Papert's Legacy: Logo, Legos, and Playful Learning

Conference Paper · May 2017

CITATIONS
0

READS
1,397

1 author:

Jason Powell
University of North Texas
8 PUBLICATIONS **1** CITATION

SEE PROFILE

Papert's Legacy: Logo, Legos, and Playful Learning

Jason Powell
The University of North Texas
JasonPowell2@my.unt.edu

Abstract: Teaching computer programming to K-12 students in the Unites States has been promoted and implemented with varying levels of enthusiasm over the last four decades. It is possible to interpret recent attempts to encourage programming as a learning activity as an attempt by the high-tech industry to meet workforce requirements, but examining the origins as programming as an educational activity reveals a more complex picture. This paper examines how Seymour Papert attempted to transform the educational system with the Logo programming language and draws connections with modern graphical programming environments and the maker movement. Papert's constructionism has proven to be quite resilient for a learning theory that has proved so difficult to operationalize as a learning technology.

Papert's Legacy: Logo, Legos, and Playful Learning

Estimates of job demand for skilled computer programmers indicate that by the year 2022, over 1.3 million new openings will exist (U.S. Bureau of Labor Statistics 2013). A Gallup survey commissioned by Google (2015) found that—despite significant interest on the part of parents—many public schools believe that computer science as a core subject requires teachers they cannot afford and represents a distraction from core subject areas. Adding to the problem is the fragmented and inconsistent computer science teacher certification standards for K-12 schools which create a frustrating path for qualified teachers to find employment in the few schools that do offer a dedicated computer science curriculum (Computer Science Teachers Association 2013).

Technology companies are understandably vested in cultivating a skilled workforce, but the history of computer science in K-12 is an interesting manifestation of what Papert (1997) has described as public school's defense mechanism against systemic change. Despite years of investment in computer technology and rapid adoption of new technologies in the public sector, attempts to shift pedagogy in the American educational have consistently failed to result in meaningful change (Cuban 2013). Serious interest and achievement gaps in STEM subjects have been noted among female and minority students once they reach middle school (King 2014), so programming continues to be a promising way to introduce young learners to math and science subjects via the computer (Amador & Soule 2015). Arguments for the importance of cultivating what has been labeled computational thinking stress the importance of exposing students to the problem-solving and algorithmic thinking found in programming (Wing 2006). This paper examines the past and present of computer programming in K-12, from the work of Seymour Papert to the modern graphical programming environments of Scratch and Alice.

Seymour Papert and Logo

Seymour Papert's *Mindstorms: Children, Computers and Powerful Ideas* (1980) came after a decade of research and observation of children using the Logo programming language to explore the language of computer programming. Papert used a robot that he called the Turtle that accepted movement commands and could draw shapes and patterns to help children visualize the results of the code they typed. *Mindstorms* was published in a period of intense interest in personal computers and the role they might play in education, with companies like Apple, IBM, and Texas Instruments eager to deploy their machines in classrooms, with Logo as one of the few educational applications available at the time (Chakraborty, Graebner, & Stocky 1999). Papert saw Logo and the Turtle as an opportunity for children to engage in novel interactions with the computer—breaking down barriers between children and the language and customs of science and mathematics. Papert's observations led him to believe that using Logo held the

potential to unlock concepts that had previously been considered too advanced for younger learners (Blikstein 2013).

The enthusiasm for Papert's vision in *Mindstorms* (1980) and the use of Logo in education would falter towards the end of the decade. Evidence for the effectiveness of a Logo curriculum did not materialize in the form of empirical observation despite rigorous attempts to find appropriate methodologies to determine how working with Logo might contribute to learning (Pea 1987; Littlefield, Delclos, Bransford, Clayton & Franks 1989). Papert characterized the research results on Logo as "technocentrism" that focused exclusively on the use of a computer or the Logo language rather than the more fundamental departure from traditional pedagogy he had envisioned (Papert 1987, p. 23). Much later, Papert would muse that the lack of evidence, "only proved that under the particular conditions of that experiment, nothing happened that could be measured by the particular tests used" (Papert 2000, p. 729). This tension between Papert's revolutionary ideas for education and inconclusive attempts to measures its effectiveness would come to define Logo as an educational technology.

Through his work with children and Logo, Papert extended the constructivist ideas of Piaget into a new learning theory he labeled constructionism (Papert & Harel 1991). Constructivism theorized learning as a function of creating and comparing internal knowledge schemas, while Papert's constructionism focused on the external products of the learner's thought processes in the form of the Turtle's movements and onscreen graphics (Papert & Harel 1991). Constructionism also has connections to other learning theories such as the zone of proximal development, legitimate peripheral participation, and situated learning in that it emphasizes participation, collaboration, and application of knowledge (Ackermann 2001).

The LEGO Connection

Building on the idea of the original Turtle device, the marriage of Logo programming with the LEGO universe of toys was an important development. The Programmable Brick, created by Michael Resnick and others at MIT's Media Lab in 1985, embedded computer circuits into a module that enabled autonomous operation of robotic creations ("Logo history" 2015). This would result in a fruitful partnership with the LEGO company to provide a wide variety of attachments such as motors, lights, sensors, and other options ("Logo history" 2015). The first LEGO Logo kits were made available to educational institutions, but the LEGO company would eventually release commercial LEGO Mindstorm™ kits in 1998 (MIT Media Lab n.d.). Research suggests that robotic manipulation provides a fertile environment for young learners to explore and observe mathematics and science (Bers 2008).

Graphical Programming Environments

Another contribution by Resnick and MIT's Media Lab was the creation of Scratch, a programming environment that integrated a sprite-based animation and drawing stage with a drag-and-drop programing environment that used block shapes similar to LEGO bricks to represent programming commands instead of text (Maloney, Resnick, Rusk, Silverman & Eastmond 2010). Scratch was designed as an introduction to programming concepts for children 8-12, with special consideration given to ease of use and creativity (Maloney et al. 2010). Blocks in Scratch represent: (a) commands such as move, rotate, wait, and play; (b) functions which return values such as screen coordinates; (c) triggers which can execute single or groups of commands; and (d) control structures, which provide for conditional operations like if/then and while loops (Maloney et al. 2010). The environment does not require compiling of code in order to facilitate immediate feedback, and the code blocks only fit together in certain ways, guiding young users to proper grouping to achieve desired behaviors onscreen (Maloney et al. 2010). Scratch was also designed with a cloud-based system for sharing and remixing of projects, adding a collaborative social dimension to learners' experience (Maloney et al. 2010).

Scratch has been used to teach basic programming and game design concepts (Ouahbi, Kaddari, Darhmaoui, Elachqar & Lahmine 2015), problem-solving skills (Gülbahar & Kalelioğlu 2014), and geometry concepts (Smith & Neuman, 2014). Scratch's ability to sequence drawn or imported graphics and sound also make it ideal for digital storytelling, which allows learners to create personal multimedia artifacts around a wide range of subject areas (Lamb & Johnson 2011). ScratchJr is a simplified version of Scratch optimized for use by ages 5-7 on tablet and

touchscreen devices, opening the possibility for an even earlier introduction to the environment (Flannery et al. 2013).

Another graphical programming environment, Alice, was created at Carnegie Mellon University with the addition of a three-dimensional workspace suitable for creating virtual world projects (Cooper, Dann & Pausch 2000). Alice is based on the Python programming language and is targeted at learners at the high-school and undergraduate level as an introduction to computer programming methodology (Cooper et al. 2000). The added complexity of a Z-axis in the 3d environment of Alice presents a higher barrier to entry for students, but results in more sophisticated storytelling and game design experiences (Utting, Cooper, Kölling, Maloney & Resnick 2010). A version of Alice has also been studied as a digital storytelling tool meant to provide middle school girls a positive first experience with computer programming (Kelleher 2006).

Beyond the Screen

The popularity of the LEGO Mindstorms™ product line was undeniable, with 7800 teams participating in the first Lego League robotics competition, although 70% of the participants were male and white (Kafai & Burke 2014). New innovations such as electronic textiles—traditional garments infused with lights, sensors, and speakers—hold the promise of broader appeal than robotics, and open-source Arduino circuit boards further extend the possibilities for young inventors to program and connect their devices in innovative ways (Kafai & Burke 2014). The popularity and diversity of maker spaces in and out of school libraries further validate the appeal of the Papert's collaborative, creative vision of constructionism (Moorefield-Lang 2014).

Conclusions

Allowing young learners to experiment, tinker, and share both digital and tangible artifacts casts a wide net among children to identify and encourage future scientists and programmers. Despite the initial failure to operationalize Logo programming into a learning technology, the true believers in Papert's work persist in their efforts to get children thinking about thinking and falling in love with learning. This is clearly more than an effort to turn public schools into vocational training programs for the tech industry, rather the hope is to show kids a way to learn apart from the system of standardized curriculums and assessments. Papert's vision was for a learning environment where applied knowledge and creativity would afford children the confidence to see themselves as part of the world of science.

References

Ackermann, E. (2001). Piaget's constructivism, Papert's constructionism: What's the difference. *Future of Learning Group Publication*, *5*(3), 438.

Aho, A. V. (2012). Computation and computational thinking. *The Computer Journal, 55*(7), 832–835. http://doi.org/10.1093/comjnl/bxs074

Akcaoglu, M. (2014). Learning problem-solving through making games at the game design and learning summer program. *Educational Technology Research and Development*, *62*(5), 583–600. http://doi.org/10.1007/s11423-014-9347-4

Al-Jarrah, A., & Pontelli, E. (2014). "AliCe-ViLlagE" Alice as a collaborative virtual learning environment. In *Frontiers in Education Conference (FIE), 2014 IEEE* (pp. 1–9). IEEE. Retrieved from http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=7044089

Amador, J. M., & Soule, T. (2015). Girls build excitement for math from Scratch. *Mathematics Teaching in the Middle School*, *20*(7), 408–415.

Atmatzidou, S., & Demetriadis, S. (2015). Advancing students' computational thinking skills through educational robotics: A study on age and gender relevant differences. *Robotics and Autonomous Systems*. http://doi.org/10.1016/j.robot.2015.10.008

Bers, M. U. (2008). Using robotic manipulatives to develop technological fluency in early childhood. In O. N. Saracho & B. Spodek (Eds.), *Contemporary Perspectives on Literacy in Early Childhood Education* (pp. 105–225). Greenwich, CT: IAP.

Blikstein, P. (2013). Seymour Papert's legacy: Thinking about learning, and learning about thinking. Retrieved November 17, 2015, from https://tltl.stanford.edu/content/seymour-papert-s-legacy-thinking-about-learning-and-learning-about-thinking

Chakraborty, A., Graebner, R., & Stocky, T. (1999). Logo: A project history. Retrieved from http://web.mit.edu/6.933/www/LogoFinalPaper.pdf

Chang, C.-K. (2014). Effects of Using Alice and Scratch in an Introductory Programming Course for Corrective Instruction. *Journal of Educational Computing Research*, *51*(2), 185–204. http://doi.org/10.2190/EC.51.2.c
Computer Science Teachers Association. (2013). Bugs in the system: Computer science teacher certification in the U.S. Retrieved from http://csta.acm.org/ComputerScienceTeacherCertification/sub/CSTA_BugsInTheSystem.pdf

Cooper, S., Dann, W., & Pausch, R. (2000). Alice: a 3-D tool for introductory programming concepts. In *Journal of Computing Sciences in Colleges* (Vol. 15, pp. 107–116). Consortium for Computing Sciences in Colleges. Retrieved from http://dl.acm.org/citation.cfm?id=364161

Flannery, L. P., Silverman, B., Kazakoff, E. R., Bers, M. U., Bontá, P., & Resnick, M. (2013). Designing ScratchJr: Support for early childhood learning through computer programming. In *Proceedings of the 12th International Conference on Interaction Design and Children* (pp. 1–10). ACM. Retrieved from http://dl.acm.org/citation.cfm?id=2485785

Gülbahar, Y., & Kalelioğlu, F. (2014). The effects of teaching programming via Scratch on problem solving skills: A discussion from learners' perspective. *Informatics in Education-An International Journal*, *13*(1), 33–50.
Kafai, Y. B., & Burke, Q. (2014). *Connected Code : Why Children Need to Learn Programming*. Cambridge, MA: The MIT Press.

Kelleher, C. (2006). *Motivating programming: Using storytelling to make computer programming attractive to middle school girls* (Doctoral Dissertation). Carnegie Mellon University. Retrieved from http://oai.dtic.mil/oai/oai?verb=getRecord&metadataPrefix=html&identifier=ADA492489

King, D. (2014). Tech camps, other programs hope to keep girls interested in STEM fields. Retrieved December 2, 2015, from http://www.baltimoresun.com/news/maryland/bs-md-women-in-stem-20140725-story.html
Lamb, A., & Johnson, L. (2011). Scratch: Computer programming for 21st-century learners. *Teacher Librarian*, *38*(4), 64–68.

Littlefield, J., Delclos, V. R., Bransford, J. D., Clayton, K. N., & Franks, J. J. (1989). Some Prerequisites for Teaching Thinking: Methodological Issues in the Study of LOGO Programming. *Cognition and Instruction*, *6*(4), 331–366.

Logo history. (2015). Retrieved November 17, 2015, from http://el.media.mit.edu/logo-foundation/what_is_logo/history.html

Maloney, J., Resnick, M., Rusk, N., Silverman, B., & Eastmond, E. (2010). The Scratch Programming Language and Environment. *ACM Transactions on Computing Education*, *10*(4), 1–15. http://doi.org/10.1145/1868358.1868363

Mather, R. (2015). A mixed-methods exploration of an environment for learning computer programming. *Research in Learning Technology*, *23*(0). http://doi.org/10.3402/rlt.v23.27179

MIT Media Lab. (n.d.). LEGO's mindstorms. Retrieved December 2, 2015, from http://media.mit.edu/sponsorship/getting-value/collaborations/mindstorms

Moorefield-Lang, H. M. (2014). Makers in the library: case studies of 3D printers and maker spaces in library settings. *Library Hi Tech*, *32*(4), 583–593. http://doi.org/10.1108/LHT-06-2014-0056

Ouahbi, I., Kaddari, F., Darhmaoui, H., Elachqar, A., & Lahmine, S. (2015). Learning Basic Programming Concepts by Creating Games with Scratch Programming Environment. *Procedia - Social and Behavioral Sciences*, *191*, 1479–1482. http://doi.org/10.1016/j.sbspro.2015.04.224

Papert, S. (1980). *Mindstorms: Children, Computers, and Powerful Ideas*. New York: Basic Books.

Papert, S. (1987). Computer criticism vs. technocentric thinking. *Educational Researcher*, *16*(1), 22. http://doi.org/10.2307/1174251

Papert, S. (2000). What's the big idea? Toward a pedagogy of idea power. *IBM Systems Journal*, *39*(3.4), 720–729.

Papert, S., & Harel, I. (1991). Situating constructionism. *Constructionism*, *36*, 1–11.

Prensky, M. (2008). Students as designers and creators of educational computer games: Who else? *British Journal of Educational Technology*, *39*(6), 1004–1019. http://doi.org/10.1111/j.1467-8535.2008.00823_2.x

Smith, C. P., & Neuman, M. D. (2014). Scratch it out! Enhancing Geometrical Understanding. *Teaching Children Mathematics*, *21*(3), 185–188. http://doi.org/10.5951/teacchilmath.21.3.0185

Sykes, E. R. (2007). Determining the effectiveness of the 3D Alice programming environment at the computer science I level. *Journal of Educational Computing Research*, *36*(2), 223–244.

Tsalapatas, H., Heidmann, O., Alimisi, R., & Houstis, E. (2012). Game-based programming towards developing algorithmic thinking skills in primary education. *Scientific Bulletin of the Petru Maior University of Targu Mures*, *9*(1). Retrieved from http://scientificbulletin.upm.ro/papers/2012-1/GAME-BASED%20PROGRAMMING%20TOWARDS%20DEVELOPING%20ALGORITHMIC%20THINKING%20SKILLS%20IN%20PRIMARY%20EDUCATION.pdf

Turkle, S., & Papert, S. (1990). Epistemological Pluralism: Styles and Voices within the Computer Culture. *Signs*, *16*(1), 128–157.

U.S. Bureau of Labor Statistics. (2013, December). Occupational employment projections to 2022 : Monthly labor review. Retrieved November 16, 2015, from http://www.bls.gov/opub/mlr/2013/article/occupational-employment-projections-to-2022.htm

Utting, I., Cooper, S., Kölling, M., Maloney, J., & Resnick, M. (2010). Alice, Greenfoot, and Scratch -- A Discussion. *ACM Transactions on Computing Education*, *10*(4), 1–11. http://doi.org/10.1145/1868358.1868364
Wing, J. M. (2006). Computational thinking. *Communications of the ACM*, *49*(3), 33.

Yadav, A., Burkhart, D., Moix, D., Snow, E., Bandaru, P., & Clayborn, L. (2015). Sowing the Seeds: A Landscape Study on Assessment in Secondary Computer Science Education. *Comp. Sci. Teachers Assn., NY, NY*. Retrieved from http://csta.acm.org/Research/sub/Projects/ResearchFiles/AssessmentStudy2015.pdf