Programming Standing Up: Embodied Computing with Constructionist Robotics

Matthew Berland, *matthew.berland@utsa.edu*
Dept. of Interdisciplinary Learning & Teaching, University of Texas at San Antonio

Taylor Martin, *taylormartin@mail.utexas.edu*
Dept. of Curriculum & Instruction, University of Texas at Austin

Tom Benton, *soibois@hotmail.com*
Dept. of Curriculum and Instruction, University of Texas at Austin

Abstract

IPRO (shown below) is a mobile, constructionist, virtual robotics programming environment designed to teach computational literacy. Students use a simplified programming language to control the behavior of a robot agent as it undertakes various collaborative and competitive tasks. Each student programs his/her own robots using a handheld device, and the robots compete on a shared stage. The project uses handhelds to encourage collaboration and embodied cognition through physical movement and the sharing of content while programming. The design is based on other robotics environments for teaching introductory programming in which there have been measurable learning gains (Berland, 2008; Martin, 2007; Wilensky & Stroup, 1999a). IPRO is structured as a participatory simulation in that students will participate in a constructionist shared space (see Wilensky & Stroup, 1999b).
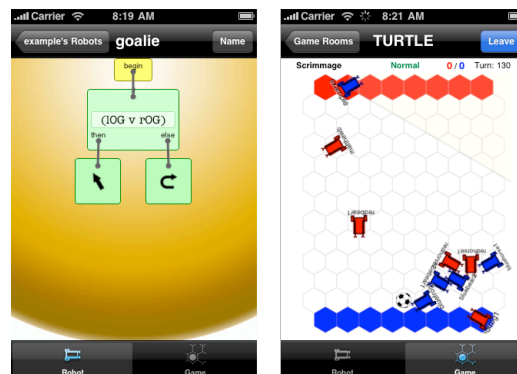
Figure 1 - IPRO Circuit and Play Modes

Constructionist programming tasks are especially well suited to this type of collaboration because they involve complex, concrete tasks with a strong connection of the conceptual and the technical. Students collaboratively refine their conceptual understanding by freely sharing their technical products. Our goal is that students in the project will approach programming as an active, physical task, and that they will be motivated to engage with the content as a result. The design of IPRO is focused on determining pathways to stronger understanding of programming content as well as to levels of increased motivation.

Keywords (style: Keywords)

Embodied cognition; robotics; collaborative; handheld; programming; computational literacy

Introduction

Given the computer's iconic status as the quintessential amodal, decontextualized thinking device, discussing embodied programming is a bit vague at best. However, if we are to understand computational thinking (CT) and programming from a constructionist perspective, considering the implications of embodied programming is essential.

There is a profound disconnect between the excitement surrounding the potential of computer-proficient youth and the actual act of programming a computer. Typically a solitary art, it often invokes the stereotypical image of a young man hunched over a glowing screen, and the path to programming proficiency is frequently presumed to demand monk-like devotion to the study of arcane syntax and keystrokes. Even for computationally literate young people who happily use software in imaginative and innovative ways, learning programming may hold little appeal when pursued in solitude. Our challenge is clear: how can programming in the classroom be recast as engaging, inclusive, and social?

While instruction in many disciplines has been made more active, mobile, and collaborative, programming instruction has resisted those advances (Ben-Ari, 2001). The goal of this project is to introduce an element of mobility into programming instruction and encourage students to collaborate as they move and congregate. We call this "Programming Standing Up". The importance of engaging and effective programming instruction cannot be understated: programming is "hard to learn" (Guzdial, 2004), a core thinking skill (diSessa, 2000), and a core job skill. Some benefits of mobility with constructive work are also well established – Klopfer, Squire, & Jenkins (2002) showed that students engage in more complex problem solving about real world content when they are able to work together in a physical space.

Constructionist programming tasks are especially well suited to this enhanced collaboration because they involve complex, concrete tasks with a strong connection between the conceptual and the technical. Students collaboratively refine their conceptual understanding by freely sharing their technical products. Pedagogy in programming classes has often valued reproduction of 'authentic' professional practice over innovative pedagogy (Ben-Ari, 2001). However, as we know from Smith, diSessa, & Roschelle (1993), authentic demonstration of expert practice is not necessarily the best path to understanding. Our goal is that students in the project will approach programming as an active, physical task, and that they will be motivated to engage with the content as a result.

To that end, we are developing, implementing, and deploying a constructionist mobile, collaborative programming platform focusing on the design, generation, and evaluation of algorithmic knowledge, strategies, and models; these are the basic elements of computer science education as described by Robins, Roundtree, and Roundtree (2003).

IPRO (for both "i<u>P</u>od <u>R</u>obotics" and "<u>I</u> (can) <u>pro</u>gram!") is a virtual robotics environment that builds upon the previous designs of the authors, along with previous research on the development of novice programming environments (a more detailed description is below). Students use a robust programming language to control the behavior of a robot agent as it undertakes various collaborative and competitive tasks. The design is based on other robotics environments for teaching introductory programming in which there have been measurable learning gains (Martin, 2007). IPRO is structured as a participatory simulation in that students will participate in a constructionist shared space (see Wilensky & Stroup, 1999b, for more detail).

The research design around IPRO is focused on determining pathways to stronger

understanding of programming content as well as to levels of increased motivation.

Our primary hypotheses are:

1. Learning to program on mobile devices will lead to a greater incidence of on-task interaction by students with their peers as well as with the world around them.
2. These new interactions will lead to an operationalized understanding of key computational concepts, and will flatten the learning curve as students move on to new programming environments.
3. These new interactions will also lead to increased student engagement with programming.
4. The process of learning to program standing up will be dramatically different from learning to program at a stationary computer.

Constructionist Pathways to Computational Literacy

The term *computational literacy* has often been used to describe proficient usage of a few standard desktop computer applications. For example, as a secondary school teacher, one author taught a class called "computer literacy," which focused on Microsoft Word, Excel, and PowerPoint. Papert (1980) argues that computational literacy should instead mirror 'print literacy' more closely. The modern conception of print literacy does not stop at reading - the print literature individual should be able to express herself in writing as well. As such, computational literacy should not stop at the ability to use computer software; rather, it should include the ability to create and/or manipulate computer software (or hardware) to communicate and disseminate ideas. The expressive and authoring aspects of computational literacy have been long ignored in the pre-collegiate curriculum, but a shift towards a more participatory picture of media and technology education is underway; this shift is supported by research while reflecting changing relationships between young people and technology (Jenkins, 2006).

Much of the modern constructionist work uses computational literacy as a focus to teach complex content (e.g., Blikstein & Wilensky, in press). However, different forms of content and different sets of students are amenable to different approaches, and teaching 'deep' computational literacy is not identical to using computational literacy to teach other content (such as mathematics). How do we help students develop this type of deep computational literacy? This is a fundamental question of constructionism. Embodied cognition provides one key to answering this question.

Embodied Cognition and IPRO

Embodied cognition recognizes and tries to understand how being in a body and interacting with a physical world shapes and impacts the development of thinking, problem solving, and learning. The typical picture of programming alluded to earlier, the solitary male hunched over a computer monitor, seems the complete antithesis of an embodied activity. However, programmers often describe the computer as an extension of themselves, as a highly responsive tool for interacting with the world (diSessa, 2000; A. Randall, writer of Perl 6 (O'Reilly), personal communication, 2010). While we will not turn our students into expert programmers overnight, we can help students develop programming experience in ways that might lead to this type and degree of proficiency.

In placing the programming environment on a mobile device, which a student uses while standing up in a group of other students, we change fundamental aspects of the system characterizing the relationship between the learner and what is to be learned. That system now

includes new pathways, such as moving one's own body or other objects in the classroom, the ability to easily show other students what is happening with your simulation or with the code underneath it, and the ability to organize in groups that afford more direct sociability than if students (or pairs) are seated at individual computers. Considering some of the main themes of embodied cognition can help us identify implications of this changed system. If cognition evolves from perception and action (Anderson, 2007; Fischer, 1980; Seitz, 2000; Varela, Thompson, & Rosch, 1993; Wilson, 2002), programming a robot will be easier if one can attempt to embody a robot during the process. This relies in large part on possibilities for off-loading some cognitive work to the environment (Hutchins, 1995; Kirsh, 2008); in this case that might include acting out what one thinks certain commands or combinations of commands might do, potentially leading to better programming and clearer communication between students about programming. Embodied cognition claims that much less of what we do is guided by plans made ahead of time, and more is guided by on-the-fly tightly coupled act-plan cycles (Suchman, 1988). If so, the mobile programming environment affords exactly this type of work; fresh ideas can be explored in physical space, programmed, and evaluated in virtual space without interruption. This tight coupling of the physical and the virtual is important to have in mind while viewing this project through the lens of embodied cognition. The authors have done significant research on the effects of virtual versus physical environments on instruction in programming as well as mathematics problem solving and have found unique benefits in both cases. For example, virtual environments can help struggling children learn mathematics faster because children can see any or all steps of a problem solution as many times as they like (Martin & Schwartz, 2005). Meanwhile, working in the physical environment is often linked to deeper conceptual processing and understanding (Penner, Lehrer, & Schauble, 1998). We hypothesize that the IPRO environment could capitalize on the benefits of both these approaches.

Why Constructionist Virtual Robotics?

Constructionist research suggests that IPRO will be beneficial for learning because it approaches learning as an active participant (Harel & Papert, 1990); embodied cognition research suggests how Programming Standing Up could change the nature of the programming task to make it more accessible. But why do this programming in the context of robotics?

Strong Support for Constructionism

Robotics have been associated with constructionist approaches since its beginnings (Papert, 1980). It continues to be a core element of many constructionist projects (e.g., Hancock, 2003; Portsmore, 1999; Resnick & Ocko, 1991). As Wilensky (2000) notes, tools that utilize the individual components needed to complete the aggregate can both help the student understand the final concept but also allow the investigator to understand the process of learning. Wilensky differentiates "black box" projects in which subjects begin in the middle of the process of creation with "glass box" projects in which subjects can see the process of creation from start to finish. Programmable autonomous robotics curricula provide a consummate example of this kind of "glass box" work.

Significant Evidence to Suggest Learning Benefits for Math and Science

As part of a broader initiative to improve math and science education, several thousand schools have implemented robotics classes and clubs for K-12 students. Anecdotal evidence about how robotics classes have improved student interest level, creativity, and reasoning skills are well documented (Genalo & Gilchrist, 2006; Lau, McNamara, Rogers, & Portsmore, 2001). Recently, work on the specific benefits of robotics and LEGO-like toys has been materializing. Using a simulated standardized math test, Lindh and Holgersson (2007) tested 996 fifth and ninth grade students in Sweden. They found that, for students who were slightly below average in math, taking robotics in 5th grade improved math test scores relative to their counterparts who did not

take robotics. Wolfgang, Stannard, and Jones (2003) followed a group of 27 students from pre-K through 12th grade and discovered several significant correlations. Pre-K students who were adept with (non-robotic) LEGO blocks went on to score high on standardized high school tests, took more honors and higher math courses, and had a higher weighted grade point average in math courses.

Access issues

On one hand, the immediacy and physicality of robotics would appear to make it a natural context for teaching programming. Coded commands can instantly be visualized as a simulation during testing – following a successful test, students can program their actual robot and get the satisfaction of seeing their programming "come to life." However, robotics is sometimes characterized as having narrow appeal, primarily to the types of students already sitting in the seats of our college engineering classrooms. As such, an important element of our research agenda is examining how IPRO can increase participation of underrepresented groups in engineering. Considering girls as one of these underrepresented groups, an examination of the literature suggests favorable reading of how girls benefit from robotics courses and competitions. Beisser (2006) reports that girls immersed in a LEGO/Logo environment demonstrate significant improvement in self-efficacy beliefs regarding computer use and their likelihood of being computer professionals in the future. Weinburg and colleagues (2007) show that participation in Botball competitions led to increases in both self-efficacy perceptions and ratings of interest in STEM careers for 7th grade girls. In a qualitative case study of girls in a Botball program, Stein & Nickerson (2004) found that girls were equally as interested in the competitive nature of the game robotics environment as boys and were not driven away by it.

We believe this evidence suggests that the issue of equity and robotics is a real problem and worthy of further study, particularly in light of the widespread use of robotics activities in classrooms and after school clubs.

IPRO: A Constructionist Mobile Programming Environment

IPRO is an iPhone & iPod Touch virtual robotics programming environment and game space that we are developing for this project. It is made up of two fundamental components: a development environment ('the board') for the IPRO language and a shared game space ('the field') where students' virtual robots will coexist.

Elements of IPRO

The Field

The field (shown in Figure 2) is where students' robots compete in teams, collaboratively and competitively accomplishing tasks and working towards goals. Students will each design their own robot on the board based on challenges to come in the field. The simplest field activity is a variant of soccer. Soccer is a game in which several studies have shown success to scaffold students in robotics and computational literacy (Sklar, 2002).
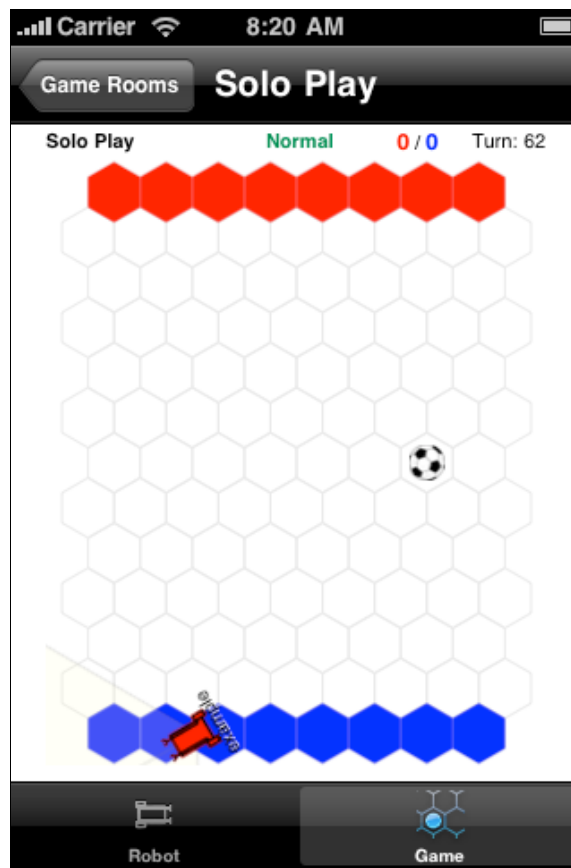
Figure 2 - The IPRO Field

The Board

The board is where students program their robots using single-layer encapsulated functions. The language and function are similar to programming in Scheme, a programming language commonly used for teaching and learning (such as in the textbook by Abelson, Sussman, & Sussman, 1985).

IPRO Activity: IPRO-Soccer

The IPRO framework can be used for a variety of activities, but the initial version is designed to support multi-agent online games of robot soccer. There may be up to 12 robots on the field at any given time, divided equally into red and blue teams as they enter the game. Each team attempts to move the ball into the goal (both shown in Figure 3). Each student designs one robot, but they must work in concert with teammates in order to be successful. The benefits of framing IPRO activities in terms of soccer are:

1. It is a familiar game to many teachers who have used robotics in their classroom support.
2. There is a significant amount of outside information about strategies for both human and robot soccer.
3. There is relatively rapid feedback about the success of a strategy.
4. It scales well with robotics and can be attempted with physical robotics as well (Sklar & Eguchi, 2004).
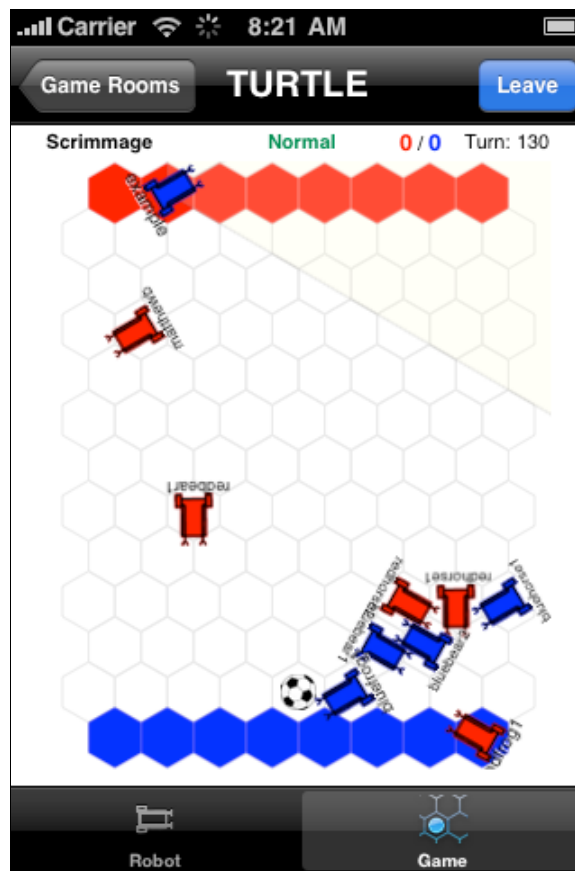
Figure 3 – IPRO Team Play

IPRO Language

The IPRO language is a visual programming language based deriving from the Scheme programming language implementing a simple functional reactive programming paradigm (e.g., Cooper & Krishnamurthi, 2006), in that it uses the concept of signals rather than constants. An example is shown below in Figure 4.
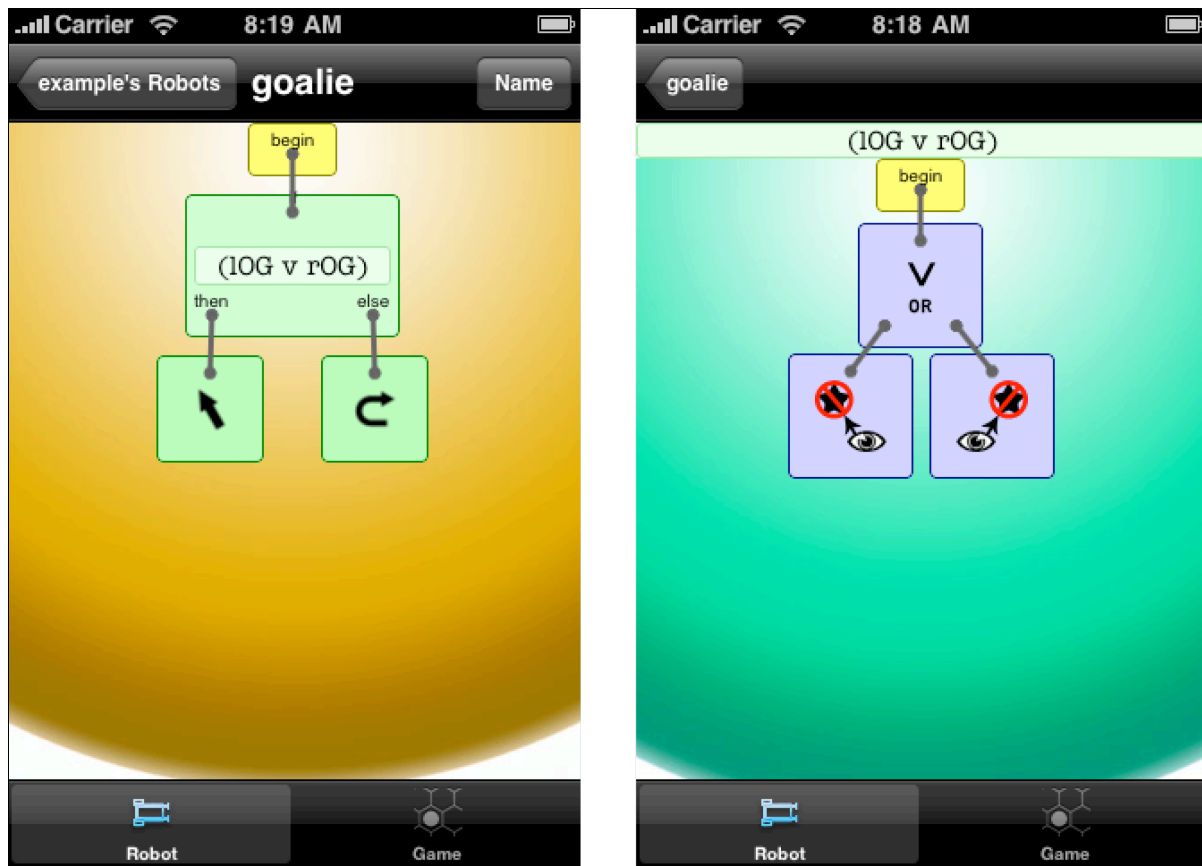
Figure 4 - IPRO Logic Flows

The fundamental components of a program are sensors and motors. Each IPRO virtual robot has a symmetrical left and right version of each sensor, and it can move in one of four directions on a hexagonal grid each time step. The semantics of the language remain fairly simple:

1. During each time step, a set of conditional logic branches is evaluated.
2. Each conditional branch is true or false based on the logic of the sensors. The default set of sensors:
   a. ROBOT-SENSOR = Returns a value that corresponds to an inverse of the distance from the agent-robot to the nearest robot.
   b. GOAL-SENSOR = Returns a value that corresponds to an inverse of the distance from the agent-robot to the agent-robot's targeted goal.
   c. BALL-SENSOR = Returns a value that corresponds to an inverse of the distance from the agent-robot to the nearest ball.
3. The output of the conditional logic must necessarily be some action in the virtual space: MOVE-FORWARD-LEFT, MOVE-FORWARD-RIGHT, MOVE-BACKWARD-LEFT, MOVE-BACKWARD-RIGHT, or TURN-RIGHT.

There are several important differences between IPRO and Scheme.

1. IPRO is significantly simplified, only using those primitives that are relevant to an activity.
2. IPRO is constantly evaluated, so that the student can see the effects of her program immediately and make changes accordingly.
3. No Syntax or symantic errors are possible – all possible programs compile. However, they may not all be relevant to an activity.

Understanding & Modeling the Process of Learning to Program

The IPRO environment presents a unique opportunity to model how students learn and collaborate effectively in the classroom. By collecting data on how students share programs and programming wisdom with each other using their handheld devices, we can, over time, characterize the collaboration that happens as well as create predictive models.

These novel forms and structures of data, along with the analysis, visualization, and modeling techniques we propose, have not heretofore been accessible to school-based studies. Berland (2008) uses multi-agent network theory to illuminate both qualitative/exploratory data and quantitative performance data in mixed-methods studies (using the model from Abrahamson, Blikstein, Lamberty, & Wilensky, 2005), but thus far the resources available thanks to the cyber infrastructure of mobile hardware (GPS, short-range wireless connections, and custom applications) have not been utilized to map and model collaboration.

IPRO collects data not only on students' individual programs but on how they share specific parts of those programs. A concrete example of how this might look in a robot soccer game follows:

1. Juliana builds program to search for the soccer ball. Her program includes three functions, one of which checks to see if the robot is pointed in the correct direction ("correct-direction?").
2. Maria is building a striker, but her striker should only target the correct goal, so she asks Juliana for help. She walks over to Juliana and asks to use her "correct-direction?" function.
3. Juliana shares the function with Maria by utilizing the "share" button on her interface.

All of this data is logged by the system, so we can follow the sharing as it happens see who is collaborating with whom. We can analyze this data statically as well as dynamically.

Conclusions

Like engineering, programming in the classroom does not lend itself to clean cycles of textbook-administered instruction and assessment. However, as computational and systems-thinking skills become increasingly vital, understanding and improving how they are learned has become a necessity. As such, our paper addresses areas of both educational theory and practice. This study is designed to innovate the experimental study of embodied cognition theory as well as use constructionist robotics to better understand the collaborative learning process. Our goal is to reconsider programming instruction and generate testable, predictive models to usefully guide future research. Positive findings will contribute to improving the practice of computer science instruction in the classroom by demonstrating improved learning outcomes in core content areas and presenting programming as an engaging activity for all learners.

References

Abelson, H., Sussman, G.J., Sussman, J. (1985). *Structure and Interpretation of Computer Programs*. New York: MIT Press and McGraw-Hill.

Abrahamson, D., Blikstein, P., Lamberty, K. K. & Wilensky, U. (2005). Mixed-media learning environments. *Paper presented at the annual meeting of Interaction Design and Children 2005*, Boulder, Colorado.

Anderson, M. L. (2007). How to study the mind: An introduction to embodied cognition. In F. Santoianni & C. Sabatana (Eds.), *Brain development in learning environments: Embodied and perceptual advancements*. Cambridge: Cambridge Scholars Press.

Beisser, S. R. (2006). An Examination of Gender Differences in Elementary Constructionist Classrooms Using Lego/Logo Instruction. *Computers in the Schools, 22*, 7-19.

Ben-Ari, M. (2001). Constructivism in computer science education. *Journal of Computers in Mathematics and Science Teaching 20*(1), 45–73.

Berland, M. (2008). *VBOT: Motivating Complex Systems and Computational Literacies in Virtual and Physical Robotics Learning Environments*. Retrieved from ProQuest Digital Dissertations. (AAT 3307005)

Berland, M., & Wilensky, U. (2004). Virtual robotics in a collaborative constructionist learning environment. *The annual meeting of the American Educational Research Association*, San Diego, CA, April 12 - 16, 2004.

Blikstein, P., & Wilensky, U. (in press). MaterialSim: A constructionist agent-based modeling approach to engineering education. In M. J. Jacobson & P. Reimann, (Eds.), *Designs for learning environments of the future: International perspectives from the learning sciences*. New York: Springer.

Cooper, G.H. & Krishnamurthi, S. (2006). Embedding Dynamic Dataflow in a Call-by-Value Language. *15th European Symposium on Programming*, Vienna.

diSessa, A. A. (2000). *Changing Minds: Computers, Learning, and Literacy.* Cambridge, MA: MIT Press.

Sklar, E. and Eguchi, A. (2004). RoboCupJunior -- Four Years Later. *Proceedings of the Eighth International RoboCup Symposium (RoboCup-2004)*

Genalo, L., & Gilchrist, J. (2006). Home Schoolers in an Engineering/Education K12 Outreach Program. *Proceedings of the American Society for Engineering Education Annual Conference and Exposition, 2006*, Retrieved March 1, 2009, from http://soa.asee.org/paper/conference/paper-view.cfm?id=1209.

Guzdial, M. (2004). Programming environments for novices. In S. Fincher and M. Petre (Eds.), *Computer Science Education Research* (pp. 127-154). Lisse, The Netherlands: Taylor & Francis.

Hancock, C. (2003). *Real-time programming and the big ideas of computational literacy.* Unpublished doctoral dissertation, MIT.

Harel, I., & Papert, S. (1990). Software design as a learning environment. *Interactive Learning Environments, 1*(1), 132.

Hutchins, E. (1995). *Cognition in the Wild*. New York: MIT Press.

Jenkins, H. (2006). *Confronting the Challenges of Participatory Culture: Media Education for the 21st Century*. White Paper for the MacArthur Foundation.

Kirsh, D. (2008). Distributed cognition: a methodological note. In Itiel Dror and Stevan Harnad (Eds.) *Cognition Distributed*. Philadelphia: John Benjamins Publishing Co.

Klopfer, E. & Squire, K. & Jenkins, H. (2002). Environmental Detectives: PDAs as a Window into a Virtual Simulated World. *IEEE International Workshop on Wireless and Mobile Technologies in Education*, 2002, Växjö, Sweden. 95-98.

Lau, P., McNamara, S., Rogers, C., & Portsmore, M. (2001). LEGO Robotics in Engineering. *Proceedings of the American Society for Engineering Education Annual Conference and Exposition, 2001*, Retrieved March 1, 2009, from http://soa.asee.org/paper/conference/paper-view.cfm?id=16121.

Lindh, J., & Holgersson, T. (2007). Does lego training stimulate pupils' ability to solve logical problems? *Computers & Education. 49*, 1097-1111.

Martin, F. (2007). Little Robots that Could: How Collaboration in Robotics Labs Leads to Student Learning and Tangible Results. *Intelligent Automation and Soft Computing, 13*(1).

Martin, T. & Schwartz, D.L. (2005). Physically distributed learning: Adapting and reinterpreting physical environments in the development of fraction concepts. *Cognitive Science, 29*(4), 587-625.

NRC (2009). *Report on a Workshop on the Scope and Nature of Computational Thinking.* Retrieved December 15, 2009 from nap.edu: http://www.nap.edu/catalog.php?record_id=12840.

Papert, S. (1980). *Mindstorms*. New York: Basic Books.

Penner, D.E., Lehrer, R., & Schauble, L. (1998). From physical models to biochemical systems: A design-based modeling approach. *Journal of the Learning Sciences, 7*(3&4): 429-449.

Portsmore, M. (1999). ROBOLAB: Intuitive Robotic Programming Software to Support Life Long Learning. *APPLE Learning Technology Review*, Spring/Summer, 1999.

Resnick, M., & Ocko, S. (1991). LEGO/Logo: Learning Through and About Design. In (ed. by I. Harel and S. Papert (Eds.), *Constructionism*. Norwood, NJ: Ablex.

Robins, A., Roundtree, J., & Roundtree, N. (2003). Learning and Teaching Programming: A Review and Discussion. *Computer Science Education, 13*(2), 137-172.

Seitz, J. (2000). The Bodily Basis of Thought. *New Ideas in Psychology, 18*, 23-40.

Sklar, E. & Parsons, S. (2002). RoboCupJunior: a vehicle for enhancing technical literacy. *Proceedings of the AAAI-02 Mobile Robot Workshop*.

Stein, C. & Nickerson, K. (2004). Botball Robotics and Gender Differences in Middle School Teams. *Proceedings of the 2004 American Society for Engineering Education Annual Conference*.

Suchman, L. (1988). Representing practice in cognitive science. *Human Studies, 11*(2-3), 305-325.

Varela, F. J., Thompson, E., & Rosch, E. (1993). *The embodied mind: cognitive science and human experience*. Cambridge, MA: MIT Press.

Wilensky, U. (2000). *Modeling Emergent Phenomena with StarLogoT*. Retrieved December, 2000, from CONCORD.org.

Wilensky U., & Stroup, W. (1999a). *HubNet*. Center for Connected Learning & Computer-based Modeling. Northwestern University. Evanston, IL.

Wilensky, U., & Stroup, W. (1999b). Learning through Participatory Simulations: Network-Based Design for Systems Learning in Classrooms. *Computer Supported Collaborative Learning (CSCL'99)*. Stanford University: December 12 - 15, 1999.

Wilson, M. (2002). Six views of embodied cognition. *Psychonomic Bulletin & Review, 9*(4), 625-636.

Wolfgang, C., Stannard, L., & Jones, I. (2003). Advanced Constructional Play with LEGOs Among Preschoolers as a Predictor of Later School Achievement in Mathematics. *Early Child Development and Care, 173*, 467-475.