# RoboBuilder: A Computational Thinking Game

David Weintrop
dweintrop@u.northwestern.edu
Phone: (734) 417-1704

Uri Wilensky
uri@northwestern.edu
Phone: (847) 467-3818

Northwestern University, 2120 Campus Drive, Evanston, IL 60208

**Abstract**

RoboBuilder is a blocks-based, program-to-play game designed to introduce students to core aspects of computational thinking in a fun and engaging environment. RoboBuilder challenges players to design and implement strategies to make their on-screen robot defeat a series of progressively more challenging opponents. Users play RoboBuilder by inventing and implementing strategies for their robot. The robots then carry out their instructions in an animated battleground. This poster presents the design rationale for RoboBuilder and discusses key aspects of the game that contribute to giving learners a positive, hands-on introduction to core computational thinking skills including computationally expressing ideas, algorithmic thinking, and debugging.

## Significance and Relevance of the Topic

Papert introduced the term computational thinking (CT) in his 1996 paper, arguing that fluency with computation will be necessary and powerful infrastructure for learning [Papert 1996]. In her 2006 article, Jeanette Wing reintroduced the notion of CT and argued that it was a fundamental skill deserving a position alongside reading, writing, and arithmetic as part of the core knowledge one needs to be successful in the 21st century [Wing 2006]. We believe that central to this skill is the ability to translate or encode ideas into representations that leverage computational power. We align the goals for teaching CT with diSessa's notion of computational literacy, where people are not just consumers of computational artifacts, but producers as well, creating what he calls a "two-way literacy" [diSessa 2000]. Accompanying this expressive computational skill, we include abstraction, algorithmic thinking, and debugging as core CT skills.

To accomplish our goal of teaching these skills, we developed RoboBuilder, a program-to-play computational thinking game. RoboBuilder employs a constructionist design [Papert and Harel 1991], challenging players to implement their in-game strategies using a specially designed visual programming language. Thus players must construct working programs to play the game, providing the experience of reifying ideas using a computational medium, a practice central to our notion of CT.

A great deal of research has been done on designing introductory programming environments that make the task of programming more accessible to younger learners as well as more fun and engaging [Kelleher and Pausch 2005; Papert 1980; Wilensky 2001; Repenning et al. 2000]. One approach that has gained popularity in recent years is visual programming environments that utilize a language-primitives-as-puzzle-pieces metaphor, providing visual cues of how commands can be assembled. This approach has been found to be an effective way of allowing novice learners to have early programming successes [Maloney et al. 2008]. To achieve this functionality, we extended the OpenBlocks library [Roque 2007], creating a domain specific set of blocks for the players to use in the game.

RoboBuilder is designed as a game that players must program in order to play. This approach stems from work identifying games as constructive context for learning [Gee 2003]. RoboBuilder adds to the growing list of games-based learning environments that have been designed to teach core computer science, programming and computational thinking concepts [Leutenegger and Edgington 2007; Lee et al. 2012]. RoboBuilder is an extension of Robocode [Nelson 2001], a problem-based learning environment deigned to teach students how to program Java and has been found effective and motivating for students [O'Kelly and Gibson 2006].

## Poster Content

This poster will present RoboBuilder and highlight design features of the game and show how they support computational thinking. A portion of the poster will be dedicated to presenting the game's interface, objective, and programming environment. The majority of the poster will show how different computational thinking skills are enacted during gameplay. For example, we include algorithmic thinking as an important computation thinking skill, so the poster will include a brief description of algorithmic thinking, then include images of programs written by students playing RoboBuilder, as well as quotes from students, showing how the game supported the use algorithmic thinking to accomplish the in-game challenge. Similarly, there will be a section of the poster about debugging, where we will show how the design of the game and its interface facilitate players debugging the strategies they develop. Additionally, at the poster session, the presenter will have a laptop that will allow conference attendees to play RoboBuilder. Finally, the poster will include the URL and a QR code to the website where RoboBuilder is available for free download.

## Citations

DISESSA, A.A., 2000. *Changing minds: computers, learning, and literacy*, Cambridge, MA: MIT Press.

GEE, J.P., 2003. *What Video Games Have to Teach Us About Learning and Literacy*, Palgrave Macmillan.

KELLEHER, C. AND PAUSCH, R., 2005. Lowering the barriers to programming: A taxonomy of programming environments and languages for novice programmers. *ACM Comput. Surv.*, 37(2), pp.83–137.

LEE, T.Y., MAURIELLO, M.L., INGRAHAM, J., SOPAN, A., AHN, J. AND BEDERSON, B.B., 2012. CTArcade: learning computational thinking while training virtual characters through game play. In *Proceedings of the 2012 ACM annual conference extended abstracts on Human Factors in Computing Systems Extended Abstracts*. CHI EA '12. New York, NY, USA: ACM, pp. 2309–2314.

LEUTENEGGER, S. AND EDGINGTON, J., 2007. A games first approach to teaching introductory programming. In *ACM SIGCSE Bulletin*. pp. 115–118.

MALONEY, J.H., PEPPLER, K., KAFAI, Y., RESNICK, M. AND RUSK, N., 2008. Programming by choice: urban youth learning programming with scratch. *ACM SIGCSE Bulletin*, 40(1), pp.367–371.

NELSON, M., 2001. Robocode. *IBM Advanced Technologies*.

O'KELLY, J. AND GIBSON, J.P., 2006. RoboCode & problem-based learning: a non-prescriptive approach to teaching programming. In *Proceedings of the 11th annual SIGCSE conference on Innovation and technology in computer science education*. pp. 217–221.

PAPERT, S., 1996. An exploration in the space of mathematics educations. *International Journal of Computers for Mathematical Learning*, 1(1).

PAPERT, S., 1980. *Mindstorms: Children, computers, and powerful ideas*, Basic books.

PAPERT, S. AND HAREL, I., 1991. Situating constructionism. In I. Harel & S. Papert, eds. *Constructionism*. Norwood N.J.: Ablex Publishing Corp., pp. 1–11.

REPENNING, A., IOANNIDOU, A. AND ZOLA, J., 2000. AgentSheets: End-user programmable simulations. *Journal of Artificial Societies and Social Simulation*, 3(3).

ROQUE, R.V., 2007. *OpenBlocks: an extendable framework for graphical block programming systems*. Massachusetts Institute of Technology.

WILENSKY, U., 2001. Modeling nature's emergent patterns with multi-agent languages. In *Proceedings of EuroLogo*. Linz, Austria, pp. 1–6.

WING, J.M., 2006. Computational thinking. *Communications of the ACM*, 49(3), pp.33–35.