tecnologias, sociedade e conhecimento
vol. 1, n. 1, nov/2013

artigo

Beyond Top Down: Designing with Cubelets

Mark D Gross, (Modular Robotics Incorporated and Carnegie Mellon University) [1]
Christie Veitch, (Modular Robotics Incorporated)

Abstract
Cubelets is a modular robotic building set that embodies a parallel distributed computing model. Unlike central-brain, top-down command-and-control models that conventional robotics construction kits employ, a parallel distributed computing model accounts for emergent phenomena in the world. We compare the top-down and parallel distributed models in the familiar task of constructing a maze-following robot. As we consider how these different robotics models contribute to maze following we also examine what role the parallel distributed model has in influencing students' success, both at introductory robotics and at developing critical thinking skills.

Keywords: Robotics; Emergence; Complexity

[1] Contato: mdgross@cmu.edu

1. Introduction: Beyond Top Down

Some decades ago now, Nobel Prize winning economist Thomas Schelling wrote an influential article titled “On the Ecology of Micromotives” (1972). Using a simple cellular automata model to represent the distribution of “white” and “black” residents in a city, he showed how local preferences such as, “I don’t want to be the only person of my color in my immediate neighborhood” could produce segregated cities. Although such a benign preference might initially appear to be inconsequential, in fact Schelling showed that it results in radical segregation. In this model there is no designer causing segregation. It can even be said that no individuals intend to contribute to segregating the city but that is the ultimate outcome. Without explicit design for segregated cities, segregation can still arise from the combined local acts of individuals.

Here is an example of complexity—an unexpected global phenomenon emerges from the interaction of local behaviors. It is difficult to grasp, partly because we are accustomed to thinking about clear and direct relationships between cause and effect. If a city is segregated, surely someone must have designed it so. Not to deny the powerful political and economic forces that do in fact work to maintain segregated cities, Schelling shows that even simple individual decisions made locally can have profound global effects.

We have a bias towards simple, single-action and single-solution thinking. We teach engineers to think top-down: What is the problem? How can we solve it? This is the essence of engineering as we teach and learn it today - it makes a great deal of sense for many problems and so permeates our strategic thinking beyond engineering. As we tackle big problems in our daily lives we follow similar suit and use the “divide and conquer” strategy: What are the parts of the problem? How can we solve each part individually?

And yet, the problems of a new millennium challenge the top down command and control models that carried us to where we are today. On a number of fronts, we are faced with the emergence of global effects of local interactions. Top-down thinking fails to address these situations. We must learn to operate within this new paradigm in order to respond effectively to today’s challenges. Advancing our critical thinking and successful problem solving depends on seeing and thinking about the world in profoundly different ways than before. For example, AI pioneer Marvin Minsky’s Society of Mind (1988) argues that thinking is essentially a complex community of agents. In The Blind Watchmaker, Richard Dawkins (1987) shows how the filtered randomness that takes place in evolution accounts for what might otherwise be thought to be a process of intentional design.

In the top down paradigm a designer determines the behavior of a system. The machine works because we designed it that way. This single-cause and command and

control perspective influences how we view many kinds of systems - machines, politics, and even ourselves. We think of ourselves as a body controlled by a brain— our eyes and ears tell our brain about the world, and our brain tells our hands and feet what to do. To be sure, our bodies are far more complex, but this simple top-down view dominates our understanding.

The top-down systems design paradigm has helped us achieve amazing feats of engineering and technology: computers, reaching the moon, nanomaterials, modern medicine. It has also gotten us into trouble in not foreseeing the effect of changes and innovations. Designer pesticide DDT kills mosquitos and saves human lives as intended, yet, Rachel Carson showed, it also profoundly disturbs the food chain. No one set out to destroy the ozone layer or change the balance of carbon dioxide in the atmosphere. Still, due to unintended consequences, the planet is in the throes of global climate change. Although significant specific climate change culprits have been identified, a complex network of interactions and contributing factors makes it difficult to turn back these effects or even to fully predict what may result from proposed solutions, highlighting that many systems of local interactions are interwoven. If we learn anything in this post-industrial era, we must understand that the world is more connected and complex than we understood before. We've learned that local perturbations can have massive global effects (the proverbial "butterfly effect"). The top-down paradigm will no longer suffice; we must learn to see the world differently (Wolfram, 2002; Johnson, 2001). And we must educate a new generation of scientists and engineers to go beyond top-down thinking.

As we catalog the challenges of the 21st century, we are increasingly concerned with teaching science, technology, engineering, and mathematics (STEM). Our students must become problem-solvers and innovators, and we must prepare them to intervene in a world of complex problems. To address this challenge, many in-school and out-of-school programs teach robotics. Robotics offer a compelling and scalable instance of interdisciplinary STEM learning, and many of today's young students find robotics competitions engaging and motivating. The Lego FIRST, VEX robotics, and Robot Soccer World Cup competitions are three well-known examples. In these, student teams compete to design and build robots that accomplish a challenging task. Through designing and building robots students learn real-world STEM skills. They learn mechanics, electronics, and programming, and they learn to work together to engineer complex devices that meet a stated goal.

Because of the specific technologies that most robotics education employs, these approaches to teaching engineering assume and reinforce the top down paradigm. Simply

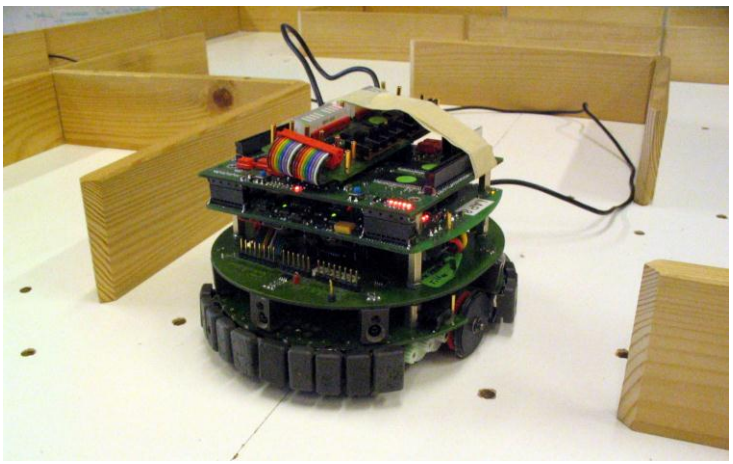
put, the robots students build comprise a single central 'brain' computer that controls the robot with a collection of sensors and actuators that mediate between the computer and the physical world. Engineering a successful robot is a matter of constructing the mechanics and electronics and then bringing it to life by writing command-and-control software that runs on the brain computer. Certainly designing and building robots with conventional single-brain central processor kits is an effective way for students to learn important engineering principles. Yet we believe that this approach to teaching STEM through robotics misses an opportunity to teach engineering design outside the top-down paradigm, and to begin to engage young students with designing with complexity in a hands-on way.

Our robot construction kit, Cubelets, breaks the top-down view of the world. It embodies instead, the local-to-global paradigm. Like other robot construction kits, it's in the constructionist tradition of Papert (1980). However, by giving young people experience in how large-scale behaviors emerge from local interactions, we aim to scaffold new ways of thinking about the world. To illustrate the difference, we turn now to a traditional elementary robotics task, to build a robot that finds its way through a maze from start to goal.

1.1 The maze-solving task: strategy and specific engineering challenges

A common challenge in beginning robotics education is the maze-solving task. Students must design and build a robot that autonomously navigates a physical maze. Figure 1 shows a typical example.

Figure 1 – EPFL's "Cyclope" robot in a maze.



Source: Wikipedia

As long as the walls of the maze form no loops or circuits, a simple wall-following strategy gets the robot through the maze. That strategy is to trace a wall on the left or right of the robot, steering continually to stay with the wall. (Choosing which matters only if the task is timed and if left-hand and right-hand walls do not comprise a "trip" through the maze

of equal lengths). Specific engineering challenges of building a robot that solves a maze in this way include:

- sensing — the distance to the left (or right) wall.
- acting — turning precisely while moving to maintain a fixed distance to the wall.
- controlling — executing the wall-following strategy.

Although solving a maze is an elementary robotics challenge it serves to illustrate the differences between following the usual top-down approach and the distributed computing approach that Cubelets encourages.

1.2 Thinking with your hands: Manipulatives Matter

We are certainly not the first to argue for the importance of understanding emergent behavior and complex systems through computational thinking. Perhaps best known is the work of Mitchel Resnick on StarLogo and Uri Wilensky on NetLogo (Resnick, 1994; Wilensky; Resnick, 1993; (see also Bonabeau et al., 1999). In these languages a programmer writes code to control the behavior of a single on-screen 'turtle'. This produces emergent behavior when each turtle in a swarm executes the same code in parallel. For example, a simple follow-your-neighbor program causes turtles to exhibit flocking behavior. Although these screen-based systems provide powerful platforms for experimenting with emergence of complex behaviors, we believe that a physical instantiation of these phenomena offers significant advantages.

The current fascination with screens—smart phones, tablets, laptop computers—reminds us that interactive computer simulations on displays can be convenient and instructive. As compelling as screen based simulations of physical phenomena may be for some, for many others experimenting with physical robots is more engaging. Hands-on learning eliminates a layer of abstraction. When a real-world phenomenon is represented on a screen (e.g., birds flocking), some explanation or schema must accompany it that relates the screen simulation to real-world phenomena. Discovery based, hands-on activity allows students to engage directly and with less explicit instruction. To be sure, some kinds of learning are more safely or thoroughly accomplished through simulations (e.g. flight training), but when students can accomplish learning objectives with physical materials, manipulate objects, and produce results, they have more control and freedom to explore. They also can debug and improve their creations without additional abstraction. Manipulating objects, making and creating, and hands-on learning is in the rich tradition of Dewey (Dewey, 1925), Froebel (see Brosterman, 1997) and Papert (1980) and as we've noted above, others in the constructionist community have addressed the significance of understanding parallel and distributed models of computation. The Cubelets modular robotics construction kit brings

these two models together into one tool that is both inviting to children and powerful enough to solve classical robotics tasks.

2. The top-down model

Typically, robotics students are equipped with a kit or robotics system to address various tasks, such as the maze-solving task we discuss here. Common kit choices include Vex Robotics and Lego Mindstorms; but students also build robots from components such as Arduino and Raspberry Pi microcontroller boards and self-designed laser cut pieces that serve as a robot chassis. Regardless of the choice of kit and equipment, students are expected to first build a physical and mechanical robot structure including sensors and motors, and then impart to it a top-down program to control and mediate the sensory input and the actuator output.

Students building a maze-solving robot have various options. Most simply, students can opt to use dead reckoning and write a program that amounts to a map complete with specified distances and times. This solution is unsatisfying on many levels as it will result in success only when one knows the exact dimensions and every turn of the maze the robot is to traverse in advance. Further, a robot that solves this maze cannot use this program to solve any other. Just as a Mars Rover depends on its ability to navigate by both predetermined mapping and unpredictable sense and response, a student-built robot that can solve a maze course autonomously fares better when it does not rely on a fully articulated and dead-reckoned map.

For these reasons, dead reckoning is seldom used in building maze-solving robots; indeed, in most competitions, a dead reckoning algorithm would not be considered meeting the maze-solving goal. Most simple maze-solving robots eschew dead-reckoning and opt to either construct either a Linear Time Invariant (LTI) system or a system that includes hysteresis. Both these robot types employ a central processor attached to peripheral sensors and actuators. They differ in that a hysteretic system records past states and data from trials as well as responding to immediate sensory information.

An LTI solution is predicated on the central processor that interacts with peripheral devices (sensors and actuators). The processor operates as a governor that manages rule-based cases in real time. If the peripherals are touch sensors and wheeled actuators, the robot can be programmed to respond to current information such as hitting a wall and correcting, or sensing an opening (“no wall”) and turning to enter it. A robot of this type is liberated from merely acting out a stored map, and because it interacts with its environment, it is more effective than the dead-reckoning solution. If its wheels move slower than a map-program had anticipated (due to weak batteries, slipping wheels, or other real-world

exigencies), the robot is not locked moronically into carrying out miscalculated instructions prepared in advance, but instead can respond, for example, following a left hand rule as it comes to an opening on the left, whenever the speed and traction of its wheels causes it to arrive there.

In this solution to the maze-solving challenge, a program can manage more than one rule of sense-and-react allowing an LTI robot to respond to various conditions in a maze including light, color, and proximity of walls (using infrared or ultrasonic sensors). However, robots of this design do not learn. In each attempt through any maze, the robot simply transmits information from its sensors through the central program to apply a rule to it as it encounters light, color, or other sensory input much like the proverbial goldfish encountering the landscape new each time it rounds the edge of its bowl.

In order for a robot to not only react, but to gather information and then deploy it in later episodes, the central program must not merely function as an arbiter of sense-act rules, but also gain and store information through experience and act on it in the future. Here again we would expect to see a central program acting as the robot's "brain" attached to peripheral sensors and actuators. The physical structure could be quite similar to the LTI robot described above: typically a chassis holding the central processor with attached sensors and wheels. Upon first encountering the maze, the robot's program has no memories and so it behaves as an LTI robot, responding to lights, colors, and proximity according to sense-and-react rules in its program. The difference is later, on subsequent runs of the maze in an expanded program that not only selects and applies rules responding to stimuli, but that also remembers what rules were used in previous trials, and with what result.

3. The distributed intelligence model

3.1 Cubelets— brief explanation and technical summary

Cubelets are a modular reconfigurable robot construction kit, inspired by Valentino Braitenberg's (1984) book, *Vehicles: Experiments in Synthetic Psychology*. The kit consists of small (40mm) plastic blocks that snap together with magnets to make interactive robots. Each type of Cubelet has a different function: Some are Sense Cubelets, some are Action Cubelets, and some are Think Cubelets. When you snap together a Sense and Action Cubelet with a Battery Cubelet, you've built a simple robot that senses a signal from its environment and acts in response. For example, a robot that uses the Distance Cubelet as its sensor and Drive Cubelet as its actuator takes as stimulus an object (or a hand) near its sensor, and translates the magnitude of this proximity input into a corresponding wheel speed.

Unlike most other robot construction kits, a Cubelets robot has no single central processing computer or 'brain' module that controls its behavior. Rather, each individual Cubelet acts locally on information it receives from its neighbors. For example, a Distance Cubelet uses an infrared proximity sensor to sense the distance to any object in range and announces that distance in the form of a number (0-255) to each of its neighbors. A Drive Cubelet receives numbers (in the range 0-255) from its neighboring Cubelets and runs its motor at a speed based on these numbers. With individual functions built into each block, the Cubelets in a robot constantly pass numbers from one block to the next—these numbers are generated by Sense Cubelets and are consumed by Action Cubelets.

For example, the simplest Cubelet robot that you can build has only one Sense Cubelet, one Action Cubelet, and a Battery Cubelet. Take the three-Cubelet "Follow Me" robot shown in figure 2. The Distance Cubelet senses when you place your hand in front of it, and produces a number, which it passes to the Distance Cubelet behind it. The Distance Cubelet, oriented to drive "forward" (in the direction of the Sense Cubelet), turns the number into a motor speed, and drives the three-block construction toward your hand.

Note that even in this simplest of robot constructions there is no explicit program controlling the robot's behavior. Rather, its behavior results—emerges, if you will, from the combination and configuration of its Cubelet parts. To the extent to which we could say the robot is executing a program, that program is implicit in the construction as a whole. There is no one place in the robot that the program is stored. There is no central processor, no 'brain' to the robot. Behaviors that we deem "intelligent" are solely the result of local interactions— there is no representation of the problem or the context, as Brooks (1991) described over twenty years ago in his influential article, *Intelligence without Representation*.

Figure 2.A three-Cublelet 'follow-me' robot made of a Distance and Drive Cubelet.

Source: Author

3.2 A Cubelets maze-solving robot

Building a Cubelets robot that navigates a maze is surprisingly simple. Figure 3 shows the five-Cubelet robot. It has two Distance (Sense) Cubelets and two Drive (Action) Cubelets (and, of course a Battery Cubelet to power it). One Distance Cubelet points left; the other points right; and the two Drive Cubelets both drive forward.

The robot steers by differential drive: When the left Drive Cubelet goes faster than the right Drive Cubelet, the robot steers right, and conversely, when the right Drive Cubelet goes faster than the left one, the robot steers left. The Distance Cubelets, which point outward (left and right), control the speed of their respective Drive blocks. The left Distance Cubelet controls the speed of the left Drive Cubelet and the right Distance Cubelet controls the speed of the right Drive Cubelet. (Actually, the number from both Distance Cubelets reaches each Drive Cubelet. However, the number from the closer Distance Cubelet is weighted heavier than the number from the farther Cubelet. The farther from the Sense Cubelet source, the lower the weight on its number value. In short, the effect on each Drive Cubelet of the nearer Distance Cubelet is stronger.)

Figure 3. A Maze Solver: two Distance Cubelets, two Drive Cubelets and a Battery.

Source: Author

This simple five-block robot follows walls. Suppose we place the robot next to a wall on its right (see figure 4). The right Distance Cubelet senses the wall, and causes the nearest Drive Cubelet (the right Drive Cubelet) to go slightly faster than the left Drive Cubelet, because the left Distance Cubelet doesn't sense a nearby wall. So the robot turns slightly to the left, away from the wall. At some point, the right Distance Cubelet no longer

sees a wall nearby, and so it sends a smaller signal to its adjacent Drive Cubelet. Now both Drive Cubelets move at the same speed and the robot drives straight along the wall.

Next suppose the robot, traveling along a wall to its right, reaches a convex corner (figure 4-a). The wall on the right falls away, so the right Distance Cubelet produces a much smaller number (figure 4-b), which slows the right Drive Cubelet. The left Distance Cubelet continues to produce the same number, so the robot turns to the right (figure 4-c), until the right Distance Cubelet again sees a wall (figure 4-d).

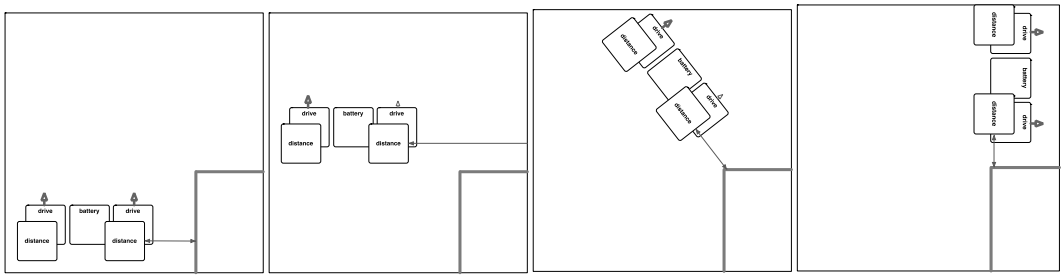


Figure 4.(a) The robot approaches a convex corner; (b) The right Distance Cubelet senses an opening on its right as its sensor value drops; (c) This causes the right Drive Cubelet below it to slow down, which makes the robot start turning; and as it completes the turn the right Distance Cubelet once again senses a nearby wall and the robot proceeds straight again.

Likewise, when the robot traveling along the wall encounters a concave corner, the right Distance Cubelet sees a closer wall. It produces a larger number, increasing the right Drive Cubelet's speed, which steers the robot to the left.

4. Discussion

Comparing the Cubelets maze-solving robot with the central command-and-control solutions that more conventional robotics kits favor we immediately see some differences. The Cubelets robot is far simpler in design and construction. Depending on one's point of view, this is either a strength or a disadvantage. The functional behaviors of sensing and acting are built in to the Distance and Drive Cubelets, and therefore the kit does not afford students the opportunity to design the details of sensing and actuation. Also students need not learn to manage a programming environment, so there is no separate coding step.

In a Lego Mindstorms or Vex Robotics robot, the student can adjust the positioning of sensors, the location of the drive wheels, and the construction of the chassis, and they can author and edit code that controls the robot's performance. These engineering decisions enable the designer to optimize the performance of the robot, and thereby they provide learning opportunities. In a Cubelets robot on the other hand, the kit's designers have already made these choices. This focuses the student's attention on higher-level design principles and abstracts away many implementation details. The Cubelets robot-builder can

experiment with alternative designs much more quickly, because reconfiguring the Cubelets building blocks is as fast as repositioning them (no wires, no plugs) and requires no coding to change its behavior. Among other consequences, this design invites younger and less technically sophisticated students to experiment with robotics.

If we examine how the Cubelets maze-solver succeeds in navigating the maze walls, we can see that a Cubelets robot is, in fact, an linear time-invariant (LTI) system. It responds in real time to stimuli it senses, but it has no memory of previous episodes. By definition and design, Cubelets have no central "brain", so there is no simple way to program a Cubelets robot to remember its experience of running a maze. Thus every Cubelets robot responds only to real-time sense data in its present environment. (It is at least theoretically possible to add memory and learning functions, by reprogramming the firmware in the Cubelets, a function that the kit supports. However, that is another story.)

We can see similarities between how a command-and-control robotics system using an LTI design and a Cubelets robot solves a maze. In Lego Mindstorms, Vex Robotics and other commonly used conventional robotics kits—as well as in Cubelets — a robot with no memory requires sensors and actuators. The sensors convey information about where the walls are. Any sensor that detects edges, colors or touch will have limitations and without a program that provides learning from earlier maze attempts, these robots can only process signals as they travel through the maze.

And we can see one significant architectural difference: A Cubelets robot is a parallel system. Instead of sensors and actuators that serve as peripheral devices conveying information to or from a central control module, Sense and Action Cubelets talk directly to one another. And there are basic physical differences: a Cubelets robot has no need for a chassis to carry the central control module. More profoundly, there is no provision for memory. Both these differences have resounding design effects. On the physical level, students need not construct, only assemble. Students can (indeed, must) eschew the concerns of stability, weight, and the relationship between these characteristics and the speed with which a robot can traverse a maze. These elements are predetermined by the Cubelets.

In the same vein, absent the ability to afford memory to the robot "brain", students can only design a successful maze-solving LTI or an unsuccessful maze-solving LTI robot. Using Cubelets, students quickly realize that if a solution exists, it can best be found by iterative design—quickly building and rebuilding a robot and testing it in maze conditions, each time adjusting or refining its operation. This rapid design-and-test process we've observed with Cubelets differs powerfully from accounts of classic, top-down maze-solving

robotics. There, students tackle, as separate endeavors, hours designing, building, and refining a mechanical structure and then as many or more hours reprogramming and perfecting code to “run” their robot. Viewed this way, the Cubelets model supports more of the engineering design lessons student robotics is intended to address in facilitating testing and revising designs more efficiently.

A central brain unit allows optimization in how a robot solves a maze, but it also segregates hardware design from software engineering and, later, necessitates integrating these two design components, a process that demands its own design process and decisions. Each component hews to a different set of constraints, and integrating hardware and software designs is not always straightforward. In Cubelets, reprogramming the robot is tantamount to rebuilding the robot itself—switching Sense Cubelets, adding Cubelets, or reconfiguring those already in use. This eliminates the disconnection between hardware and software (structure and program) because form and function are one and the same. It also supports students to quickly construct and test solutions in real time, using the actual hardware.

Still, if the goal is to build the best maze-solver, then the traditional, top-down approach has obvious and significant advantages. By incorporating a central brain module, designers can choose among options to exploit in how the robot will not only solve a single maze, but also be optimized to solve the same maze faster or to solve future mazes. In psychology, human problem solving and processing often involves a speed-accuracy trade-off. Classical robotics offers choices and trade-offs as well: Robots can be designed as dead-reckoned, Linear Time Invariant, or able to remember and learn. Much like our own thinking, robots can be single-tasked here-and-now, or combine real-time data with past experience. Sophisticated designers also can mix these different kinds of thinking, for example a program that controls the robot using a map for some portion of the maze, but solves the rest by a set of rules that mediate between peripheral sensors and actuators in real time. Students can also choose a half-step between pure LTI robots and programs that incorporate past-states influencing current performance (memory/learning) by introducing a time-delay in how the program transfers to real-time sensor data to actuator response, allowing robots to “ponder” data rather than immediately react. Having the option to introduce memory/learning from past trials to a robot brain is emblematic of the preferential importance we place on our own memory.

In fact, all the top-down solutions mentioned are ones that a human might employ, and in many circumstances we would use more than one. In a suddenly darkened room, a person might use memory of the room’s dimensions and obstacles to exit, or proceed to a

wall and then use a left-hand or right-hand rule to find egress, or mix these ways of recalling and using sensory data. The real advantage of the top-down architecture is the ability to choose which of these several strategies (or combinations thereof) the program will afford the robot.

A distributed intelligent model as in Cubelets doesn't let students easily create robots with different kinds of problem solving strategies, but it does offer students a direct and immediate way to design and test a robot. This most directly benefits those who want to teach or learn beginning robotics without first having to devote discrete project time to learning to code. By simplifying maze-solving to be real-time inputs and outputs that are parallel and transferring data directly to one another this classic and elementary robotics task can engage novices. This, in turn, is supported by students of all levels being able to solve these tasks directly with their hands. With no pause to re-program and no disconnect between form and function, students can quickly design, test, and redesign. With small, local changes to the configuration or orientation of just a few components students may produce a different emergent behavior. This robotics platform also affords the ability to instantly test proposed solutions and adjustments. This not only facilitates novices and non-programmers to approach maze-solving and robotics, but also elegantly limits the conditions of maze solving to real-time inputs and outputs. More advanced robotics enthusiasts and novices alike are bound to consider more carefully the maze conditions and what the robot needs to "see" or "know" a priori to being in the maze. A posteriori although the robot hasn't learned anything, those who designed it have learned and can use their empirical observations to immediately adjust inputs and outputs.

Introducing this system to students has educational benefits that surpass the mere ability to introduce robotics problem solving to less technically fluent students. Students of all levels of expertise encounter a powerful example of how discrete individual functions combine and form a complex and sophisticated system—in this case one that can navigate a maze. Students who come to the task because of an interest in robotics and technology or an attraction to the designing and making that Cubelets affords, leave with a powerful and hands-on demonstration of how local interactions integrate to produce emergent and sometimes intelligent-seeming behavior. A parallel and distributed robotics kit such as Cubelets requires students to tackle problems by considering parallel and distributed solutions, but invites them to do so in an engaging hands-on task. Expanding on the powerful motivation that manipulatives supply, Cubelets offers a strong platform for design-and-test engineering education, but also puts problem-solving tools, directly in the hands of students, that embody local-to-global emergence.

5. Conclusions

A parallel and distributed robotics system like Cubelets affords simple maze solving and a more concrete understanding of design-and-test engineering. Cubelets also places the tools of causing small, local behaviors that combine to form a complex system. Students gain experience and observe how local interactions can cause unexpected emergent behaviors.

In our experience with small groups of students we found that students quickly grasped the distributed computation paradigm of Cubelets and once they grasped the paradigm, they could use it to design and build a functional left-hand or right-hand navigator for simple mazes. Moreover, the students we worked with were able to construct solutions with little guidance. Notably, a group of under-resourced and at-risk students with no robotics background were able to consider, test, and refine maze-solving solutions with little more than a Cubelets demonstration. Given minimal guidance (e.g., using prompts such as asking these students how they would find their way out of a darkened room) they successfully constructed maze-navigating Cubelets robots.

Three questions remain for future work. First, we wonder how robustly can Cubelets address classical robotics tasks? To be sure, the physical form of Cubelets — 40 mm cubes that connect magnetically with a single function on each block — limits the designs that robot builders can realize. Second, to what extent can the distributed computation model that Cubelets employs address the kinds of robotics challenges that students encounter; and conversely, what kinds of robotics challenges are suited to the hardware and software architecture of Cubelets? Finally, there is the question of “transfer”. To what extent, if at all, do students adopt the meme of local-interactions producing global effects and transfer this way of thinking to other learning and problem solving?

6. References

BONABEAU, E.; THERAULAZ, G.; DORIGO, M*. **Swarm Intelligence***: From Natural to Artificial Systems (Santa Fe Institute Studies in the Sciences of Complexity), Oxford: Oxford University Press, 1999.

BRAITENBERG, V. ***Vehicles***: Experiments in Synthetic Psychology. Cambridge, MA: MIT Press, 1984.

BROOKS, R. A. Intelligence without representation. In ***Artificial Intelligence*** 47, p.139–159, 1991.

BROSTERMAN, N. ***Inventing Kindergarten***. New York: Abrams, 1997.

DEWEY, J. (1925). ***Experience and Nature***. New York: Dover Publications, 1958.

DAWKINS, R. ***The Blind Watchmaker***: Why the Evidence of Evolution Reveals a Universe without Design. New York: Norton, 1987.

JOHNSON, S. ***Emergence****:* The Connected Lives of Ants, Brains, Cities, and Software. New York: Scribner, 2001.

MINSKY, M. ***Society of Mind***. New York: Simon and Schuster, 1988.

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

RESNICK, M. ***Turtles, termites, and traffic jams***: explorations in massively parallel microworlds. Cambridge, MA: MIT Press, 1994.

SCHELLING, T.On the Ecology of Micromotives. ***The Public Interest*** 25: p.61–98,1972.

WILENSKY, U.; RESNICK, M. Beyond the Deterministic, Centralized Mindsets: A New Thinking for New Science. Paper presented at the annual meeting of the ***American Educational Research Association***, 1993.

WOLFRAM, S***. A New Kind of Science***. Champaign, IL: Wolfram Media, 2002.