many numbers as they could out of a possible maximum of 40.

3. RESULTS

Overview of Analysis. Performance on the experimental task followed a simple pattern. First, children had to position the Count beneath the number they wished to capture using the cross-hair button. Then, they had to capture the number by pressing one of the red buttons to make him jump. These steps in playing the game (positioning and capturing) were influenced both by interface type and the type of number (moving or stationary) being captured. The analysis provided next follows this sequence, and differences due to interface and number type, where significant, are noted. The goal is to characterize children's performance and to identify specific influences on it.

The basic unit of analysis was the children's behavior when catching numbers during the session. The data were aggregated in two different ways for statistical analysis. First, scores were summarized within children for moving and stationary numbers separately and submitted to 2 (Interface

Type) × 2 (Median Age Split) × 2 (Sex) × 2 (Number Type: Moving vs. Stationary) analyses of variance (ANOVAs), where number type was a within-subject variable and the rest were between-subjects variables. This type of analysis is referred to hereafter as the mixed-model ANOVA. If the variable was not one that would be affected by number type, or if the results of the first analysis did not yield a significant effect for number type, the variables were reanalyzed using simple 2 (Interface Type) × 2 (Median Age Split) × 2 (Sex) ANOVAs. This type of analysis is referred to hereafter as the independent-model ANOVA. Because children's experience was collected as a post hoc variable, the effects of experience were analyzed separately using *t* tests, and experience was not included as a variable in the ANOVA models.

Total Numbers Captured. The total numbers children caught (moving and stationary combined) during game play is a broad index of their competence with the controller and the interface. Independent-model ANOVA results indicated no significant main effects or interactions for any of the independent variables on total numbers captured. However, significant differences between the two interface conditions are apparent when experience is considered as a variable. There is no significant difference between inexperienced and experienced children in the discrete condition, where approximately equal scores were obtained ($M = 34.33$ numbers for inexperienced children, $M = 36.44$ for experienced). In the continuous movement condition, however, substantial differences appear ($M = 35.09$ numbers for experienced children, $M = 22.33$ numbers for inexperienced children), $t(15) = -2.33$, $p < .03$. The discrete movement condition thus appears to facilitate inexperienced children's performance.

Performance When Positioning the Count. To capture a number, the Count must first be placed within the number's active area by using the cross-hair button. In both conditions, it is possible to overshoot, or move the Count past the number to be captured. In the discrete condition, an extra button press sends the Count to the next location on the screen, clearly away from the target. In the continuous condition, either an extra button press or too long a button press can cause the Count to move past or off the target number by both very short and very long distances. The total moving and stationary numbers that were overshot were tallied separately and divided by the total number of each number type that was captured to yield a percent score indicating the proportion of each number type overshot by each child.

A mixed-model ANOVA on the percent overshot scores indicated main effects for interface type, $F(1, 28) = 10.99$, $p < .003$, and for type of number, $F(1, 28) = 7.60$, $p < .01$, and a significant interaction between the two, $F(1, 28) = 6.39$, $p < .01$. The main effect for interface type clearly

indicates that the continuous interface gives rise to more incidents of overshooting (M = 16% of trials vs. M = 4% in the discrete condition). The effect for number type is due to the fact that stationary numbers elicit more overshooting than do moving numbers (M = 14% of stationary numbers captured, M = 6% of moving numbers). The interaction, however, reveals the key finding. The percent of moving numbers overshot does not differ significantly between the two conditions (M = 8% of moving objects in the continuous condition, M = 3% of trials in the discrete condition). However, children overshot stationary numbers on M = 24% of trials in the continuous condition but only M = 4% of trials in the discrete condition.

The lack of a significant difference between interfaces for the moving numbers is apparently due to the simple fact that, because of their motion, the moving numbers allow for a larger margin of error when children are positioning the Count to capture them. The number will actually come to the Count at his position as long as he is within the range of the path of the number's left-right movement; this makes moving numbers easier for children to capture because if they make an error in stopping the Count, they nonetheless still have him end up in a position where he can jump and capture the number as it moves.

Strategies for Positioning the Count in the Continuous Condition. One interesting finding in the present study is the difference between the two conditions in terms of the methods children use when positioning the Count. In the discrete condition, no real method is required: A button press moves the Count directly beneath the stationary numbers and directly to the middle of the path of the moving numbers. The continuous condition presents a different picture. To position the Count, the child moves the Count until he is under the closest number and then stops him at that point. Two distinct methods of achieving this performance in the continuous condition were identified. The first, *creeping,* is defined by a series of specific steps. First, there is an initial long button press that brings the Count within view of the number to be captured. At this point, the child releases the button, stopping the Count, and then presses the button in a series of short, discrete movements that bring the Count bit-by-bit into the active area of the number. When the Count looks up, the child stops moving him and presses the red button. The second strategy, *sliding,* involves just one long button press. In sliding, the child moves the Count by holding down the button and monitoring his progress toward the target number. The moment he looks up at the number, the child either releases the button to stop him and then presses the red button or simply presses the red button while he is walking, causing the Count to jump while still moving and capture the number on his way by. Children were scored for the use of these strategies for each number they

attempted to capture. The number of trials during which each strategy was used was then divided by the total number of trials to create a percent score indicating the relative use of each strategy.

A 2 (Sex) × 2 (Median Age Split) × 2 (Creeping Score vs. Sliding Score) ANOVA revealed a significant effect only for the difference between the two scores, $F(1, 14) = 7.94$, $p < .014$. Children clearly appear to prefer sliding as a strategy, using it on 68% of trials versus 32% for creeping. However, although sliding is the more popular strategy, there is actually a shift in the frequency of how often it is used as game play progresses. It appears that creeping is a strategy that is deployed during initial play, whereas sliding is deployed as children become familiar with the interface and task. To test this hypothesis, the ordinal position of each number was correlated with the percent of children using the creeping or moving strategy of that number. Results indicate a significant negative correlation between the place of the number and creeping, $r(40) = -.48$, $p < .002$, and a significant positive correlation between the place of the number and sliding, $r(40) = .43$, $p < .006$. As the children capture more numbers, the number of children creeping decreases and the number of children sliding increases. No relationships were observed between the particular strategy children used on different trials and the frequency of overshooting, total numbers captured, or experience.

Capturing the Number. Regardless of whether the numbers are stationary or moving, positioning the Count is key to successfully capturing numbers. As indicated before, despite the presence of the clear feedback of the Count's looking up when his position is right, children in the present study still overshot the target, especially stationary numbers in the continuous condition. Often, this overshooting was not detected until the child actually tried to capture the number. The proportion of trials during which children tried to capture a number when the Count was not positioned correctly was calculated for each type of number. A mixed-model ANOVA reveals significant effects for interface type, $F(1, 28) = 28.35$, $p < .0001$, and for type of number, $F(1, 28) = 12.96$, $p < .001$, and a significant Interface × Number Type interaction, $F(1, 28) = 12.90$, $p < .001$. The significant effect for interface type reflects the fact that children overshot and tried to capture numbers in the continuous condition significantly more than in the discrete condition ($M = 26\%$ of trials vs. $M = 8\%$ of trials, respectively). The significant effect for number type reveals that it is the stationary numbers that are more frequently jumped at when the Count is in the wrong position ($M = 22\%$ of stationary numbers, $M = 12\%$ of moving ones). The significant interaction reveals that although the discrete condition gave rise to equal numbers of erroneous captures for both number types ($M = 8\%$ of both moving and stationary numbers), there was a substantial difference for the continuous condition ($M = 15\%$ of moving numbers, $M = 36\%$ of

stationary numbers). The relationship between overshooting and jumping at the wrong time is confirmed by the substantial correlation between these variables for both number types, $r(36) = .51, p < .002$, for moving numbers and, $r(36) = .60, p < .0001$ for stationary ones.

The performance demands for the two types of numbers differed depending on the interface used. For stationary numbers, the only issue is that of overshooting. In both conditions, once the Count is in the active area, all the child need do is make him jump to capture the number. Moving numbers, however, present an extra challenge in both interface conditions that is over and above the positioning issue. They can be captured only by timing one's jumps so that the Count intercepts the number at the right time during its left-to-right trajectory over his head. The center of the trajectory of the number is the most efficient place from which to jump in the continuous condition, although the Count can be placed anywhere within the trajectory's width; he is automatically positioned in the center of the trajectory in the discrete condition.

One measure of the ease of number capture is the average number of extra jumps required. Correctly positioned, the Count requires only one jump (at the right time with moving numbers) to capture any number. A mixed-model ANOVA of average extra jumps showed that children are challenged by the need to coordinate the Count's jump with the position of the moving numbers. The only significant effect is that of number type, $F(1, 28) = 50.13, p < .0001$ ($M = 0.48$ extra jumps per moving number, none for stationary ones).

4. DISCUSSION

The results of the present study suggest a seemingly confusing portrait of young children's competence. There is no difference between the two interface conditions in terms of total numbers captured, yet the data clearly indicate that controlling continuous movement is more difficult for children than controlling discrete movement. And the moving numbers, which theoretically should be more difficult for children to capture, actually appear to be easier to capture than the stationary numbers. Understanding the current findings requires an analysis of both the performance demands of continuous movement as an interface and the motivational context (in this case, a video game) in which a given interface is placed.

Children using the continuous movement interface exhibited higher levels of several behaviors that should have resulted in poor performance on the present task. Their key performance problem, however, was overshooting—positioning the cursor too far to the left or right of the target numbers, especially the stationary ones, so they could not be captured. They not only positioned the character incorrectly but also failed to notice they were not in

the correct position until they had tried and failed to capture the target number. The superior performance of children with the moving numbers is due largely to the fact that the number's motion alleviated the need for precise positioning. The number's motion meant that it often "intercepted" the character as it jumped, so even an inexact placement of the cursor could still result in success; overshooting had less of an impact on performance because no repositioning was necessary. For the stationary numbers, in contrast, even a small error in positioning the character meant failure. The character had to be carefully repositioned to catch the number. Continuous movement thus presented significant difficulties for young children in terms of cursor control, but only when precise positioning of the cursor was required.

Preschoolers' performance in the discrete movement condition, in contrast, was virtually error free. The need to position the character with precision was completely eliminated, so capturing stationary numbers was easy: The character was always perfectly located to do so, and it was also positioned in the center of the path of the moving numbers, so ease in capturing them was also optimized. The data indicate that these advantages produce significantly fewer of the previously mentioned problems found in the continuous movement condition, especially overshooting. There was very little overshooting of the target using discrete steps.

The explanation for the differences in interface performance lies in a combination of both human–computer interaction research and research on the development of cognition. Adult human–computer interaction has been described very effectively as a product of the coordination of a hierarchy of task-related goals and the methods necessary to accomplish them (cf. Booth, 1989; Olson & Olson, 1990). A key assumption of these theories is that adult working memory capacity is sufficient to allow for the various components of behavior required for each task to be recalled and executed, so that sophisticated tasks can be performed in an organized manner. A large amount of data has established that children's working memory capacity is much smaller than that of adults, and recent theories of cognitive development have suggested that the growth of short-term memory, variously identified as "short-term storage space" (Case, 1985), "working memory" (Foreman, Warry, & Murray, 1990), or "mental capacity" (Johnson & Pascual-Leone, 1989), is a key factor in cognitive growth. It is argued that children's memory capacity, especially that of preschoolers, is simply too small to allow them to activate all the schemata associated with successfully performing a variety of tasks.

The present results regarding cursor control can be understood by relating this research to the task analysis suggested by current models of human–computer interaction. Strommen (1993) has suggested that cursor control is the product of a set of executive schemata that must be coordinated in working memory. Control of a continuous motion cursor can be under-

stood as the product of three general schemata: (a) choosing a target position for the cursor and executing the appropriate methods for moving the cursor there (i.e., choosing a direction, pressing the appropriate button to move the cursor in that direction, etc.); (b) monitoring the cursor's progress toward that target, which involves continuously checking the cursor's speed, direction, and relative position; and (c) stopping the cursor by executing the appropriate method (in this case, releasing a button) when Schema b indicates the target has been reached.

Young children's working memory generally allows them to access only a single schema at a time; that is, their working memory is insufficient to allow for two or more of them to be active at the same time (Case, 1985). Overshooting occurs because children can "load," or recall, the stopping schema from long-term memory only at the point where the monitoring schema indicates that the cursor has met the target. Exchanging the monitoring and stopping schemata right when the cursor and target meet results in the monitoring/moving schema remaining active while the stopping schema is activated, and the cursor is thus moved past the target. As the results indicate, children's attempts to "repair" their overshoots are often overshot themselves, for the same reason: The child is constantly swapping the schemata in active memory, and the delay between ceasing the active schema and engaging the new one allows for repeated errors. Under this explanation, discrete movement is easier for children because it removes the need to dynamically exchange the monitoring and stopping schemata during cursor placement. The cursor stops itself; the child need never invoke the extra schema. The only task confronting children in the discrete condition is to monitor cursor position until it stops in a place that allows for the capture of target numbers.

The hypothesis that discrete movement is easier because the child is not required to exchange the monitoring schema for the stopping schema to control the cursor in motion receives some support from two other results in the present study: (a) the superior performance of inexperienced users (but not experienced users) when using discrete movement and (b) the tendency of children in the continuous movement condition to switch from creeping to sliding as a strategy for stopping the character/cursor under the target numbers. Working memory research has shown that experience lessens the amount of memory required to activate cognitive schemata because with practice, both adults and children "chunk," or consolidate, originally distinct schemata into a single schema that requires less memory space than the original, separate ones. Discrete movement is especially helpful for novice users because they have not yet had enough practice to chunk their cursor control schemata. By removing the need to invoke the stopping schema, discrete movement reduces the working memory space required for cursor control and thus simplifies the task new users have to master.

The transition from creeping to sliding as a strategy for continuous

movement can also be viewed as evidence of the evolution of chunking during task execution. Recall that creeping involves children moving the cursor toward the target in a series of discrete steps; each step is executed, and its results monitored, before the next step is taken. In effect, the children convert the continuous movement into discrete movement to assure themselves of accurate cursor placement. But, as they become experienced with the task, they change their strategy to sliding, or simply moving the character along until it reaches the appropriate position, and then stopping it abruptly under the target. This performance strongly suggests that the monitoring and stopping schemata have become chunked in the course of the activity itself and have come to be executed as a single cognitive act.

The analysis of the present results in terms of working memory and executive schemata suggests why discrete movement is easier for young children than continuous movement. It does not, however, explain why, despite the performance problems observed in the present study for continuous movement, there is no difference between the two types of movement in total numbers captured. The lack of difference is due to the fact that, in the continuous condition, children persisted in trying repeatedly to capture a given number until they did so, despite all the problematic behaviors reported earlier that made such a goal difficult. The explanation for this result lies in the motivational context of the task.

Had children been performing a standard laboratory task with a computer, it is not clear that they would have persisted in the face of the performance problems they encountered using the continuous movement interface. The present task, however, was designed to have the look and feel of a Nintendo game. The children responded to the task and to handling the Nintendo controller itself with a surprising degree of concentration and effort. They were "trying to win," or competing, and capturing the numbers appeared to give them a sense of accomplishment. Children's spontaneous declarations were frequently in the form of self-congratulatory comments ("Yes! Alright!") and gestures, and these declarations were clearly more frequent when children were using the continuous rather than the discrete movement interface.

In conclusion, it appears that the developmental status of the user is an important factor in the design of computer interfaces. Young children experience notable difficulties when controlling continuous movement interfaces, difficulties that are not apparent with a discrete movement interface used for the same task. A possible explanation of the cognitive processes that produce preschoolers' performance problems was presented. Future studies are needed to experimentally validate this model, but the current results clearly suggest that, regardless of the precise explanation, there are important limitations on young children's cognitive abilities that can adversely affect their use of common interface designs. The present results suggest that children compensate for these limitations with practice, but whether there is

a ceiling to the improvement in their performance over time that is related to fundamental cognitive developmental issues should be investigated.

It is also apparent that the performance problems engendered by a given interface can be compensated for by providing a context that motivates children to persist in accomplishing the specific task they are attempting. If the task were a standard preschool computer-learning activity, like matching lowercase and uppercase letters, and the children were supposed to "catch" the answer, it is not clear that the children would have persisted as they did in the present task. The Nintendo-like design of the present task provided children with a motivational situation that led them to exert serious effort in overcoming the performance problems presented by the continuous movement interface. Future studies should determine if there is a relationship between interface difficulty and motivational context and if there is a ceiling to such effects. It is likely that an interface can be too taxing, even for a highly motivating context to overcome. However, it may also be true that, given technical constraints on interface functionality, a less than optimal interface could be made more acceptable by embedding it in an entertaining or other metaphor that sustains user effort.

Technology designed for children, especially preschoolers, must satisfy many additional challenges that do not apply to technology designed for adults. By studying children's competence, not just with different types of cursor control but also with different graphic designs and task structures, it may be possible to invent new and more creative ways to allow adults to use technology more effectively. In the end, by widening our concept of "the typical user" to include the youngest among us, we may make some surprising and valuable discoveries about the development of technologies for adults that provide insights not only into human–computer interaction but also into the nature of cognition itself.

REFERENCES

Booth, P. (1989). *An introduction to human–computer interaction.* Hillsdale, NJ: Lawrence Erlbaum Associates, Inc.

Case, R. (1985). *Intellectual development: Birth to adulthood.* New York: Academic.

Children's Computer Workshop. (1983). *Cookie monster munch* [Atari Game cartridge]. Sunnyvale, CA: Atari.

Children's Television Workshop. (1987). *Big Bird's special delivery* [Computer program]. New York: Hi Tech Expressions.

Children's Television Workshop. (1988). *Sesame Street 123* [Nintendo game cartridge]. New York: Hi Tech Expressions.

Children's Television Workshop. (1989). *Sesame Street ABC* [Nintendo game cartridge]. New York: Hi Tech Expressions.

Children's Television Workshop. (1990). *Big Bird's hide and speak* [Nintendo game cartridge]. New York: Hi Tech Expressions.

Children's Television Workshop. (1991). *Sesame Street countdown* [Nintendo game cartridge]. New York: Hi Tech Expressions.

Foreman, N., Warry, R., & Murray, P. (1990). Development of reference and working spatial memory in preschool children. *Journal of General Psychology, 117,* 267–276.

Grover, S. C. (1986). A field study of the use of cognitive-developmental principles in microcomputer design for young children. *Journal of Educational Research, 79,* 325–332.

Johnson, J., & Pascual-Leone, J. (1989). Developmental levels of processing in metaphor interpretation. *Journal of Experimental Child Psychology, 48,* 1–31.

Olson, J. R., & Olson, G. M. (1990). The growth of cognitive modeling in human-computer interaction since GOMS. *Human-Computer Interaction, 5,* 221–265.

Revelle, G. L., & Strommen, E. F. (1990). The effects of practice and input device used on young children's computer control. *Journal of Computing in Childhood Education, 2*(1), 33–41.

Revelle, G. L., Strommen, E. F., & Offerman, S. C. (1990). *The effects of cursor and device differences on children's computer control.* Unpublished manuscript.

Rice, B. A. (1987). *The Sesame Street crayon: Letters for you* [Computer program]. Geneva, IL: Merit Software.

Strommen, E. F. (1993). Preschoolers at the interface: A cognitive model of device difficulty. In M. R. Simonson & K. Abu-Omar (Eds.), *15th Annual Proceedings of Selected Research and Development Presentations, AECT '93* (pp. 991–1000). Washington, DC: Association for Educational Communications and Technology.

Strommen, E. F., Razavi, S., & Medoff, L. M. (1992). This button makes you go up: Three year olds and the Nintendo controller. *Applied Ergonomics, 23,* 409–413.

HCI Editorial Record. First manuscript received March 23, 1992. Revision received May 2, 1993. Accepted by Elliot Soloway. Final manuscript received June 21, 1993. — *Editor.*