Learning about Complexity with Modular Robots

Eric Schweikardt
Computational Design Lab
Carnegie Mellon University
tza@cmu.edu

Mark D. Gross
Computational Design Lab
Carnegie Mellon University
mdgross@cmu.edu

Abstract

We present progress with roBlocks, a reconfigurable modular robotic system for education. Children snap together small, magnetic, heterogeneous modules to create larger, more complex robotic constructions. The design of the system is described and the algorithms that handle data transfer and manipulation are explained. Users tend to begin exploring the system through a series of simple robot patterns but quickly progress to more involved constructions. Many years before they learn formally about hierarchy and modularity, children can develop intuitions about these concepts by designing modular robots. Additionally, young users often spontaneously engage in creative debugging practices.

1. Introduction

Our world is a confusing and complex place. Global phenomena emerge as the result of interacting local behaviors. Classical notions of causality that have guided Western thought for centuries continue to serve, but we have come to see that complexity plays an important role in our world. Understanding complexity, and learning to think in terms of complexity, is essential to addressing the problems that face humankind in the twenty-first century: global warming, the management of mega-cities, food, energy, and water for a growing population. This is, of course, not a new idea: the "systems thinking" and cybernetics movement of the mid-twentieth century [1-3] and the scientists and mathematicians who worked on understanding the dynamics of systems such as arms races [4], populations [5], and industrial, urban, and world dynamics [6, 7] laid the groundwork.

We have designed and built a robotic construction kit, called roBlocks, which we intend as a vehicle for conveying the fundamental perspective of complexity: that the behavior of a system need not be programmed from the top down, but may result from the interactions of independent components. Our kit is designed for young learners (roughly ages 10-15) who have no particular technical or computer knowledge.

We share this goal with educational computing environments such as StarLogo and NetLogo [8, 9]. In these screen-based environments, children write programs that describe the behavior of individuals (e.g., a bird) and then explore what happens when the individual program is executed by large numbers of individuals (e.g., flocks and flocking behavior). A different approach but with similar goals is simulations such as SimCity or SimEarth. In these games a player must make decisions in a complex and time varying system. The rules of the system are concealed from the player, so the challenge is to achieve goals without knowing how the system works.

The roBlocks project is based on the idea that the acts of designing and building real objects develops creativity and scientific curiosity. Our work is inspired by Seymour Papert's idea of *constructionism*; that building things is a particularly good way to learn since the artifacts are tangible – they can be easily discussed and critiqued [10].

Figure 1. A few roBlocks.

Our approach is inspired by the lovely book *Vehicles*, by Valentino Braitenberg [11]. Braitenberg shows how, by assembling increasingly complex

978-0-7695-3409-1/08 $25.00 © 2008 IEEE
DOI 10.1109/DIGITEL.2008.49

robots out of sensors, effectors, and simple neurons, from the ensemble gradually behavior emerges that seems intelligent. Along similar lines is Brooks's approach to robotics, outlined in his early papers "Intelligence without Reason" and "Intelligence without Representation" [12, 13]. Here again the idea is that rather than resulting from a top-down decision making process, intelligent behavior emerges from communicating local components.

2. The roBlocks construction kit

roBlocks are 40mm plastic cubes that snap together with magnetic connectors [14]. Children as young as nine snap them together to create constructions that drive around on a tabletop, reacting to light and sound. Each roBlock is different. *Sensor* blocks, including specific blocks for sensing light, sound, touch, motion and (infrared) distance, take in data from the environment and pass it on to connected blocks. Multicolored *Think* blocks apply functions to those data including sum, maximum, minimum, inverse and threshold. *Action* blocks translate data passed to them into various types of action. A tread block contains a small motor and drives around on a tabletop according to its given value. Other blocks have rotating faces, bright LEDs and piezoelectric speakers. The fourth block category, *utility* blocks, includes a block containing a small lithium-ion battery that must be included in each construction, a Zigbee wireless block, and passive data-connection blocks that allow the physical form of a construction to be less constrained by its programmatic layout.

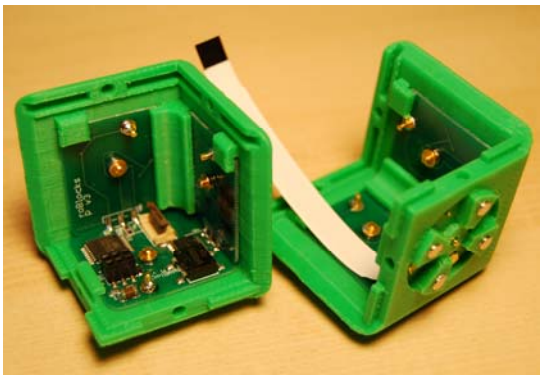

Figure 2. Inside a roBlock.

In a roBlocks construction each block possesses a single dynamic one-byte *value*, which determines how it operates [15]. Sensor blocks compute this value from environmental input. A light sensor block, for example, has a value of about 5 in a dark room, and a value of over 200 outside on a sunny day. A touch sensor has a resting value of zero but jumps to 255 when it detects contact. blocks, on the other, hand, actuate according to their value, which they derive from data passed to them by their neighbors. A *Rotation* block with a value of zero does not move, but the same block with a value of 127 would rotate at half speed. roBlocks pass their values to their connected neighbors. Sensors act as sources and action blocks as sinks, and constructions form an implicit directed graph that may have cycles. The blocks operate asynchronously, transferring data with no centralized clock. Each block's value is determined by the number of steps from each data source in a weighted average. Two sensor blocks at either end of a chain of blocks, for example, create a gradient of block values, with blocks closer to a high sensor reading exhibiting higher values. This weighted averaging scheme allows users to create densely packed 3D lattices of blocks and accurately predict the value at any block.

Each roBlock body is made of two identical three-face halves that screw together enclosing electronics inside. We make the bodies on our 3D printer and using different colors of plastic to indicate type of block. Each face of the blocks is identical, and hermaphroditic connectors allow each block to connect to any other block at any of four possible orientations. Embedded magnets and spring probes on each connector provide both physical and electrical connectivity between blocks. On the back of each connector the magnets are attached with conductive epoxy to the circuit boards shown in Figure 2. Each roBlock has identical electronics: an Atmel AVR microcontroller, programming header, H-bridge motor controller, shift register, and power circuitry.

3. roBlocks robots

The simple robot shown in Figure 3 is built with five roBlocks: two sensor blocks (*Light* and *Knob*), a *Maximum* think block, an LED block and a power block.

Figure 3. A little robot that indicates a number based on the values of two connected sensor blocks.

The network and data flow diagram of this robot is shaped like a "Y", with the two sensor inputs merged at the think block and passing data to the numeric LED block. Any two sensors could be used here – the *Maximum* block will cause the output to correspond to the higher of the two sensor values. As we've chosen a *Knob* as one of our sensors, the user can set its data value manually. With this combination we've created a sort of *Threshold* robot in which the light sensor value is taken into account only if it becomes greater than the value of the *Knob* sensor. Braitenberg [11] describes how a simple threshold device can be a key element in creating lifelike, emergent behaviors.

Figure 4. A mobile robot that steers away from light sources.

The mobile robot shown in Figure 4 is also built with only five blocks: two *Light* sensor blocks, two *Tread* blocks and a power block. Each connects directly to one of the sensors and so responds more powerfully whenever its adjacent sensor is stimulated more powerfully. With two *Tread* blocks, we have created a differential drive robot that turns away from a stimulus, appearing to exhibit the intention of avoiding light. Children as young as six can make the transition from ascribing intent to a robot (e.g., "it doesn't like light") to understanding how its structure could produce an apparently intentional behavior [16].

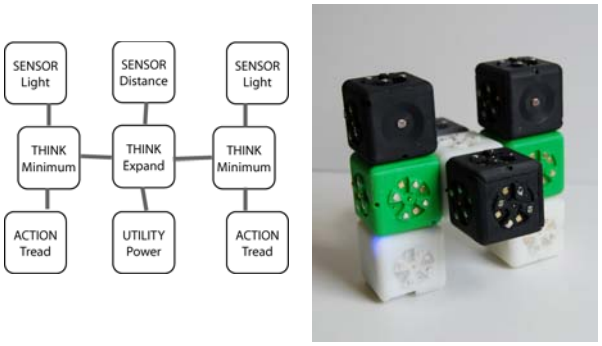

Figure 5. The construction graph and a photo of a robot that avoids falling.

A simple modification illustrates the behavior typical to the *think* category of blocks. The robot shown in Figure 5 adds a third *distance* sensor block in front and pointing down, connected through an *expand* block (which converts a one-byte block value into a binary 0 or 1) and two *minimum* blocks to the drive subassemblies. Normally, the third distance sensor block would output a high value, allowing the robot to operate just like the robot shown in Figure 2. When encountering the edge of a table upon which the robot is moving, however, the *expand* block outputs a zero instead of 1, and the *minimum* blocks immediately stop both drive motors. This robot exhibits interesting and non-linear behavior. Whereas the previous robots suggested a simple action-reaction model, this new robot evokes the notion of rules. Even though we haven't expressed an explicit *if*-then statement ("if the robot sees the edge then stop"), the minimum blocks create the effect of a conditional by using only a simple mathematical calculation.

4. Observations of young users

We recently completed a set of informal user test sessions. Each session began with a quick introduction to the kit and a demonstration of how the blocks operate. We demonstrated a couple of simple robot constructions with a single sensor and single *action* block, and showed how *think* blocks placed in between change the robot's behavior. Then, we observed the groups in free play sessions, answering questions and making suggestions when necessary. Often, we would ask subjects to explain their constructions in an attempt to elicit the understanding and mental models they hold concerning the operation of the blocks.

Session 1: Sunday, March 9, 2008		
Location: Home of User A and B, brothers		
User A	11 (6th grade)	Male
User B	9 (4th grade)	Male
User C	9 (4th grade)	Male
Session 2: Tuesday, April 8, 2008		
Location: Office of User D and F's mother		
User D	7	Male
User E	7	Male
User F	5	Male
Session 3: Friday, April 18, 2008		
Location: Lunch hour at local middle school		
User G	12 (7th grade)	Male
User H	13 (7th grade)	Male
Session 4: Friday, April 18, 2008		
Location: Lunch hour at local middle school		
User I	12 (7th grade)	Male
User J	13 (7th grade)	Male
User K	13 (7th grade)	Male

Session 5: Friday, April 18, 2008		
Location: Lunch hour at local middle school		
User L	13 (7th grade)	Female
User M	13 (7th grade)	Female
Session 6: Wednesday, April 23, 2008		
Location: The girls' home, with their parents		
User N	11 (5th grade)	Female
User O	10 (4th grade)	Female

Table 1. User testing sessions.

We conducted six test sessions over two months, each with two or three children working together in each session. Fifteen children participated: 11 male and 4 female. Most participants were between 9 and 13 years old, but we tested with two 7-year-olds and one 5-year-old. Two test sessions took place at the children's homes, with their parents available and with access to other toys. One session was conducted at the office of the subjects' mother, and three sessions were conducted at a local middle school during lunch hour with children who had expressed interest when asked by the 7th grade science teacher, who also attended the sessions. The remarks in the following sections draw on our observations of children constructing robots.

5. Learning intuition

It is important to distinguish between familiarity with a particular concept and explicit knowledge of a certain representation of that concept. Claims of educational benefit are often made rather spuriously. One might hear that playing with LEGO bricks teaches kids about physics, or, perhaps more plausibly, that playing with Cuisenaire rods teaches kids about math. While balancing a LEGO construction might involve torque and moment arms, and thereby help students build intuitions about mechanics, students aren't exposed to the technical language and formal mathematical representations that we use to convey these ideas.

Take, for example, the concept of *weighted average*, the method each roBlock uses to calculate its one-byte value. Children as young as nine have proven remarkably adept at understanding this calculation, rearranging blocks and noting that data streams are "stronger if they're closer" when mixed. But they are clearly not learning that:

$$\bar{x} = \frac{\sum_{i=1}^{n} w_i x_i}{\sum_{i=1}^{n} w_i},$$

where the x's stand for data values and w's stand for their corresponding weights. Some would claim that familiarity with this equation indicates true understanding of *weighted average*, but this equation is only one representation of the concept. A simple roBlocks robot like that shown in Figure 6 is, in fact, another representation of a weighted average. If the *action* blocks A_0 and A_1 were both *flashlight* or *bar graph* blocks, a user would notice a gradient of values across the construction. The robot shown in Figure 6 produces a stair-step gradient, but if there were more blocks between the sensors, the gradient would be smooth. This pattern, basically diffusion of data values throughout a construction, is not explicitly contained in the weighted average algorithm, but instead emerges from the interactions between modules. Both the equation and the emergent pattern are valid and interesting ways to think about weighted average.

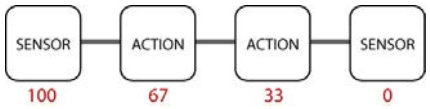

Figure 6. A simple roBlocks robot illustrating the diffusion algorithm, or weighted average.

We don't mean to suggest that fiddling with construction toys should supplant traditional mathematics or science education. Yet we do believe that early exposure to STEM concepts in different representations can improve understanding when the concepts are encountered again later in a formal education environment. Specifically, we posit that beginning to learn to *think computationally* [17] at an early age prepares students well for more advanced subjects in science, technology, and mathematics. In *Changing Minds* [18], diSessa makes a persuasive argument that early bits of encountered knowledge (or "phenomenological primitives") lay the groundwork for successful understanding of scientific concepts later on in life.

6. Concepts of complexity

Complex systems are distinguished from those that are complicated or chaotic. A laptop motherboard is complicated, due to its numerous parts, signal paths and different chips, but it is generally predictable and deterministic in its operation. Turbulent flow, by contrast, is chaotic. Enormous sensitivity to initial conditions makes modeling difficult and prediction almost impossible, even though turbulence is just a natural unfolding of physical laws [19]. Complex systems, the subject of this inquiry, are characterized by numerous components, tightly coupled, that simultaneously carry out their own goals or programs. Complex systems are hard to understand. They often give rise to emergent behavior, larger global patterns

that are not easily reduced to the components and interactions that produce them [20]. We discuss the idea of complexity through the notions of modularity, hierarchy, and emergent behavior.

6.1. Modularity

Modularity is used to describe a multitude of varying situations. We speak of modular housing that is assembled from prefabricated components, or modular programming, where well-defined interfaces separate chunks of computer code. Lipson [21] defines modularity as the "localization of function," and this idea is the core concept of roBlocks. Each function, whether sensing, actuating, or computational, is encapsulated within its own sealed plastic cube, accessible but pre-defined. The interfaces between functions are the magnetic connectors, designed so that any two functions can communicate at any of four orientations. Young users are aware that they can remove and replace blocks at will, removing and replacing the programmatic functions at the same time.

A high level of modularity requires loose coupling between modules. In other words, components that are strongly interdependent with their neighbors are not very modular. roBlocks takes loose coupling to an extreme, with each block maintaining a single, dimensionless data value. This block-level modularity seems to resonate with young users. As the blocks were being described to the users at the beginning of Session 2, a seven-year-old commented that the *bar graph* and *numeric* block were "the same," noticing that they both displayed a graphic readout of their value, albeit in different representations. Later, having built a construction that reacted both to ambient light levels and the presence of nearby objects, he began switching out different blocks, noting that "really, all of the white blocks are the same." While the blocks all have different functions, the modularity and loose coupling of the system create an interface that allows them to be easily substituted for one another.

On a slightly different level, we have been pleased to notice that many users spontaneously begin building *meta-modules*: assemblies of several roBlocks that can be re-used. These are modules of modules, and users seem to build them naturally, in the process of creating a construction in several steps. In Session 6, for example, an eleven-year-old girl combined a sensor, think, and action block to create a simple mobile robot that would slow down as it approached an object. She used this meta-module in several other constructions, determining along the way that connecting it to other blocks by way of a *blocker* (a black *blocker* block transmits power but not data) would ensure that its operation would not be influenced by the other blocks. Eventually, she built a second, identical meta-module, and by attaching the two side-by-side with *blocker* and *power* blocks, created a robot that actively turned toward any object before slowing down.

In many systems, such as homo sapiens or even a PC running Photoshop, behavior can seem somewhat separate from physical structure. People often feel that they are something more than their physical being, that there's an extra little piece of "soul" floating around. And the dividing line between computer hardware and software is strangely dark, with software able to run on many different machine designs. But this is an illusion: bits are bits and neurons are neurons. With roBlocks, however, behavior is directly caused by the physical structure of a construction. It's clear that the design of the robot's body gives rise to its behavior, just as it does in biological systems. Pfeifer and Bongard thoughtfully address this issue in their new book [22]. In a certain sense, systems like roBlocks can encourage critical thinking about traditional ideas of mind-body dualism. This mildly subversive notion isn't a goal of the project, simply a pleasant side effect.

6.2. Hierarchy

Modular systems inspire a vision of several encapsulated black boxes, all communicating through specified interfaces. Often, the modules express a certain regularity—this is certainly the case with most modular robots. In real-world systems, however, things are not so simple: modules and meta-modules exist at many different levels of hierarchy. The complex system of world government, for instance, is made up of unions or alliances, which are made up of countries, which are, in turn, made up of states or districts, and eventually, people. Biological systems, with layers ranging from populations down to cells and chromosomes, are even more complex.

roBlocks, with their regular structure and modular functional breakdown, might appear to be restrictive in modeling hierarchical systems. But several children have used meta-modules in very creative ways. During Session 1, an eleven-year old boy used the *max*, *min*, and *sum* think blocks and a variety of sensors to create a complex, hierarchical input chain to a single *bar graph* (Figure 7). His nine-year-old brother immediately commented that it was "like a ladder," referring to the common tournament scoring system. It required some discussion to determine exactly how data was flowing through the system, and creating a desired output required a certain amount of thinking. Isaac noted that there were many valid input combinations for a particular output value. Manually manipulating the sensor inputs on even this simple

hierarchy demonstrated the difficulty of determining the root cause of an observed outcome.

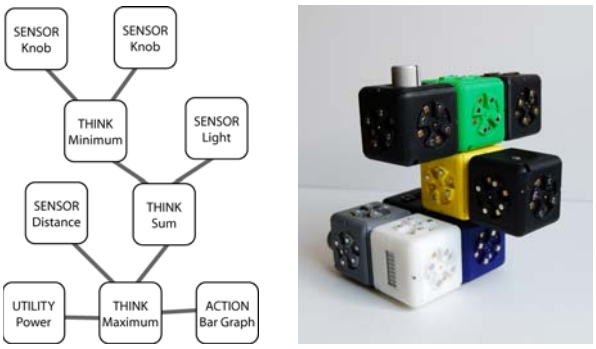

Figure 7. A young user's hierarchical robot.

6.3. Emergence

Most of what we observe in the world was never explicitly programmed. Fluctuations in an economy, for example, emerge from millions of local interactions between its component people, businesses, and governments. Almost all of the important systems in the world exhibit some sort of emergence, and this is precisely what makes them difficult to manage. Climate change, war, food shortages—these are not problems that can be solved by blunt decree. They are problems that demand careful probing of causes, patterns, and concurrency.

Cellular automata are a classic model of emergence; the gliders and glider guns in Conway's *Game of Life* [23] proved a valuable tool for provoking thinking into how intentional-looking behaviors can emerge from several mindless low-level rules. More recently, we have seen several software systems [8, 9] that allow children to create their own rule sets, populating cellular automata worlds with various entities and behaviors, and then watching the ensuing patterns unfold on the computer screen.

While no doubt screen-based simulations can be effective for certain goals, the emergent behavior and patterns displayed in a cellular automata grid are an extreme abstraction from the complexity that we see around us, whether in nature or in societal systems. roBlocks represents an attempt to move these concepts from the computer screen into the real world.

The main challenge children face in assembling a pile of roBlocks into an intended construction involves translating from a *distal* to a *proximal* description [24]. The distal description is the global goal behavior as viewed by an outside observer, like "chase the other robot" or "stop when you see a red object". The proximal description is the actual set of instructions, or "recipe," that gets carried out to achieve that goal. For most robots, the recipe is the programming that coordinates its actions, but when children build with roBlocks, the relevant recipe involves the physical configuration of the individual blocks. This is a subtle difference, but one worth elaborating on.

Adding a roBlock to a construction is not the same as adding another function to a body of computer code. A roBlock does represent a particular function, but in a construction, *all* of the modules communicate with *all* the other modules, *all* the time. So in essence attaching a roBlock is more like adding an ingredient to a recipe – one must be mindful of how it will react with all of the other ingredients, combining to create emergent phenomena.

As emergent behavior is often difficult to predict it can be challenging to design a roBlocks construction that behaves according to a particular high-level goal. Novice users are often confounded as they begin to create large constructions—with small sensor variations affecting every other block, it becomes nearly impossible to understand the control structure of the entire construction.

We often prompt our test subjects to build robots that perform some distal behavior, and it is intriguing to hear them talk through their constructions. For "chase" is not as simple as adding a "chase" block – it is a higher-level behavior that must emerge from low-level, mindless interactions. Users as young as nine have been able to clearly explain how certain blocks communicate and function concurrently in order to perform a certain high level behavior, but most of their robots have been fairly simple, made of fewer than eight roBlocks.

It will be interesting to see how users make sense of larger constructions. The behavior that emerges from 6-10 block constructions can be either comprehensible or confounding, depending on the structure of the robot. Children are quick to use hierarchy and modularity to manage the complexity of their robotic constructions—we are curious to see the building patterns they use when given many more modules to design with.

7. Debugging

In observation we've seen that children are generally capable of building simple robots to meet their design goals, but that a greater number of blocks can begin to confound them. During the first 20 minutes of encountering the system, for example, many users explore the space of constructions that can be built with single *sensor* and *action* blocks: simple robots that respond to a stimulus in a linear fashion.

As they become familiar with the data model, they begin to add more blocks. Since most of the blocks don't explicitly display their data values, system behavior becomes harder to predict. When writing software, the solution to this problem involves various debugging techniques: we've seen children implement a few of these techniques without prompting.

Many users begin to use certain *action* blocks to examine the data values in the system, stepping through their construction and tracing the data flow. The *numeric* and *bar graph* blocks both use arrays of green LEDs to display their value to a user. Snapping one of these blocks onto any other block in the construction displays the data value of that block/ In effect these blocks are probes; the equivalent to printing variable values to the screen when debugging a piece of software. This technique enables kids to step through their constructions and figure out exactly where and how data is changing, so that they can make targeted modifications to their robot.

Most of the roBlocks robots children build are autonomous; their behavior is determined through the sensing of environmental conditions. The absence of a user in control of the robot makes these robots difficult to test. A robot might have *light* and *distance* sensor blocks, for instance, and it may be difficult to manipulate both the room's light level and the position of some other object in order to test the robot's behavior. Some of our young users have gotten around this difficulty by temporarily substituting a *knob* block for a particular sensor. This allows them to easily change the sensor value (by adjusting the potentiometer on the block) and simulate conditions that the original robot may encounter. This practice is similar to the programming technique of hard-coding certain variables during debugging to test the operation of others. Indeed, controlling certain variables in an attempt to examine the behavior of others is the fundamental idea of scientific experiment.

We've been surprised to see many users begin by trying to build complete robots, finally attaching a *power* block when the rest of the blocks are in place. On reflection, however, our experience teaching students to write software has shown us that novice programmers *often* begin by attempting to write a complete program and then run the whole thing. With more experience, programmers learn the value of additive programming; creating functional pieces of code and adding to them makes it much easier to isolate errors. Many of our users have the same experience with roBlocks – they learn that by starting a construction with the *power* block, they can observe the functionality of their robot as it's being constructed, instead of only at the end, when it may have become unwieldy to debug.

While many computer scientists recognize the great value of debugging skills for writing software, we imagine these ideas to be much more broadly applicable. The US National Science Education Standards put forth by the National Research Council [25] describe a general theme of inquiry. Chapter Three of the standards specifically recommends *less* emphasis on "focusing on student acquisition of information" and *more* emphasis on "focusing on student understanding and use of scientific knowledge, ideas, and inquiry processes". The standards also recommend less emphasis on "presenting scientific knowledge through lecture, text, and demonstration" and more emphasis on "guiding students in active and extended scientific inquiry".

8. Conclusions and future work

At DIGITEL2007, we presented a brief survey of toys with multiple nodes of computation [26]. This year, we discuss the ways in which these toys can support notions of how complex systems function. We are currently working on improving the roBlocks kit and creating a programming interface that children can use to modify the behavior of individual blocks.

The informal results and anecdotes we've presented are preliminary, but they give us an idea as to the educational affordances of modular robotic kits like roBlocks. Most importantly, we have seen that by providing children with tools to design and build their own physical, concurrent, complex systems, we can scaffold their understanding of difficult concepts like emergence, modularity, and hierarchy. Although these are not the standard "reading, writing, and arithmetic" subjects of secondary education, issues of complexity are paramount in addressing the problems that will face our society in the generations to come.

Acknowledgements

We thank the children who participated in our study, their parents, and their science teacher, Ms. Katie Levedahl of the Sto Rox School District. Ben Wojtyna, Andrew Jones, and Drew Hendrickson contributed many hours of robot assembly help. This work was supported by National Science Foundation Grant ITR-0326054.

8. References

[1] L. von Bertalanffy, *General System theory: Foundations, Development, Applications*, 1968.

[2] S. Beer, *Platform for Change*. London and New York: John Wiley, 1975.

[3] G. Pask, *An Approach to Cybernetics*: Hutchinson, 1961.
[4] A. Rapoport, *Fights, games, and debates*: University of Michigan Press, 1974.
[5] V. Volterra, "Variations and fluctuations of the number of individuals in animal species living together," in *Animal Ecology*: McGraw-Hill, 1931.
[6] J. W. Forrester, *World Dynamics*: Pegasus Communications, 1973.
[7] J. W. Forrester, "System Dynamics and K-12 Teachers," presented at the University of Virginia school of education, May 30, 1996, 1996.
[8] M. Resnick, "Turtles, Termites, and Traffic Jams." Cambridge, MA: MIT Press, 1994.
[9] U. Wilensky, "Modeling Nature's Emergent Patterns with Multi-agent Languages," presented at Proceedings of EuroLogo 2001, Linz, Austria, 2001.
[10] S. Papert, "Situating Constructionism," in *Constructionism*, I. Harel and S. Papert, Eds. Norwood, NJ: Ablex Publishing Company, 1991, pp. 1-11.
[11] V. Braitenberg, *Vehicles: Experiments in Synthetic Psychology*. Cambridge, MA: MIT Press, 1984.
[12] R. Brooks, "Intelligence Without Reason," presented at Proceedings of the Twelfth International Joint Conference on Artificial Intelligence (IJCAI-91), Sydney, Australia, 1991.
[13] R. Brooks, "Intelligence Without Representation," *Artificial Intelligence*, vol. 47, pp. 139-151, 1991.
[14] E. Schweikardt and M. D. Gross, "roBlocks: A Robotic Construction Kit for Mathematics and Science Education," in *International Conference on Multimodal Interaction*. Banff, Alberta, Canada, 2006.
[15] E. Schweikardt and M. D. Gross, "The Robot is the Program: Interacting with roBlocks," in *TEI 2008: Second International Conference on Tangible and Embedded Interaction*. Bonn, Germany, 2008.
[16] D. Mioduser, S. T. Levy, and V. Talis, "Kindergarten children's perception of robotic-control rules," in *International Conference on the Learning Sciences*. Seattle, WA, 2002.
[17] J. M. Wing, "Computational Thinking," *Communications of the ACM*, vol. 49, pp. 33-35, 2006.
[18] A. A. DiSessa, *Changing Minds: Computers, Learning, and Literacy*. Cambridge, MA, USA: MIT Press, 2000.
[19] J. Gleick, *Chaos: Making a New Science*. New York: Penguin Books, 1987.
[20] J. H. Holland, *Emergence: From Chaos to Order*. Reading, MA: Addison-Wesley, 1998.
[21] H. Lipson, "Principles of modularity, regularity, and hierarchy for scalable systems," *Journal of Biological Physics and Chemistry*, vol. 7, pp. 125-128, 2007.
[22] R. Pfeifer and J. Bongard, *How the Body Shapes the Way We Think: A New View of Intelligence*. Cambridge, MA: MIT Press, 2006.
[23] M. Gardner, "Mathematical Games: The fantastic combinations of John Conway's new solitaire game "life"," in *Scientific American*, 1970, pp. 120-123.
[24] N. E. Sharkey and J. Heemskerk, "The neural mind and the robot," in *Neural Network Perspectives on Cognition and Adaptive Robotics*, A. J. Browne, Ed. Bristol, UK: IOP Press, 1997.
[25] N. R. C. National Committee on Science Education Standards and Assessment, "National Science Education Standards," National Academies Press 1996.
[26] E. Schweikardt and M. D. Gross, "A Brief Survey of Distributed Computational Toys," presented at DIGITEL 2007: The First IEEE International Workshop on Digital Game and Intelligent Toy Enhanced Learning, Jhongli, Taiwan, 2007.