Music Learning with Massive Open Online Courses (MOOCs)
L. Steels (Ed.)

doi:10.3233/978-1-61499-593-7-213

Chapter 14. A Half Century Perspective On The Role Of Computers In Learning And Teaching

Ken KAHN
University of Oxford

Abstract. For more than fifty years people have been exploring how computers might enhance learning and teaching. The malleable nature of computers has enabled suggestions that a computer can act like flash cards, personal tutors, textbooks, reference books, virtual laboratories, quizzes, virtual spaces, lecture halls, and study groups. Perhaps the most radical suggestion has been to see the computer as something learners can creatively mold into something personally meaningful that is dynamic, interactive, and shared. And that the process of constructing such computational artefacts is rich in learning opportunities. These range from a deeper understanding of the subject matter of the constructions to high-level skills in thinking and problem solving.

Keywords. Constructionism, technology-enhanced learning, Seymour Papert, history of computers and education

Introduction

The idea that computers can play an important role in learning and teaching is over fifty years old. This chapter describes the history of attempts to use computers to support and enhance learning from a personal perspective. Instead of a complete history it attempts to highlight groundbreaking and significant ideas and computer systems that have led to today's efforts to provide technology-enhanced learning. Some systems use the computer to emulate older paper-based technologies. Others attempted to give the computer the role of teacher or tutor. The systems that are described in the most detail in this chapter are those that attempt to use the computer to provide novel learning experiences that were impossible or impractical before.

The 1950s through the 1970s were dominated by "computer-aided instruction" systems that attempted to teach in a very didactic and mechanical manner. These were based upon behaviorist theories of learning. Research laboratories at MIT, Xerox PARC, and the University of Edinburgh were exploring a very different approach. Instead of the computer programming the student, the student was given tools for programming the computer. Creativity and exploration were emphasized. Early attempts at computer tutoring systems were made. Programming languages designed specifically for learners were developed.

The 1980s saw the wide-spread dissemination of personal computers and programming languages for children. There were efforts to enhance these languages with new ideas from computer science. Media creation was combined with program creation. Intelligent tutoring systems were demonstrated to work well for a limited number of topics.

The 1990s saw the wide-spread use of "multi-media" to enhance education. Programming languages for children expanded into new territories. Learners were supported in building computer games and programming robots.

The 2000s saw the integration of the web into educational software. Learners were connected by the World Wide Web and able to easily share their constructions. Multi-user three-dimensional virtual spaces became popular places to explore their potential to enhance learning. Many explored the benefits of each learner having their own personal computer.

The 2010s saw the introduction of MOOCs (massive open online courses), web-based programming environments, and mobile devices.

1. 1950s and 60s

Alan Perlis saw the potential of computer programming for learning by science, mathematics, and engineering students in the mid-1950s. He began teaching the first freshman course on computer programming in 1958. In 1961 in a lecture at MIT he said "The purpose of a course in programming is to teach people how to construct and analyze processes [1]. J.C.R. Licklider commented "... I see computer programming as a way into the structure of ideas and into the understanding of intellectual processes that is just a new thing in this world".

In 1964 Kemeny and Kurtz introduced the Basic programming language, the first programming language designed for learners and beginners [2]. It contained many comprises due to the hardware limitations of the day. Variable names, for example, were limited to one letter followed by digits. While initially limited to use in universities, Basic became very popular with schools and hobbyists in the 1970s and 80s.

In 1967 Seymour Papert, Wally Feurzeig, Cynthia Solomon, and Danny Bobrow developed the Logo programming language. Unlike Basic, which was designed to provide the minimal language that can support student programming, Logo was designed to be a rich and powerful language. Logo is the result of "child-engineering" the best ideas in computer science at the time. It borrowed very heavily from the Lisp programming language which was being used by artificial intelligence researchers. Logo was conceived of as both a tool for learners to use to express themselves creatively and an "object to think with" [3]. Initially the projects created using Logo focused upon word and list processing and mathematics. For example, children constructed programs that generated poetry. By 1969 Logo was enhanced to control "floor turtles", robots that could be commanded to move forward or turn. This became the basis of the very successful turtle graphics when "screen turtles" were introduced in 1972.

A very different trend that began in 1960 is "computer-aided instruction". This was pioneered by the Plato system [4]. The Plato system initially focused on presenting multiple-choice or numeric questions and automated responses. Its initial innovations were in computer graphics and display terminals that it pioneered. The Plato system grew over time to include interactive simulations, educational games, and discussion forums.

But unlike the efforts around Basic and Logo, Plato was based upon a didactic teaching method instead of the programming languages' support of learner-centered problem solving and creativity.

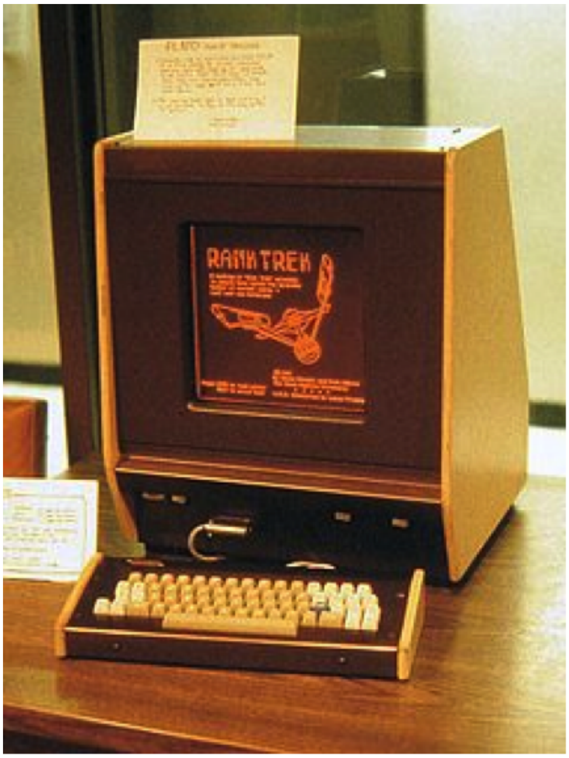


Figure 1. The PLATO System

The idea of using computers in education was very radical in a period where computers were few and very expensive. As Hal Abelson, one of the earlier pioneers of Logo programming, said "You really have to try hard to get into the mindset of that time, because a computer in those days was something that cost several million dollars. And the idea that you would take the most advanced computing research equipment around anywhere, and you would let fifth graders ... start playing with it, it was just mind boggling. For the first 10 years of that, people just thought we were nuts" [5].

2. 1970s

The next decade saw substantial progress in efforts around the Logo programming language. Since the center of this research was the MIT Artificial Intelligence Laboratory it is perhaps not surprising that many efforts attempted to connect Logo and AI. Gerry Sussman [6] and Ira Goldstein [7] produced systems that helped debug and teach Logo. Danny Hillis wrote about AI projects that children could do in Logo. Radia Perlman developed special hardware to provide interfaces appropriate for very young children to construct Logo-like programs [8].

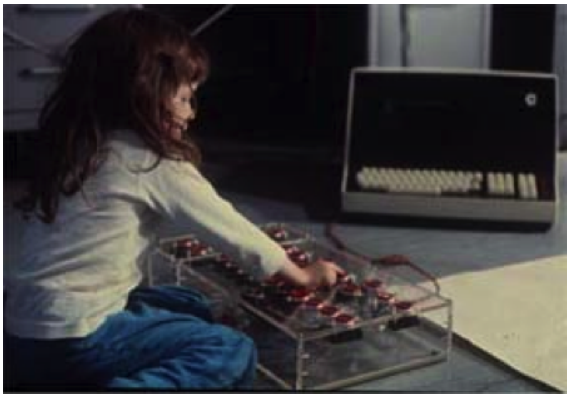


Figure 2. Radia Perlman's Button Box for Preschoolers

This was the decade when the concept of object-oriented programming was incorporated into programming languages for children. Smalltalk 72 and 76 were designed for children and inspired by Logo. (Smalltalk 80, however, was developed as a tool for professional programmers.) Director was another object-oriented language for children that was designed to support the programming of animation [9].

During the 1970s some versions of Logo were created to support the programming of music, color graphics, three-dimensional graphics, and animations. Implementations of Logo appeared on computers inexpensive enough for schools to acquire and the use of Logo by students expanded beyond the laboratory by the end of the decade.

Researchers on intelligent tutoring systems made substantial progress this decade. A notable example is Buggy [10], which was able to diagnosis students' arithmetic mistakes and respond appropriately.

Research on the use of computer games for learning began in this decade as well. Games were developed for educational purposes and researchers explored the educational value of games designed for entertainment purposes [11][12]. The first computer game, MIT Space War, created in 1961, attempted to have accurate positioning of stars and simulation of gravity and hence could be argued to be "educational". Seymour Papert later argued that more serious learning can result from challenging entertainment games than with many "edutainment" games that attempt to be both educational and entertaining [13]. Educational games and educational uses of commercial games has continued to be an active area of development and research for nearly fifty years.

3. 1980s

With the spread of relatively inexpensive personal computers, programming languages for children became widespread in schools and the home. This, combined with Seymour Papert's very influential 1980 book, Mindstorms: Children, Computers, and Powerful Ideas, led to an explosion of activities around Logo. Many schools in the US required its teaching. It became part of the UK National Curriculum in 1988. Far too often, however, the spirit of Logo was lost and children were taught Logo in a way that was far from the creative, exploratory, reflective style it was designed for.

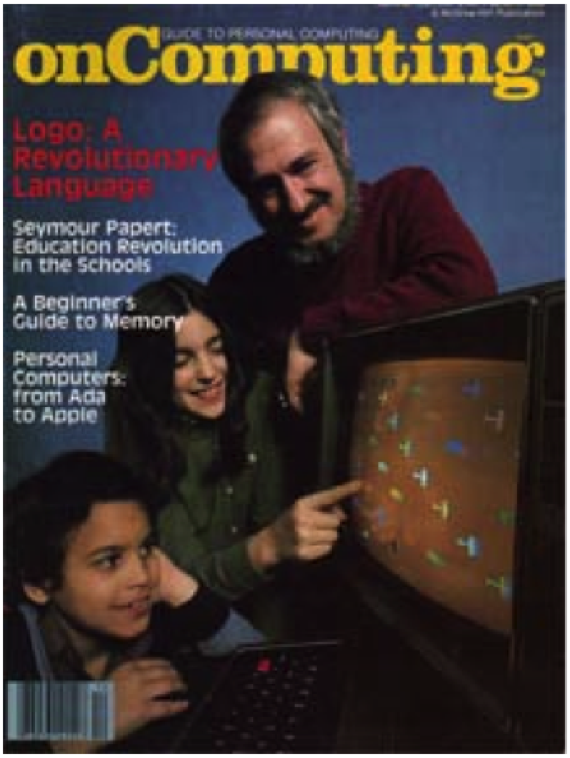

GUIDE TO PERSONAL COMPUTING
onComputing
Logo: A Revolutionary Language
Seymour Papert: Education Revolution in the Schools
A Beginner's Guide to Memory
Personal Computers: from Ada to Apple

Figure 3. Logo becomes mainstream

Abelson and diSessa wrote Turtle Geometry: The Computer as a Medium for Exploring Mathematics, a book that explores how advanced mathematics could be taught building upon the turtle geometry of Logo [14]. While this undoubtedly helped counter the misconception that Logo was only for primary school children, it was commonly held that Logo was too childish for use by older students. A three-volume book, Computer Science Logo Style by Brian Harvey [15], was aimed at high school teaching and was partially successful in countering this. This misconception is particularly ironic given that Logo was based upon Lisp, an AI programming language with very powerful primitives for dealing with symbolic information.

This decade saw a flourishing of experimental variants of Logo and other programming languages for children. Object Logo [16] was an object-oriented programming language that contained classical Logo as a sub-language. Multi-Logo [17] explored Logo running in multiple processes. Boxer [18] tightly integrated a powerful Logo dialect with a sophisticated user interface. Efforts were made to take other artificial intelligence languages and adapt them for use by school children [19] [20].

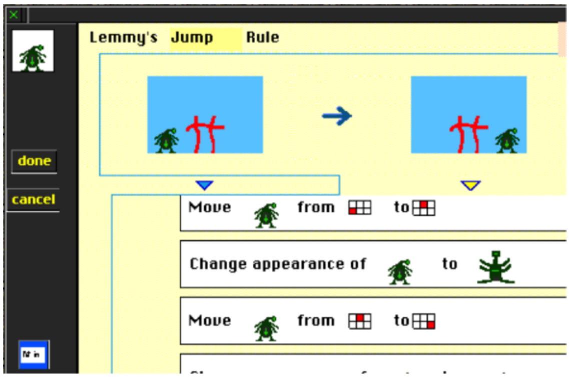
Intelligent tutoring systems made strong advances but only in a few select subjects such as teaching algebra, geometry, or computer programming [21].

4. 1990s

The 1990s saw a good deal of activity around adding concurrency and visual syntaxes to programming languages for children. One of the drivers towards concurrency was agent-based modeling. The idea is that one can learn about complex systems by constructing, observing, and experimenting with simulations of interacting entities. This began with StarLogo [22] to be followed by NetLogo [23] and Agentsheets [24]. These efforts to introduce agent-based modeling to school children were described in Mitchel Resnick's book Turtles, Termites, and Traffic Jams [22]. By using these tools, students could acquire a deeper understanding of the underlying processes in scientific phenomena. Topics include those in the physical, biological, and social sciences as well as the humanities including history, philosophy, and language. The educational value of computer programming expanded by providing new ways of learning most school subjects.

Concurrency appeared in other programming languages for children. Stagecast Creator [25] was based upon concurrent rewrite rules. ToonTalk [26] followed the design philosophy of Logo to child-engineer the best computer science programming language ideas. Three decades after Logo's design borrowed from Lisp, ToonTalk's design built upon the ideas of concurrent constraint programming [27]. All of these languages supported programs with multiple simultaneous activities, but only ToonTalk provides general mechanisms for communication and coordination between multiple processes.

The other major trend in the 1990s was to explore graphical syntaxes for programming languages. Agentsheets and Stagecast Creator (then called KidSim) supported expressing programs as graphical rewrite rules. For example, here is how one expressed in KidSim that a character should jump over obstacles [25]:



Move from to
Change appearance of to
Move from to

Figure 4. A KidSim rule for jumping over fences

These graphical rewrite rules are intuitive and surprisingly expressive but support abstract rules poorly. Agentsheets addresses this by combining graphical rewrite rules with a spreadsheet metaphor and a scripting language for advanced users.

Agentsheets and Stagecast Creator/KidSim also supported program construction by demonstration. ToonTalk took this to the extreme: the only way to construct programs was via demonstration followed by removal of details to obtain abstraction [28]. ToonTalk has no static syntax; programs are created and viewed as animations in a game-like environment. Programming by demonstration in ToonTalk can be successfully performed by preschoolers [29]. The lack of a static syntax does interfere with scanning and editing programs however. Unlike other programming languages for children, ToonTalk programs can be completely text-free, making them particularly suitable for pre-literate children and internationalization.

In 1996 LogoBlocks [30] pioneered a graphical syntax that subsequently became hugely popular. It introduced shaped blocks that can be dragged and dropped to assemble programs. These blocks correspond to program commands, expressions, data, and control structures. Palettes of blocks enable users to construct programs by selecting the needed parts. Most importantly, these blocks snapped together only when the parts fit together like a jigsaw puzzle. Syntax mistakes are not expressible in such a system. Unlike textual programming languages, the user doesn't need to remember what primitives are available, but instead can select them from palettes. In the next decade the syntactic ideas of LogoBlocks were integrated with StarLogo TNG [31], Scratch [32], Snap! [33], MIT App Inventor [34] and many more [35].

Another programming language trend of the 1990s was to support robot construction kits. A pioneering example of this was LEGO/Logo [36]. As Seymour Papert wrote [37], "LEGO/Logo is a computer-based system that offers a new approach to elementary science education. LEGO/Logo places engineering and design activities at the center of the science curriculum. Using the system, students build machines out of LEGO building pieces (including gears, motors and sensors), connect the machines to a computer, [and] then write computer programs to control the machines. These activities can provide a more meaningful and motivating context for learning traditional science-curriculum concepts while also introducing elementary school students to important engineering and design concepts that are rarely addressed in today's curricula."

Idit Harel [38] and later Yasmin Kafai [39] explored the idea of children programming educational games for younger children. Children using the Logo programming language designed and implemented games to teach concepts about fractions to younger children. Kafai created a sustainable school culture consisting of three grade levels. The oldest children built the games for the youngest children with assistance from middle children who the next year became the game makers. Studies demonstrated that the children who designed and constructed educational games learned the subject matter of their games very well even if the games themselves were not particularly pedagogically effective for the younger students.

The 1990s also saw the rise of multi-media CD-ROMs. For example, Microsoft's Encarta encyclopedia included much that paper alternatives lacked, including audio, animations, videos, and interactive applications. Subjects could be connected by hyperlinks. Novel interactive books on CDROMs where illustrations were animated and reacted to clicks became very popular [40]. So-called "edutainment" games appeared on many CD-ROMs in the 90s.

5. 2000s

The most interesting developments in the first decade of the 21st century were creative and game-changing uses of the Internet. An early example of this was the European WebLabs project [41] which supported children in exploring mathematics and science computationally and sharing and discussing their discoveries in web reports. This added extra dimensions to their learning. In publishing on the web students reflected deeply about what they discovered and worked hard to communicate it effectively. The discussions attached to each report often contained constructive criticism and suggestions. The students were not only doing science and exploring mathematics by constructing computer programs, but were also engaged in the process of academic publication to peers.

The Modelling4All project [42] built a web-based tool (the Behaviour Composer) to support teaching, research, and public engagement with agent-based modelling (ABM). By building on the popular open-source NetLogo agent-based modelling system, the project was able to focus upon higher-level issues of enabling a range of users, including those with no programming experience, to produce open, modular, transparent, sharable models. The Behaviour Composer is web-based both in the sense that one can construct and run models from a modern web browser as well as in supporting sharing models, model components, and interactive tutorials as public web pages.

The Scratch programming language from MIT became very popular after launching its website in 2007. Seven years later, over 7 million projects have been shared on the website, over 4.5 million users registered, and 35 million comments posted. Users support and learn from each other. About 30% of projects are "remixes" where someone makes a variant of another's project (with attribution maintained) [43]. As discussed earlier, Scratch's syntax contributes significantly to its popularity; however, the popularity of other child-engineered programming environments with a similar syntax pales in comparison. The website provides support, motivation, millions of sample projects, and a sense of community that accounts for the popularity of Scratch.

Second Life, a communal three-dimensional virtual world, became very popular in the 2000s. Thousands of avatars controlled by their "owners" interact in this virtual world. "Residents" of Second Life can earn virtual money, build virtual objects, buildings and spaces, and communicate with other residents. A teen-only Teen Second Life was launched in 2005. Educators saw this as potentially a new and effective place for teaching and learning. Many museums opened up Second Life "branches" that exploited the unique capabilities of this virtual world. For example, the US Air and Space Museum built replicas of rockets that visitors could enter and launch. Schools and universities also opened locations. Some uses were recreations of ordinary lecture- oriented teaching while others explored new possibilities. For example, Dr. Peter Yellowlees created virtual hallucinations based upon the experiences of schizophrenia patients. Visitors could experience first-hand what it's like to have schizophrenia [44].

Due to the appearance of inexpensive micro-controllers, educational robotics kits evolved from being cabled to a controlling personal computer to running programs inside the robot itself. Lego's Mindstorms (inspired by Seymour Papert's book of the same name from 1980) became popular. Robot behaviors were still programmed on personal computers, but once downloaded into a micro-controller, they became autonomous. Students used these kits to make a wide range of interactive gadgets. The Lego Corporation offered RoboLab, a graphical dataflow language, to schools using Mindstorms. Researchers implemented dozens of other languages for controlling Mindstorms bricks.

2006 saw the launch of the One Laptop per Child project [45]. The dream was to support the dissemination of inexpensive laptops to every child in the developing world. Special hardware and software was developed. The laptops were designed to have very low power requirements so that electricity could be provided by other means if electrical power wasn't available. The laptops can easily be connected in a network to share

Figure 5. The US Air and Space Museum in Second Life

resources and support multi-user applications. Over two million laptops were produced and in a few countries there were enough to provide a laptop to each child (Uruguay for example). Two million is a significant number, but much fewer than the hundreds of millions initially expected.

6. 2010s

By 2010 web-based technology (JavaScript, CSS, and HTML5) began to be mature enough that serious programming environments could be built to run in any modern browser, including those on tablets and smartphones. Implementations of Logo, ToonTalk, and dialects of Scratch appeared that ran immediately in a browser without any installation or plugins. Programs could be stored seamlessly to cloud storage so that students could move easily between school, home, and libraries as they constructed computational artefacts.

Snap! is a more powerful variant of Scratch, implemented as a web application [33]. It contains new primitives for supporting first-class functions (functions that can create or use other functions) and lists. Unlike Scratch, it is suitable for an advanced high school or beginning university computer science course. It illustrates a tension between programming languages designed to be easy to learn, such as Scratch, and those designed to support more advanced computational concepts and the construction of larger, more complex programs.

ToonTalk was built as a Microsoft Windows application. ToonTalk Reborn is a reimplementation and redesign for the web [46]. ToonTalk programs can be associated with any browser element, giving them interactivity. Widgets constructed in ToonTalk can be embedded inside web pages. Programs and widgets can be dragged between browsers. Programs can be published as automatically generated web pages surrounded by editable rich text.

The Khan Academy [47] began as an online mathematics learning site relying heavily on short videos. It has delivered over 400 million lessons in many school subjects. It delivers 4 million exercise problems daily. Many teachers use it to "flip the classroom" where watching videos as homework replaces classroom lectures. This frees up classroom time for personal support of students as they attempt to do exercises.

This decade has also seen the rise in online tutorials and puzzles designed to teach programming. Code.org has promoted the "Hour of Code" which has reached almost 50 million people. A very impressive online programming tutorial is from the Khan Academy [48]. Each programming lesson replays the actions of an expert with audio commentary. The web page is split between the coding area and an area displaying the result of running the code. Edits of the code are immediately reflected in the output/visualization area. Students can at any time pause the playback and experiment with their own edits or additions to the code area and receive instant feedback.

Another trend of this decade is the programming of smart phones. The MIT App Inventor [34] enables learners to build Android apps in a web browser that can be run either on a phone or in a phone emulator in the browser. It relies upon a variant of the block syntax made popular by Scratch. Pocket Code [49] enables learners to build phone apps on their phones. Its block syntax and interface was designed to work on small screens and touch sensitive devices.

Massive open online courses (MOOCs) became a hot topic in computer-supported learning when in 2011 Stanford University offered a free course Introduction to AI. Its enrolment quickly reached 160,000. Since then courses have been offered at hundreds of universities world-wide with total enrolment of many millions [51] [50]. Because of the large numbers of students, MOOCs generate "big data". This data can be mined to continually improve courses based upon solid evidence.

7. Looking back and forward

'The best way to predict the future is to invent it.' Alan Kay [52].

Great inventions in using computers to supporting learning have been made during the last fifty years. These include programming languages designed for children, intelligent tutoring systems, online courses, shared virtual spaces, robotics kits, and thousands of games. And learning doesn't stop with software specifically designed for education but includes use of mainstream developments such as Wikipedia, Google Earth and Maps, social media, computer graphics and animation authoring systems, photo and video editing, 3D printing, spreadsheets, presentation tools, and collaborative document editors.

As computational technology becomes widespread and matures and as the price of computational hardware decreases, we get closer to the fulfilment of the dreams of Seymour Papert, Nicholas Negroponte, and many others that learning by every child on the planet can change dramatically for the better. Children increasingly have devices that enable them to creatively express themselves in a medium that brings their ideas and creations to life. In doing so they acquire powerful ideas for becoming better problem solvers, thinkers, and learners.

References

[1] M. Greenberger, editor, Computers in the World of the Future, MIT Press, Cambridge, MA, 1962.

[2] Dartmouth College Computation Center, A Manual for BASIC, the elementary algebraic language designed for use with the Dartmouth Time Sharing System. Archived from the original on 2012-07-16. http://www.bitsavers.org/pdf/dartmouth/BASIC_Oct64.pdf, 1964

[3] Seymour Papert, Mindstorms: Children, Computers, and Powerful Ideas, Basic Books, 1980.

[4] http://www.platohistory.org/blog/timeline/

[5] Larry Hardesty, "The MIT roots of Google's new software", MIT News Office, August 19, 2010, http://newsoffice.mit.edu/2010/android-abelson-0819

[6] Gerald Sussman, HACKER: A model of skill acquisition, MIT doctoral thesis, 1973.

[7] Ira Goldstein, Understanding simple picture programs, MIT doctoral thesis, 1974.

[8] Leonel Morgado, Maria Cruz, and Ken Kahn, "Radia Perlman A pioneer of young children computer-programming", Current developments in technology-assisted education, Proceedings of m-ICTE, 2006.

[9] Ken Kahn, "Director Guide", Technical Report 482B, MIT AI Lab, December 1979.

[10] John Seely Brown and Kurt VanLehn, "Repair Theory: A Generative Theory of Bugs in Procedural Skills", Cognitive Science, vol. 4, no. 4, pp. 379-426, 1980.

[11] Barbara White, Designing Computer Games to Facilitate Learning, MIT doctoral thesis, 1981.

[12] Thomas Malone, "Toward a theory of intrinsically motivating instruction." Cognitive science 5.4 (1981): 333-369.

[13] Seymour Papert, "Does Easy Do It? Children, Games, and Learning", Game Developer, June 1998.

[14] Harold Abelson and Andrea diSessa, Turtle Geometry: The Computer as a Medium for Exploring Mathematics, MIT Press, 1981.

[15] Brian Harvey, Computer Science Logo Style, Volumes 1, 2 and 3, MIT Press, Second edition, 1997.

[16] Gary L. Drescher, "Genetic AI: Translating Piaget into LISP", Instructional Science, Volume 14, Issue 3-4, pp 357-380, May 1986.

[17] Mitchel Resnick, "MultiLogo: A Study of Children and Concurrent Programming", Interactive Learning Environments, 1:3, 153-170, 1990, DOI: 10.1080/104948290010301

[18] Andrea A. diSessa, "Twenty reasons why you should use Boxer (instead of Logo)", M. Turcsányi- Szabó (Ed.), Learning and Exploring with Logo: Proceedings of the Sixth European Logo Conference. Budapest Hungary, 7-27, 1997.

[19] Ken Kahn, "A grammar kit in Prolog", M. Yazdani, editor, New Horizons in Educational Computing. Ellis Horwood Ltd., 1984. Also In Instructional Science and Proceedings of the AISB Easter Conference on AI and Education, Exeter, England, April 1983.

[20] Richard Ennal, "Teaching logic as a computer language in schools," Proceedings of the First International Logic Programming Conference, Marseille, France, September 1982.

[21] John R. Anderson, C. Franklin Boyle, Albert T. Corbett, and Matthew W. Lewis, "Cognitive Modelling and Intelligent Tutoring", Artificial Intelligence, Vol 42, pages 7-49, 1990.

[22] Mitchel Resnick, Turtles, Termites, and Traffic Jams: Explorations in Massively Parallel Microworlds, MIT Press, Cambridge, MA, 1994.

[23] Uri Wilensky, NetLogo, http://ccl.northwestern.edu/netlogo/. Center for Connected Learning and Computer-Based Modeling, Northwestern University. Evanston, IL, 1999.

[24] Alex Repenning, "Creating User Interfaces with Agentsheets," 1991 Symposium on Applied Computing, Kansas City, MO, IEEE Computer Society Press, Los Alamitos, pp. 190-196, 1991.

[25] David C. Smith, Allen Cypher, and James Spohrer, "KidSim: Programming Agents without a Programming Language", Communications of the ACM, 37(7, pp. 54 - 67), July 1994.

[26] Ken Kahn, "ToonTalk – An Animated Programming Environment for Children", Proceedings of the National Educational Computing Conference, Baltimore, Maryland, June 1995. Extended version in the Journal of Visual Languages and Computing, June 1996.

[27] Vijay Saraswat, Concurrent Constraint Programming, MIT Press, Cambridge, MA, 1993.

[28] Ken Kahn, "Generalizing by Removing Detail", Communications of the ACM, 43(3), March 2000. Extended version in Henry Lieberman, editor, Your Wish Is My Command: Programming by Example, Morgan Kaufmann, 2001.

[29] Leonel Morgado, Maria Cruz, and Ken Kahn, "Working in ToonTalk with 4-and 5-year olds", Proceedings of the IADIS International Conference, e-Society, Vol. II, IADIS, 2003.

[30] Andrew Begel, LogoBlocks: A Graphical Programming Language for Interacting with the World, MIT Advanced Undergraduate Project, http://research.microsoft.com/en-us/um/people/abegel/mit/begel-aup.pdf, May 1996.

[31] Eric Klopfer, Hal Scheintaub, Wendy Huang, Danial Wendel, and Ricarose Roque, "The Simulation

Cycle: combining games, simulations, engineering and science using StarLogo TNG", E-Learning and Digital Media, Volume 6 Number 1, 2009.

[32] Mitchel Resnick, John Maloney, Andrés Monroy-Hernández, Natalie Rusk, Evelyn Eastmond, Karen Brennan, Amon Millner, Eric Rosenbaum, Jay Silver, Brian Silverman, and Yasmin Kafai, "Scratch: Programming for All", Communications of the ACM, Vol. 52 No. 11, Pages 60-67, 2009.

[33] Brian Harvey and Jens Mönig, "Bringing 'No Ceiling' to Scratch: Can One Language Serve Kids and Computer Scientists?", Constructionism 2010 Proceedings, Paris, 2010.

[34] David Wolber, Hal Abelson, Ellen Spertus, and Liz Looney, App Inventor, O'Reilly Press, 2011.

[35] Blockly, https://developers.google.com/blockly/

[36] Mitchel Resnick, Stephen Ocko, and Seymour Papert, "LEGO, Logo, and Design" Children's Environments Quarterly, vol. 5, no. 4, 1988.

[37] The Laboratory Schools LEGO-LOGO Project, http://www.ucls.uchicago.edu/students/projects/1994-95/Lego-Logo/ProjectDescription.html

[38] Idit Harel, Children Designers: Interdisciplinary Constructions for Learning and Knowing Mathematics in a Computer-rich School, Ablex Publishing, 1991.

[39] Yasmin B. Kafai, Minds in Play: Computer Game Design as a Context for Children's Learning, Mahwah, NJ, Lawrence Erlbaum, 1995.

[40] Wikipedia, "Living Books series", https://en.wikipedia.org/wiki/Living_Books_series

[41] Yishay Mor, Richard Noss, Ken Kahn, Celia Hoyles, and Gordon Simpson, "Thinking in Progress", Micromath, The Association of Teachers of Mathematics, 20(2), pp.17-23, 2004.

[42] Ken Kahn and Howard Noble, "The Modelling4All Project – A web-based modelling tool embedded in Web 2.0", Proceedings of Constructionism 2010, Paris, August 2010.

[43] MIT Media Lab, "Scratch statistics", http://scratch.mit.edu/statistics/

[44] BBC News, "What it's like to have schizophrenia", http://news.bbc.co.uk/1/hi/health/6453241.stm

[45] One Laptop per Child, http://one.laptop.org/

[46] Ken Kahn, "ToonTalk Reborn, Re-implementing and re-conceptualising ToonTalk for the Web", Proceedings of Constructionism 2014, Vienna, August 2014.

[47] Khan Academy, https://www.khanacademy.org/

[48] Khan Academy, "Computer programming", https://www.khanacademy.org/computing/computer- programming

[49] Wolfgang Slany, "Tinkering with Pocket Code, a Scratch-like programming app for your smartphone", Proceedings of Constructionism 2014, Vienna, August 2014.

[50] Luc Steels (ed.) Music Learning with Massive Open Online Courses (MOOCs). IOS Press, Amsterdam. 2015.

[51] Wikipedia, "Massive Open Online Course", https://en.wikipedia.org/wiki/Massive_open_online_course

[52] Wikipedia, "Alan Kay", https://en.wikipedia.org/wiki/Alan_Kay