Scaffolding as a Strategy to Make Tools More Accessible to Young Children

Hai-Ning Liang and Kamran Sedig
hliang@uwo.ca & sedig@uwo.ca
The University of Western Ontario, CANADA

Abstract. Designing educational software that both supports children's learning of complex concepts and is accessible to them is not easy. One strategy to address this challenge is to create increasingly more complex environments that incorporate different scaffold structures. These structures serve to support children in learning how to use more powerful and sophisticated interface components, and, at the same time, assist them in learning concepts that they need when moved to an environment of higher complexity. This paper reports a study that investigated if interface scaffolds can make a tool more accessible and its embedded concepts more learnable to very young children. To conduct the study, a tool with multi-level environments that allows children to create tiling patterns using procedural code was designed and used. The findings seem to suggest that scaffolds can make a complex environment and its content more accessible, even to very young children.

Introduction and Background

Designing computer-based interactive learning tools that are both supportive of children's learning of new and complex concepts and are also accessible to them is not easy (Druin 1999; Alessi & Trollip 2001; Sedig et al. 2001). This is especially the case if the intended users are 6-year-old children. When introduced to a new interactive tool, there may exist a distance between the tool and children. This distance between children and the tool can happen at two levels: interface and content. At the interface level, children might not understand how to interact with the elements and components of the interface. At the content level, the concepts which children aim to explore and learn might be too complex and unfamiliar. There are two ways to narrow this distance. At one end, children, through conscious effort, try to move closer to the tool; and, at the other end, the tool, through proper design, adapts to the needs of children and progressively increases their level of understanding. In learning tools, it might not be pedagogically sound to situate the design too close to either end. The challenge is to find the balance between requiring children to spend some degree of cognitive effort, and, at the same time, having the tool provide supporting structures to complement this effort. In this way, children will not experience either anxiety—when too much effort is needed—or boredom—when too little effort is needed. One strategy for achieving this is through a scaffolding process.

Scaffolding refers to the process of providing support structures, or scaffolds, so that children can engage in new, complex activities (Masterman & Rogers 2002; Quintana et al. 2002; Reiser 2004; Puntambekar & Hübscher 2005; Sedig & Liang 2006). Providing scaffolds can make these activities doable and accessible to children; otherwise, the activities will lie beyond their abilities. Scaffolding can be used to facilitate the acquisition of and reasoning with abstract concepts, and encourage higher-order and reflective thinking (Sedig et al. 2001; Masterman & Rogers 2002; Reiser 2004). Scaffolding has two complementary goals. The first goal is to provide support for learners to carry out specific activities so that, once they have mastered them, they can perform them independently. The second goal derives from the first one and is to help learners learn from their own efforts; that is, learners acquire and have mastery over skills which are needed to perform new and more complex activities. One way of achieving these two goals is to change the scaffolds during the progressive stages of interaction with a tool. Previous scaffolds progressively fade away and are replaced by new ones in the course of this interaction. This is intended to involve learners in a constant reflective mode and deeper processing of information.

Programming is a difficult concept for children to learn (Druin 1999; Guzdial 2004). There are however many general benefits children can derive from learning how to program. It can help children to acquire logical and abstract thinking skills, which can then enhance their ability to solve problems in other areas; it can also assist children to be able to articulate, explicate, and sequentialize their own thinking; it may lead children to develop higher-order and self-corrective thinking and learning skills, such as debugging; and it may empower children to create interactive simulations, toys, games, and artistic geometric patterns (Papert, 1980; Brusilovsky et al. 1997; Canfield, & Cypher 1999; Druin, 1999; Guzdial 2004; Fernaeus et al. 2006). However, many children find learning programming and algorithmic thinking to be 'hard' (Brusilovsky et al. 1997; Guzdial 2004; Kelleher & Pausch 2005). Many interactive learning tools intended to support learning of programming may not be accessible to learners, especially young children. Scaffolding can be used to make these tools more accessible and its embedded concepts more learnable. This article reports some general findings of a study that investigated whether scaffolding can make a tool, designed to support young children in learning how to write sequential procedural code, more accessible.

The reminder of this article is structured as follows. The next section introduces a tool, TileLand, that was used in the study. Afterwards, the study that was conducted and its research methodology are described. Next, the results of the study are presented. And, finally, in the last section, a brief summary and conclusions are presented.

TileLand

TileLand was originally conceptualized and created as an interactive learning tool to enable children to construct colorful, polygon-based tiling patterns by writing procedural code (see Sedig et al. 2002). Since then TileLand has gone through several modifications and several extensions have been added to it. TileLand has been redesigned to make it more accessible for children aged 6 and up. Its interaction rules are based on the principles of the mini-language paradigm—a "powerful way to introduce students to programming" (Brusilovsky et al. 1997, p. 65). As such, to interact with the tool, children only need to learn a small set of commands with simple syntax and semantics. Several commands have been either removed or simplified and made into a single command. This is intended to minimize the time needed to learn the language of interaction so that children can concentrate on the learning of its embedded concepts—i.e., programming through pattern creation. TileLand is a composite of several increasingly complex microworlds or environments. Each new environment in the sequence introduces different aspects of programming to children and also provides more powerful capabilities so that children can develop more sophisticated patterns. Scaffolds are introduced in each environment to support children in learning the needed programming concepts for the subsequent environments. These scaffolds fade away so that children are encouraged to perform activities with a greater degree of independence. Each environment is considered as a level; seven different levels are provided in the redesigned TileLand. Before describing each level, a general overview of TileLand is provided next.

The interface of TileLand has three main components: Action Panel, Command Panel, and Pattern Area (see Figure 3). Depending on the level, all three components or two of them are available to children along with other transitional scaffold structures. The Action Panel contains all the command buttons by which children can construct tiling patterns. These buttons are grouped according to their functions—e.g., tile selection, color selection, etc. The Command Panel contains linguistic representations of commands, displayed in chronological order. As children progress through the different environments, they will be issuing text commands typed into this panel to create patterns rather than using the Action Panel buttons. Lastly, the Pattern Area displays the patterns that children create. Patterns can be created by placing, connecting, and coloring a number of tiles on the Pattern Area. There are five available types of tiles, all of which are regular polygons: triangle, square, pentagon, hexagon, and octagon.

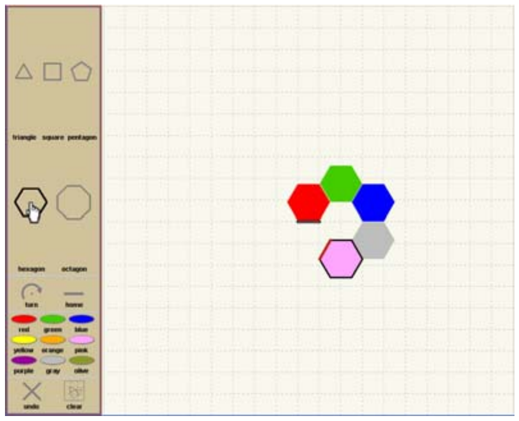
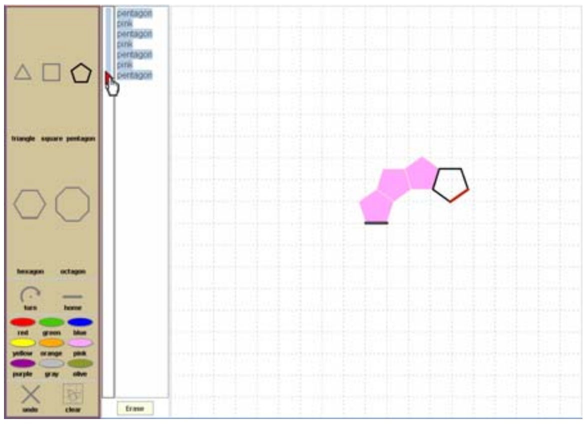
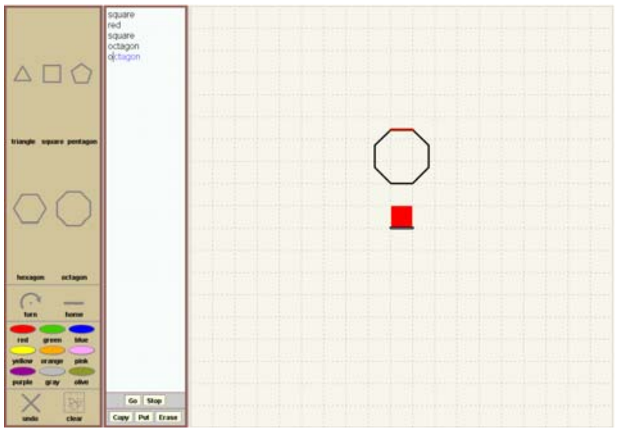


Figure 1: Level 1



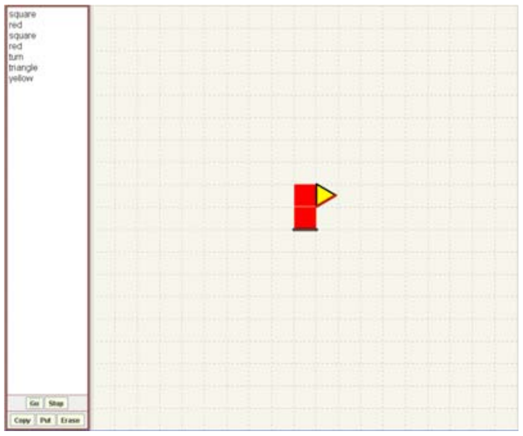
```
pentagon
pink
pentagon
pink
pentagon
pink
pentagon
```

Figure 2: Level 2



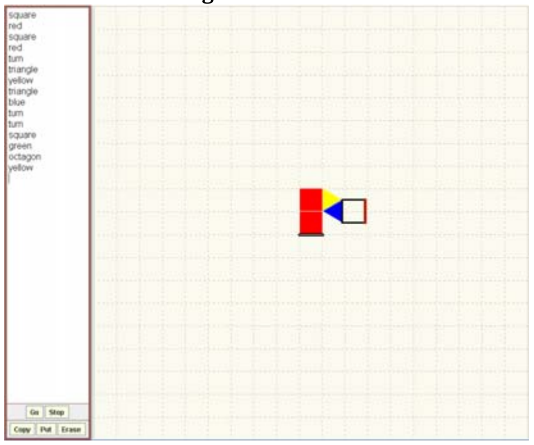
```
square
red
square
octagon
octagon
```

Figure 3: Level 3



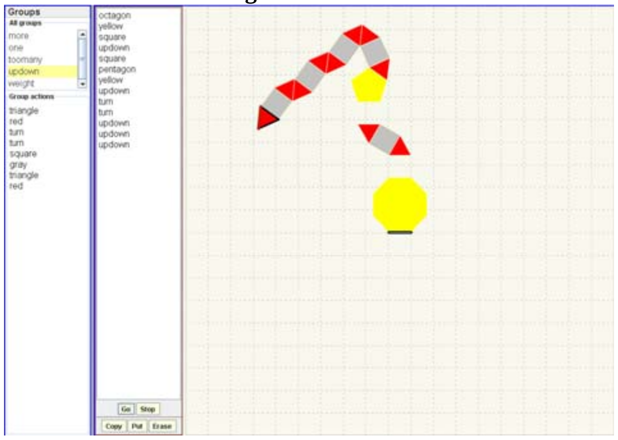
```
square
red
square
red
turn
triangle
yellow
```

Figure 4: Level 4



```
square
red
square
red
turn
triangle
yellow
triangle
blue
turn
turn
square
green
octagon
yellow
```

Figure 5: Level 5



All groups:

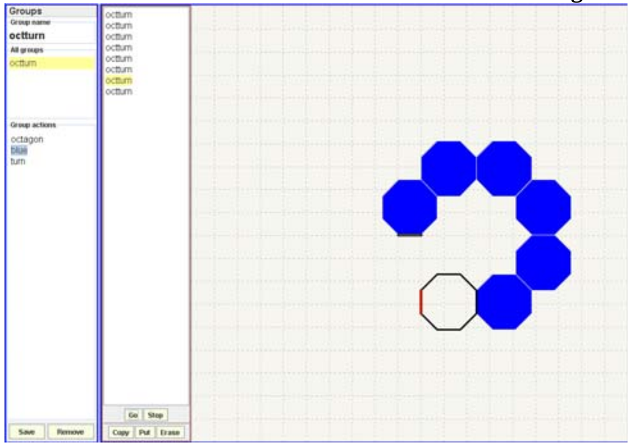
```
more
one
toomany
updown
weight
```

Group actions:

```
triangle
red
turn
turn
square
gray
triangle
red
```

```
octagon
yellow
square
updown
square
pentagon
yellow
updown
turn
turn
updown
updown
updown
```

Figure 6: Level 6



Group name: octturn

All groups:

```
octturn
```

Group actions:

```
octagon
blue
turn
```

```
octturn
octturn
octturn
octturn
octturn
octturn
octturn
octturn
```

Figure 7: Level 7

Both the original and redesigned versions of TileLand employ the conceptual metaphor of a Magic Tile to help children understand how to create tiling patterns. Only one Magic Tile is present in the Pattern Area, and it is represented as the outline of any of the five polygons. One of the sides of the Magic Tile, the Connecting Side, is highlighted and serves to indicate where the next Magic Tile connects. That is, when a new polygon is selected (e.g., by clicking on a polygonal button from the Action Panel), the previous Magic Tile disappears, and a new one appears next to the Connecting Side and takes the outline of the selected polygon. To simplify and reduce the number of conceptual commands, children can only turn the Connecting Side clockwise by clicking on the Turn button[1]. Finally, the Magic Tile can be filled with a color or left blank. Clicking on any of the color buttons from the Action Panel fills the Magic Tile, making a *footprint* on the Pattern Area. Therefore, children construct patterns by leaving footprints behind as they 'walk around' the Pattern Area. TileLand does not allow overlapping footprints. If the Magic Tile is filled with a color while it is on top of existing footprints, the underlying footprints disappear and are replaced by the new footprint. Thus, knowing these few simple rules—i.e., how to walk around the Pattern Area and make footprints—will allow children to interact with TileLand to create patterns of diverse complexity.

In addition to the above rules, TileLand has a few other complementary features: 1) Undo, which allows children to undo their previous actions; 2) Clear, which allows children to clear any footprints on the Pattern Area; and 3) Home, which allows children to reset the location of the Magic Tile and return to the center of the Pattern Area[2].

Level 1

Level 1 provides children with two components: Action Panel and Pattern Area (see Figure 1). Children's construction of patterns is only through the visual buttons on the Action Panel. Clicking on any button has an immediate visual feedback on the Pattern Area—i.e., the action is displayed in the Pattern Area without any time lag. This level is purely exploratory and is intended to introduce children to the simple rules of pattern creation described above. Children can create simple and elaborate patterns without much cognitive effort and any understanding of programming concepts. This level is intended to generate interest and motivation from children and to situate learning of programming within a context that is interesting and fun.

Level 2

Level 2 is similar to level 1, but with one additional component: *History Slider* (see Figure 2). The History Slider is a scaffold component and is designed to introduce children to a few programming-related concepts: text commands, code writing, sequentialization of code, and self-checking and self-correcting practices (i.e., debugging). The History Slider codifies the mouse-click actions children perform and sequentially displays them as text commands. This sequential list represents the history of children's interactions. The History Slider does not allow children to make modifications to the commands; but it enables them to 'walk' through the list, command by command. Dragging its interactive control upward unselects the commands below the triangle and removes the corresponding actions from the Pattern Area. Similarly, dragging the interactive control downward performs the opposite operation; that is, it selects the commands and performs the actions on the Pattern Area. Consequently, the History Slider allows children to explore the effect each command has on the pattern in a stepwise manner and at their own pace. The possibility of interactively reviewing the history of their actions can help children confirm their hypotheses, locate misplaced commands, and revise and rectify past commands. The History Slider serves as a transitional structure to lead children into the Command Panel, which becomes available in Level 3.

Level 3

In Level 3, the History Slider is replaced by the Command Panel (see Figure 3). Children can now create patterns by either using the buttons on the Action Panel or typing text commands into the Command Panel. The buttons work the same way as in the previous two levels, but with one difference. When a button is clicked, its corresponding text representation is also displayed in the Command Panel. The Command Panel behaves like a text editor. However, unlike the History Slider, children can now go into the sequence of commands and make modifications to it. To minimize the amount of typing and prevent children from making typos, the Command Panel has a built-in autofilling mechanism[3]. That is, when children enter the first letter of a command, the panel completes the remaining letters. To accept the command, children need to only press the Enter key; or if they wish, they can still type all the remaining letters, and press Enter to accept the command[4]. Once children press Enter, the action is immediately performed on the Pattern Area.

The Command Panel has four buttons: *Go*, *Stop*, *Copy*, *Put*, and *Erase*. The Go button allows children to run an animation, which shows the sequential execution of the commands that are in the panel. To emphasize the

[1] Earlier versions had both TurnLeft and TurnRight commands. From informal observations of children using the tool, it was found that children were often unsure of which direction to turn, spending a lot of time turn back and forth the Connecting Side.
[2] Similarly, from informal observations, we noted that it was easier for very young children to be given a starting position so they did not have to worry about where to begin.
[3] We assume that young children might not be able to type and/or spell correctly. Since the purpose of TileLand is not for supporting children in learning how to type, having this mechanism greatly reduces the need to learn typing.
[4] The Command Panel displays mistyped commands in red to inform children that they need to correct them.

connection between the commands and their effect on the pattern, the animation simultaneously highlights a command and, at the same time, displays its corresponding effect on the pattern. The Stop button enables children to halt an animation. Up until this level, children receive immediate visual feedback for their actions on the Pattern Area. Both Go and Stop buttons are also used here to introduce children to feedback-on-demand. That is, learners do not receive any feedback unless they ask for it (see Level 5). Finally, the Copy and Put buttons make possible for children to copy various commands and paste them in other locations within the panel.

Level 4
In Level 4, the Action Panel is no longer available to children (see Figure 4); thus, leaving them with no other option but to produce purely linguistic commands using the Command Panel to create patterns. The fading away of the Action Panel is intended to wean children from interacting with visual buttons to create patterns. This moves children away from a visual thinking strategy to a linguistic, or command-based, one. Children now need to think in a more algorithmic and procedural manner in order to create patterns. This ability is essential for children if they want to learn programming.

Level 5
There is no change in the components of the interface for Level 5; they remain the same as in Level 4. However, whereas in all previous levels, feedback is both automatically and immediately provided for every command entered, in Level 5, feedback is by demand only. That is, children do not receive any feedback unless they ask for it by pressing the Go button. At early stages of learning, immediate feedback may be more appropriate because it is corrective in nature; it allows children to observe the effect of their actions so that they can either verify their correctness or rectify any mistakes. At later stages, however, non-immediate feedback may be more preferable as it can compel children to engage in more reflective and abstract cognition, hence, higher conceptual learning (see Sedig & Liang 2006 and Alessi & Trollip 2001, p.254, for a more detailed discussions). Therefore, switching from immediate to on-demand, or requested, feedback has a twofold purpose. First, it is to encourage children to mentally visualize longer sequences of commands required in constructing a pattern. And, second, it is to promote revision and detection of errors before committing to the execution of the whole sequence of commands.

Level 6
Level 6 is similar to Level 5, but the Command Panel has been expanded to include the Group Area (see Figure 6). The Group Area contains some predefined groups of commands. It is divided into two sections: top, which shows the names of available groups; and bottom, which displays the commands associated with a group name. Groups are treated like commands, which can be called by typing their names into the Command Panel. When a group is called, all its associated commands are performed. This level is aimed at introducing children to the idea of grouped, chunked, or subroutined actions, an essential concept for performing repetitive procedures. There is a large conceptual gap between Level 5 and this level. Groups may be highly abstract for children as it can be challenging to visualize a set of commands and how its first and last commands connect to other commands (or groups) for creating patterns. However, groups are conceptually identical to methods in some Object Oriented Languages (such as Java) and an efficient mechanism to reuse code. Consequently, knowing how to use groups is important for writing efficient code.

Level 7
Level 7 is similar to Level 6, but now children can create, customize, and use their own groups of commands. A new section within the Group Area enables children to save and remove groups (see Figure 7). In this level, all scaffolds are removed, and children are provided with the full functionality of TileLand to build patterns of varied degrees of complexity. Children who have progressed to this level should have the conceptual skills and know-how to be able to access and use this functionality.

Research Methodology

This section describes the empirical study that was conducted to explore whether incorporating scaffolds into increasingly complex environments can make them more accessible to young children. The study was intended to investigate if scaffolds can assist children in bridging the gap that often exists between children and a new learning environment, hence, making the environment and its content more accessible to children.

Design
A multi-method (quantitative and qualitative) research design was used, including a number of types of data-collection instruments (activities, post test, video transcripts, log files, interviews, and direct observations). The multi-method design was used to help triangulate and cross-validate the different types of findings.

Participants
Twenty-nine children, aged 6 through 12, from an elementary school participated in this study. None of the children had seen or used TileLand before. All children were able to use computers and read simple statements. The study was conducted in the computer room of the school.

Procedure
Children from two classrooms were invited to participate in the study. Each class was comprised of children spanning three grade years: SK to grade three and grades four to seven. Children who wanted to participate were divided into two groups according to age (see Table 1). Both groups would be using TileLand to reproduce a series of sixty-five tiling patterns. Group 1 would be using only the Level 7 environment throughout the study, while Group 2 would be using all seven levels; they would be working with a different environment as they advanced through the patterns. This group was further divided into two groups, again, according to age. Afterwards, children were paired and asked to alternate doing the activities and also talked to each other aloud. Each pair interacted with the tool for five hours: five 60-minute sessions held in consecutive days. At the end of the last session, a post test was administered. After the post test, some children were interviewed to collect direct feedback on their impressions about the tool.

Group		Level(s)	Age Range	# Participants
G1		7	9-12	10
G2	G2.1	1-7	6-8	10
	G2.2	1-7	9-12	9[*]

Table 1: Group distribution

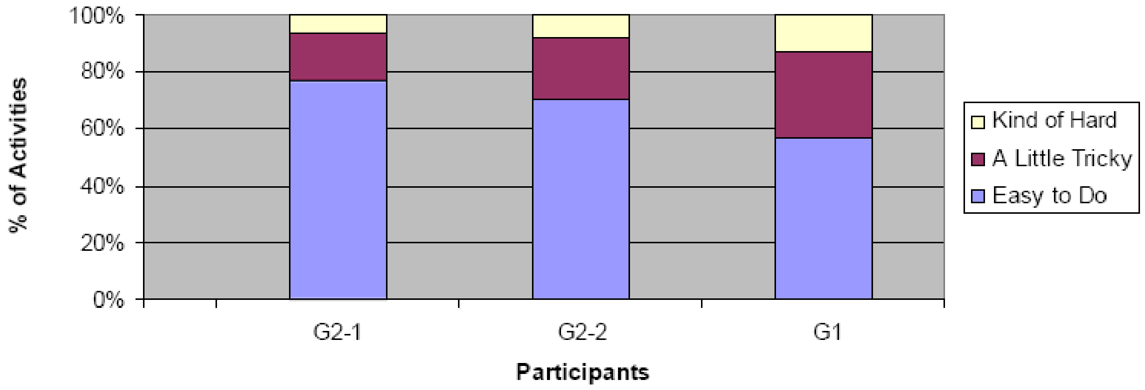
Two researchers were available to answer questions and to check children’s work. At the beginning of the first session, a researcher gave an overview of TileLand and a description of its functionality and how to use it. When new functionality was introduced (i.e., for Group 2), the researcher would provide a step-by-step description of it, but without suggesting when to use it.

Sources of data
The following were used as sources of data in this study:
(1) *Activities*. All children were asked to reproduce a series of sixty-five tiling patterns using TileLand. These patterns were progressively more complex, and hence, more difficult to reproduce, with the final patterns involving numerous repetitive patterns. Once children finished reproducing a pattern, they were asked to rate the difficulty of creating it. They were asked whether it was ‘easy’, ‘a little tricky’, or ‘kind of hard to do’. The responses were used to assess children’s perception of the level of difficulty to reproduce the different patterns.
(2) *Post test*. All children were asked to complete a 20-minute post test. In the post test, children were requested to recreate a set of patterns without using the tool. Children could either write or speak out their answers. The post test results would help assess how well children learned different programming concepts.
(3) *Video transcripts*. Children’s five sessions interacting with the tool were videotaped.
(4) *Direct observation*. As children interacted with the tool, two researchers took notes of their patterns of use. Attention was paid to children’s reaction when interacting with a new level and what and how they communicated with each other.
(5) *Interview transcripts*. After the post test, a number of children were asked some questions which were designed to seek further insight into their reactions to different interface scaffolds and components.
(6) *Log files*. Invisible to children, TileLand recorded log files of their interaction with the tool. These include commands clicked or typed, the time a command was issued, the time children take to finish a pattern, etc.

[*] One child dropped out from the study in second session (i.e., day two). The other child worked alone until the end of the study.

General Results

This section provides some general results of the study. Overall, both groups in Group 2 (G1.2 and G2.2) could manage to reproduce more patterns than Group 1. One pair in G2.2 was able to finish the entire series of sixty-five patterns and still had some time left at the end of the last session to create other patterns using the Level 7 environment. All children in Group 2 found the reproduction of the patterns to be relatively easy to do. Children in this group perceived over 70% of their completed patterns to be easy to do. Children in Group 1, however, perceived only 57% of the patterns to be easy to do (see Figure 8).



Age range: 6-8 9-12 9-12

Figure 8: Perception of the level of difficulty for completed patterns

It seems that the type of feedback acted as an important scaffold and affected children's perception of the difficulty level of patterns. Initially, Group 2 children were provided with immediate feedback—i.e., they could see the results of their actions without any time delay. Later, feedback was changed to requested, or on-demand. However, even in the last few sessions, these children did not make requests often. That is, children in this group could visualize in their minds longer and more complex patterns. In contrast, Group 1 children usually had to request feedback for any action they performed. At the beginning of the study, these children requested feedback for every command they typed in; they needed to confirm that what they were doing was correct before advancing. The log files indicate that the number of requests declined as the study progressed. However, on levels in which both groups 1 and 2 did not have immediate feedback, Group 1 children still made almost twice as many requests than Group 2 children.

One important and interesting observation was the efficiency with which children in the two groups reproduced the given patterns. On average Group 1 children reproduced patterns much more efficiently than the children in Group 2. Children in Group 1 continuously struggled with the Level 7 environment (e.g., see Figure 9a for time spent on the first five patterns); as a result, they were more conscious and careful of what commands to use in order to avoid errors. Children in Group 2, however, especially at the beginning of the study, often relied on clicking on the visual buttons to reproduce patterns, which required less reflection on their part. As such, they did not care if they made mistakes because they would erase any wrong patterns and start again. This behaviour gradually changed with the introduction of new interfaces; nonetheless, in general, children in this group used more commands than children in Group 1.

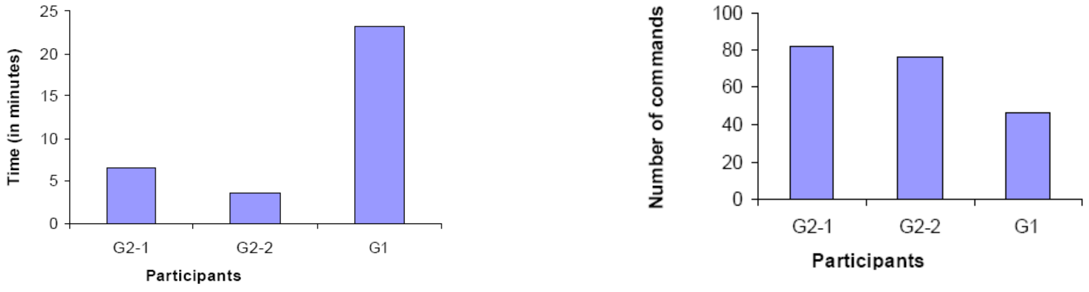


Figure 9: Time (a) and number of commands (b) used for the first five activities

Summary and Conclusion

This article reports the general results of a study conducted to investigate if interface scaffolds could make a new environment and its content more accessible to children. TileLand, the experimental tool used to conduct the study, included the following scaffolds: 1) visual buttons, which acted as commands to create patterns; 2) history slider, which assisted children in discerning the connection between textual commands and the visual buttons; 3) immediate feedback, which supported children in observing the results of their actions without any time lag; 4) sample groups, which helped children in understanding the purpose and utility of grouping several commands.

Although it is difficult to make generalizations beyond the tool and context of usage (i.e., pattern creation through composing procedural code), there are some lessons that can be taken from this study. First, the scaffolds which were placed in different levels seemed to make the tool more accessible to younger children, as evidenced by the perception of the level of difficulty of reproducing patterns. Second, the scaffolds provided cognitive support for children to help them reproduce increasingly more complex patterns with relative ease. Third, it appears that the scaffolds can be used to gradually moved children from an experiential (reactive) mode of thinking to a more reflective one, which is in harmony with other research findings (e.g., see Sedig et al. 2001). This study suggests that tools can be created in such a way that they are more readily accessible to a wider range of children if they are scaffolded properly. However, an in-depth examination of scaffolded tools is needed to better assess how each transitional environment should be designed to make the subsequent one more accessible.

References

Alessi, S.M., & Trollip, S.R. (2001). *Multimedia for Learning*. Needham Heights, MA: Allyn & Bacon.

Brusilovsky, P., Calabrese, E., Hvorecky, J., Kouchnirenko, A., & Miller, P. (1997). Mini-languages: A way to learn programming principles. *Education and Information Technologies*, 2 (1), 65-83.

Burton, R.R., Brown, J.S., & Fischer, G. (1984). Skiing as a Model of Instruction. In B. Rogoff, & J. Lave (Eds.), E*veryday Cognition: Its Development in Social Context*, Harvard University Press, Cambridge, MA. 139-150.

Canfield, D., & Cypher, A. (1999). Making programming easier for children. In A. Druin, (Ed., 1999), *The Design of Children's Technology*. San Francisco, CA: Morgan Kaufmann Publishers, Inc.

Druin, A. (Ed., 1999). *The Design of Children's Technology*. San Francisco, CA: Morgan Kaufmann Publishers, Inc.

Fernaeus, Y., Kindborg, M., & Scholz, R. (2006). Programming and tools: Rethinking children's programming with contextual signs. *Conference on Interaction Design and Children IDC '06*, June 7-9, Tampere, Finland. ACM Press.

Guzdial, M. (2004). Programming environments for novices. S. Fincher, & M. Petre (Eds.), *Computer Science Education Research*. Taylor & Francis Group, London, UK.

Kelleher, C. & Pausch, R. (2005). Lowering the barriers to programming: A taxonomy of programming environments and languages for novice programmers. *ACM Computing Surveys*, 37 (2), 83-137.

Masterman, E., & Rogers, Y. (2002). A framework for designing interactive multimedia to scaffold young children's understanding of historical chronology. *Instructional Science*, 30, 221-241.

Papert, S. (1980). *Mindstorms: Children, computers, and powerful ideas*. Basic Books, New York, NY, USA.

Puntambekar, S., & Hübscher, R. (2005). Tools for scaffolding students in a complex environment: What have we gained and what have we missed? *Educational Psychologist*, 40 (1), 1-12.

Quintana, C., Fretz, E., Krajcik, J., & Soloway, E. (2002). A case study to distill structural scaffolding guidelines for scaffolded software environments. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems: Changing Our World, Changing Ourselves, 2002*, ACM Press, New York, NY. 81-88.

Reiser, B.J. (2004). Scaffolding complex learning: The mechanisms of structuring and problematizing student work. *The Journal of the Learning Sciences*, 13 (3), 273-304.

Sedig, K., Klawe, M., Westrom, M. (2001). Role of interface manipulation style and scaffolding on cognition and concept learning in learnware. *ACM Transactions on Computer-Human Interaction*, 1 (8), 34-59.

Sedig, K., Morey, J., & Chu, B. (2002). TileLand: A microworld for creating mathematical art. *Proceedings of ED-MEDIA 2002: World Conference on Educational Multimedia and Hypermedia*, July, Denver, USA.

Sedig, K., & Liang, H. (2006). Interactivity of visual mathematical representations: Factors affecting learning and cognitive processes. *Journal of Interactive Learning Research*, 17 (2), 179-212. Association for the Advancement of Computing in Education.

Acknowledgements

This research has been funded by the Natural Sciences and Engineering Research Council of Canada.