Radia Perlman – A pioneer of young children computer programming

Leonel Morgado[*,1], **Maria Cruz**[2], and **Ken Kahn**[3]

[1] Department of Engineering, University of Trás-os-Montes and Alto Douro, Quinta de Prados, 5001-801, Vila Real, Portugal
[2] Department of Education & Psychology, University of Trás-os-Montes and Alto Douro, Edifício do CIFOP, R. Dr. Manuel Carmona, 5001-801 Vila Real, Portugal
[3] Animated Programs, President, 4 Folly Bridge Court, Shirelake Close, Oxford OX1 1SW, UK

The history of computer programming with young children (3, 4, and 5 years old) was initiated by a pioneer woman, more than 20 years ago. Between 1974 and 1976, Radia Perlman, working at the Massachusetts Institute of Technology, developed the first technological means allowing preliterate children to perform computer-programming. She went beyond the development of technology, and tried out her system with children in these age groups, providing many valuable insights and observations, which for many years remained the most consistent pieces of information on the issues and difficulties that arise when trying to help preliterate children learn to program. Her contributions are presented in light of recent developments in the field of computer programming for preliterate children.

Keywords children programming; preliterate programming; preschool; kindergarten; Perlman; TORTIS

1. Introduction

Radia Perlman (Figure 1) was the most notable pioneer on the use of computer programming with very young children (3 to 5 years old). Between 1974 and 1976, she developed a tangible programming system, called TORTIS, which she tested with young children, analyzing and reflecting upon the events taking place in the process. Her specific goal was "*to overcome the typing hurdle, and make many of the advantages provided by the learning of full computer languages accessible to children as young as three or four years*" [2, p. 4].

Figure 1 – Radia Perlman. *From: http://research.sun.com/people/images/radia/perlman2.jpg*

2. Tangible programming

A tangible programming system allows children to move physical objects to express programs, rather than type commands or manipulate pictures in a computer. In some cases, the final result of the programming with such systems must be watched on a computer screen, the most often-cited case being AlgoBlock system, developed 17 years after Perlman's system [3]. In other instances, the results of the programming process themselves are available outside the computer: a commonly-referenced example is the "Tangible Programming with Trains" system, which led to commercial products by several toy companies, such as LEGO's Intelli-train products [4].

3. The TORTIS system

Perlman's TORTIS system is tangible both in terms of the programming process, as we'll describe shortly, and in its results: it allowed children to program the behavior of a physical turtle robot, inspired

[*] Corresponding author: Leonel Morgado. E-mail: leonelm@utad.pt, Phone: +351 259350369

on a well-known subset of commands from the Logo programming language, known as TurtleTalk, consisting mostly of movement-related commands [1]: "*TORTIS* (…) *is a system of special terminals together with software*" [2], which aims to allow "*preschool children to communicate with and program the* [Logo] *turtle*" [1, p. 2]. It was "*designed so that only a few new concepts are introduced at a time but more can be added when the child becomes familiar with what he has*" [*ibid.*].

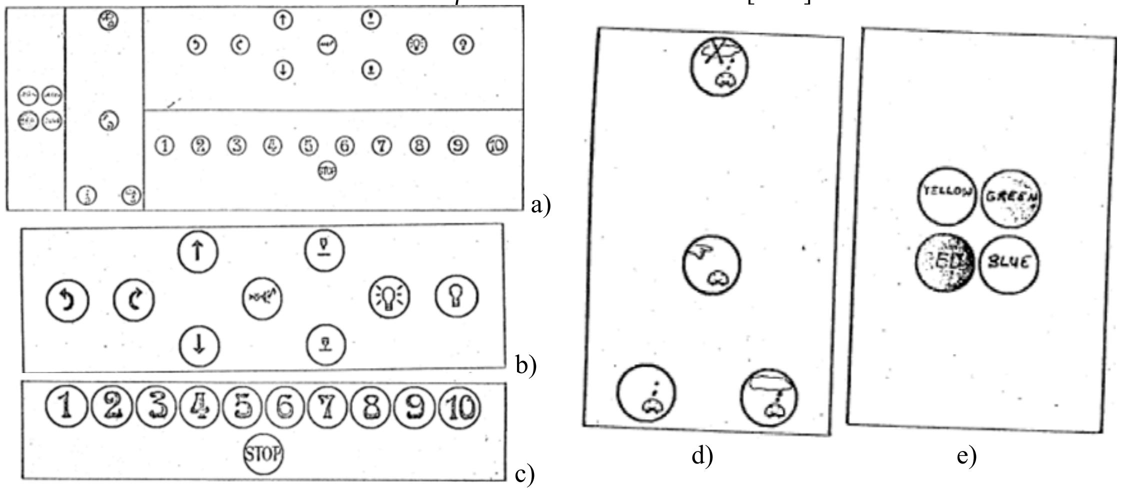

Fig. 2 TORTIS Button Box (a) and its components (b, c, d, & e) [2, pp. 9-12].

There are two kinds of terminals: one were the Button Boxes (Figure 2), which had buttons for typical TurtleTalk commands (such as "forward" and "right"), for parameters, and for control (buttons "start remembering", "do it", etc.); the other was called "Slot Machine" (Figure 3), and consisted of a set of several rectangular Plexiglas rows, each representing a procedure, "*with slots in the top for the user to place cards*" [1, p. 4], each card representing a command.

Perlman tested the first component of her system, the Button Box, with "*around 25 children aged 3 to 5*" [2, p. 14], and described the initial sessions with several of these children, the youngest aged 3½ [1].

4. TORTIS system operation

As mentioned above, the purpose of the system was to control a small "turtle" robot, consisting in "*a disk about one foot in diameter, equipped with a light, horn, and pen. It can move forward, backward, or rotate about its center* (...). *When the pen is down the turtle draws as it moves*" [1, p. 2].

The buttons in the Action Button Box (Figure 2b) matched the robot actions one-to-one. *E.g.*, pressing the button with the upwards-pointing arrow would make the robot move forward by a default amount. By combining this box with others, more complex commands could be issued. The box with number buttons (Figure 2c) allowed the child to press a number button before pressing an action button, causing the action to be executed that many times [*ibid.*]. The Memory Button Box (Figure 2d) told the turtle robot to "start remembering", "stop remembering", "do it", and "forget it", allowing children to issue combinations of action commands and number parameters, to be invoked later, using the Four Procedure Box (Figure 2e), whose buttons were colour-coded and could be assigned to procedures defined with the Memory Button Box [2]. A computer screen connected to the system presented the sequence of entered commands, and those assigned to the four coloured buttons [*ibid.*]. The box in Figure 2a presents the combination of all four boxes.

The design rationale behind having four boxes was that "*the child is presented with only a few new buttons at a time, and* (...) *there is always a set of button which the child really understands*" (*ibid.*). A child could also remove the new buttons from sight "*by unplugging the latest box and hiding it*" (*ibid.*), focusing on the part of the system with which the child was more comfortable to play with.

The second component of the TORTIS system was the Slot Machine (Figure 3). It consisted of an arrangement of several long Plexiglas boxes, called rows, each representing a procedure. On top of them were "*slots into which plastic cards* (...) *can be placed. Each card has a picture on it, representing a command*" [2].

The rationale for the Slot Machine was allowing children to visualize and reorder the sequence of commands: in the button box they would have to either memorize it or associate it them with the equivalent commands being displayed on a computer screen, not on the Button Boxes, and they wouldn't be able to edit a procedure. In the Slot Machine, not only the sequence of commands was always present in the actual instrument which the child was programming, it could be edited directly.

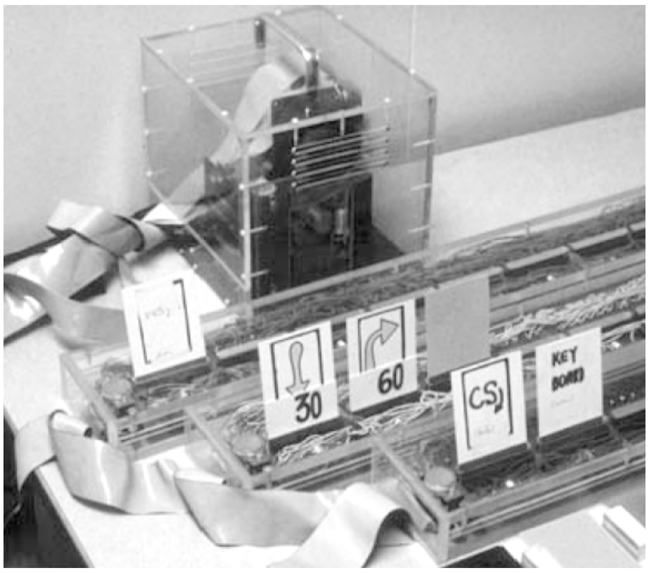

Fig. 3 TORTIS Slot Machine. *From:* [4, p. 328].

To execute a program in a Slot Machine row, a child simply pushed a button present at the end of each row, causing the command cards to be executed in order. "*A light in front of each slot lights up when the card in that slot is being executed. Each row is colored a different color. A solid-colored card represents a subroutine call to the row of that color*" (*ibid.*).

5. Brief research history on programming by preschoolers

The text-based roots of computer programming were the most obvious obstacle to the usage of programming by preschoolers. Besides Perlman's early TORTIS system, at about the same time two other systems were developed to address this issue: FASTR [5] and TEACH [6]. In them, programming used single keystrokes, overcoming the need to type words. Further systems followed similar approaches until the 1990s, when iconic-based or robotics-based programming systems started to emerge: *e.g.*, the Valiant Roamer [7], My Make Believe Castle [8], or PervoLogo [9]. The 2000s have been the stage for further iconic-based and tangible programming systems, allowing children to program without resorting to textual cues: *e.g.*, Electronic Blocks [10], Physical Programming [11] or Topobo [12].

However, a less obvious obstacle was much less tackled: the early stages of young children's cognitive, social, and verbal development. Several studies employed software-based Logo variants, similar to the FASTR and TEACH systems, but often focused on older children, including children nearing 6 years of age, who had initiated or were about to initiate formal schooling (their relevance to younger children thus quite limited). Many suffered from what Seymour Papert called "*technocentric questions*" [13, p. xiv]: researching the impact of Logo programming on children as if it were an independent variable, rather than looking at how Logo programming was used and at who, specifically, Logo programming was being used with (and detached from other educational practices). And several research efforts simply described the programming that took place, without looking into the educational processes that are taking place or the educational settings. In a nutshell, « *a venerable body of research on the use of Logo in early childhood environments, including preschool, kindergarten, and first-grade classrooms, has developed since the mid-1980s,* [but] *none of these studies describes the continuous classroom involvement of young children in Logo-rich environments that Papert envisioned.* » [14].

As for the novel systems we mentioned, available information is even more limited: documentation usually just reports children's "successful" use of the systems, or how children managed to complete a very specific activity, typically detached from preschool settings. Commendable contributions are descriptions of the programming process, albeit with just one or two young children and very few programming sessions [15], and descriptions of the educational context of programming activities, albeit with little information on the programming process [10].

Given the lack of scientific data on the use of programming with very young children, sought-after information includes in-depth analyses of children using programming systems, and approaches seeking their integration within preschool education settings and activities. Typical environments of preschools and kindergartens are structurally different from formal schooling, with emphasis on personal development rather than curriculum content.

Our own research aimed to change this state of affairs, by contributing data and insights on large-scale use of computer programming by preliterate children, employing an animated computer programming language named ToonTalk [16], to overcome both the barrier of typing commands and the limitations of single-letter commands. We documented, analyzed, categorized, and explained children's difficulties and achievements, and also looked into their teacher's difficulties. Based on this, we provided strategies to help overcome those difficulties, with a particular focus on providing a framework for integration of computer programming activities in the educational settings and practices of preschools and kindergartens [17].

6. Why Perlman was a pioneer

Perlman's most obvious pioneering contribution was technological: her hardware-based approaches became successful in much later systems: the Button Box was a precursor to the modern command board of the Valiant Roamer turtle robot [7], and the Slot Machine shares some insights with the often-cited AlgoBlock system [3] – *e.g.*, the ability to program by sequencing physical objects and the use of lights to indicate which card is in execution [2, p. 17]. Her software-based approach to the use of Logo, based on one-touch commands, avoided the typing of full textual commands, but as mentioned this approach was also being used, at more or less the same time, in the FASTR [5] and TEACH [6] systems.

But her most remarkable contribution was the way in which she paid close attention to the way in which children actually used the system, not just the technical issues or problems. She describes parent's and sibling's behavior during the programming sessions, the settings where programming is taking place (office, nursery school), and the reactions, behavior, expressiveness and comments of the children while programming [1-2]. She also pondered over the cognitive difficulties behind some aspects of programming and tried to overcome them.

Some of what she calls "*interesting things*" [2, p. 14] are quite similar to some of the programming hurdles we identified, described, and analyzed in 2006, 30 years later [17]. Specifically, one of the hurdles detected in 2006 is "*Children can easily get weary from unforeseen system behavior or difficulties*" [*ibid.*, p. 344]; Perlman's 1st and 2nd "*interesting things*" were similar early observations:

« *1) Once a child is confused by something he gets very discouraged and says, "that is too hard for me" or "I'm not smart enough for that", Both of which are certainly attitudes we do not want to cause, so it is important not to give the child new buttons before he is ready.*

2) Some children require constant interaction and suggestions about further things to try or else they start doing one thing over and over and get bored. » [2, p. 14]

Another hurdle detected in 2006 is "*Children find it hard to consider the possibility of being wrong and not knowing it*" [17, p. 350]; Perlman reported a similar problem:

« *If when drawing the picture they had, for instance, made the turtle turn the wrong way at first and then corrected, or stopped to make the turtle toot several times, they were surprised that the turtle did the same thing when drawing a picture the second time.* » [2, p. 15]

However, we differ on her interpretation of this problem. We believe that the issue here resides at the very heart of the notion of "wrongness". Perlman considered this to mean that children "*thought the picture was the procedure as opposed to the set of commands the turtle executed while remembering*" [2, p. 15]. Even so, her interpretation was an early helpful insight – an alternative, parallel perspective (if taken under a general formulation, *e.g.* "the result is the procedure", not "the picture is the procedure").

Another pioneering stance of Perlman was her concern with providing children with a visible track to the progress of a program's execution. As the Button Box buttons were pressed and the turtle robot moved the program's instructions were presented on a computer screen. According to her reports, though, children didn't manage to associate the screen commands with the movements of the turtle [*ibid.*, pp. 15-16]. The second component of the TORTIS system, the Slot Machine, was built specifically to address this problem: the slots in the top of the Plexiglas rows (Figure 3) were meant for children users to place cards representing turtle commands. The reasoning behind it was enabling children to be physically involved in the process of specifying a sequence of commands, which hopefully would contribute to overcome the problem, as she saw it, of mistaking the resulting picture for the program. This conception didn't prove fruitful; in her own words, "*initially there is so much that the child has to do to get the turtle to execute a command (find a card, put it in a position, and push a button) that it is sometimes difficult for them to understand that each card stands for some specific command*" [*ibid.*, p. 26].

Perlman did provide suggestions regarding system improvements or novel systems, precisely with the goal of achieving this immediacy in response to programming. For instance, she suggested the use of the Slot Machine to produce tunes, with a command card for each note, and a number argument specifying the duration of that note; she also suggested turning indicator lights into pushbuttons to allow children to cause the immediate execution of a specific instruction within a program, simplifying debugging [*ibid.*].

She was already expressing her concern about the way in which programming systems such as her Button Box, based on the reproduction of hidden-away procedures, lack a proper editing procedure (the Slot Machine programs were easy to edit, simply re-ordering the cards). The problem with this condition, in light of Seymour Papert's ideas [13; 18], is that "*without the ability of easy editing the child does not learn the healthy concept that there is no 'right' or 'wrong'* [in learning through programming], *and that any procedure can be fixed to do what they want, and even a working procedure can always be improved*" [2, p. 15]. Conversely, she also acknowledged that "*adding editing commands before the child has mastered the four memory buttons* [in a more general sense, before the child masters the immediate-response programming system] *would overwhelm the child with too many new concepts*" [*ibid.*]. While agreeing in principle with these observations, we consider this issue to be more of a question of the style in which children's programming activities are conducted, based on information from our own research efforts [17].

7. Final remarks

We believe Perlman was fundamentally right on her approach to the research of computer programming by young children and on her set of goals for further research, laid down in 1976: « (…) *there is a whole area of research involving working with children with the system, finding difficulties they encounter, and* (…) *inventing new ways to present the concepts they find difficult* (…) » [2, p. 29]

After the 1974-1976 work of Radia Perlman, in 1980 Seymour Papert stated "*I have seen hundreds of elementary school children learn very easily to program, and evidence is accumulating to indicate that much younger children could do so as well*" [13, p. 13]. Sadly, in that seminal work, he did not provide more information on that evidence, although it certainly included Perlman's work, seeing that it was taking place at MIT's AI Lab, of which Papert was co-director.

Since the creation of Logo, a large body of research has been developed on the use of computer programming by children, but mostly centered on children of formal school age, *i.e.* 6 years old or more [19]. Research focusing on the use of computer programming with younger children is less common, as can be seen by analyzing current review studies [14; 17; 20-21].

The work of Radia Perlman was the first and most prominent arrowhead pointing in the right direction. We have followed it and can only hope (and endeavor) that more researchers do the same.

Acknowledgement Various details provided by Radia Perlman herself in an e-mail response to our queries regarding the TORTIS system were instrumental in allowing us to better understand its operation and the overall scope of her work.

References

[1] Radia Perlman, TORTIS – Toddler's Own Recursive Turtle Interpreter System, in: MIT AI Memo No. 311/Logo Memo No. 9 (MIT AI Lab, Massachusetts Institute of Technology, Cambridge, MA, USA, 1974).

[2] Radia Perlman, Using computer technology to provide a creative learning environment for preschool children, in: MIT AI Lab Memo No. 360/Logo Memo No. 24 (MIT AI Lab, Massachusetts Institute of Technology, Cambridge, MA, USA, 1976).

[3] H. Suzuki, & H. Kato, AlgoBlock: a tangible programming language, a tool for collaborative learning, in: P. Georgiadis, G. Gyftodimos, Y. Kotsanits, & C. Kynigos (eds.), Proceedings of the Fourth European Logo Conference, EuroLogo'93, University of Athens, 28-31 August 1993 (Doukas School S.A, Athens, Greece, 1993), pp. 297-303.

[4] T. S. McNerney, Tangible Programming Bricks: An approach to making programming accessible to everyone, Master's thesis (Program in Media Arts and Sciences, School of Architecture and Planning, Massachussets Institute of Technology, Cambridge, MA, USA, 2000).

[5] P. Goldenberg, FASTR – A Simple Turtle Runner, in: Logo Working Paper No. 30 (MIT AI Lab, Massachussets Institute of Technology, Cambridge, MA, USA, 1974).

[6] Cynthia Solomon, & Seymour Papert, Teach: A Step Toward More Interactive Programming, in: Logo Working Paper No. 43 (MIT AI Lab, Massachussets Institute of Technology, Cambridge, MA, USA, 1975).

[7] Valiant Technology, Valiant Roamer User Guide, retrieved May 18th, 2006, from http://www.valiant-technology.com/archive/freebies/userguide/user1.htm.

[8] D. Bearden, & K. Martin, My Make Believe Castle – An Epic Adventure in Problem Solving, Learning and Leading with Technology, ISSN 0278-9175, 25, 5, pp. 21-25 (1998).

[9] S. F. Soprunov, ПервоЛого: Пособие для учителей ("PervoLogo: Benefits for the teachers") (INT, Moscow, Russia, 1996).

[10] P. Wyeth, & H. C. Purchase, Programming without a Computer: A New Interface for Children under Eight, in: Proceedings of the First Australasian User Interface Conference, ISBN 0-7695-0515-5 (IEEE Computer Society, Washington, DC, USA, 2000).

[11] J. Montemayor, A. Druin, A. Farber, S. Simms, W. Churaman, & A. D'Amour, Physical programming: designing tools for children to create physical interactive environments, in: D. Wixon (ed.), Proceedings of the SIGCHI conference on Human factors in computing systems: Changing our world, changing ourselves, ISBN 1-58113-453-3, (ACM Press, New York, NY, USA, 2002), pp. 299-306.

[12] H. S. Raffle, Topobo: A 3-D Constructive Assembly System with Kinetic Memory, Master's dissertation (School of Architecture and Planning, Massachusetts Institute of Technology, Cambridge, MA, USA, 2004).

[13] Seymour Papert, Mindstorms: Children, Computers, and Powerful Ideas, ISBN 0-465-04629-0 (Basic Books, New York, NY, USA, 1980).

[14] C. W. Gillespie, Seymour Papert's Vision for Early Childhood Education? A Descriptive Study of Head Start and Kindergarten Students in Discovery-based Logo-rich Classrooms, Early Childhood Research & Practice, ISSN 1524-5039, **6**, 1 (2004).

[15] O. Zuckerman, & M. Resnick, A physical interface for system dynamics simulation, in: CHI '03 extended abstracts on Human factors in computing systems, ISBN 1-58113-637-4 (ACM Press, New York, NY, USA, 2003), pp. 810-811.

[16] Ken Kahn, ToonTalk™—An Animated Programming Environment for Children, Journal of Visual Languages & Computing, ISSN 1045-926X, **7**, 2, pp. 197-217 (1996).

[17] Leonel Morgado, Framework for Computer Programming in Preschool and Kindergarten, Doctoral thesis, (Universidade de Trás-os-Montes e Alto Douro, Vila Real, Portugal, 2006).

[18] Seymour Papert, The Children's Machine: rethinking school in the age of the computer, ISBN 0-465-01063-6 (Basic Books, New York, NY, USA, 1993).

[19] D. H. Clements, & B. K. Nastasi, Metacognition, learning, and educational computer environments, Information Technology in Childhood Education Annual, ISSN 1522-8185, **1999**, 1, pp. 3-36 (1999).

[20] D. H. Clements, The Future of Educational Computing Research: The Case of Computer Programming, Information Technology in Childhood Education Annual, ISSN 1522-8185, 1999, 1, pp. 147-179 (1999).

[21] D. H. Clements, & J. Sarama, Strip Mining for Gold: Research and Policy in Educational Technology—A Response to "Fool's Gold", Educational Technology Review, ISSN 1551-3696, **11**, 1, pp. 7-69 (2003).