Bitwise Biology: Crossdisciplinary Physical Computing Atop the Arduino

John Grasel, Wynn Vonnegut, and Zachary Dodds

Harvey Mudd College Computer Science Department
301 Platt Boulevard
Claremont, CA 91711
jgrasel, wvonnegut, dodds@hmc.edu

Abstract

We present the design and deployment of a physical computing platform developed for a crossdisciplinary introduction to biology and computer science. Using the accessible Arduino interface as its foundation, students instantiate increasingly nuanced physical interactions with the environment. Biological and computational ideas receive equal attention through three layered projects that span from circuit design through the co-evolution of predator-prey robot behaviors. The low-overhead platform presented here scales to support sophisticated projects at surprisingly modest time-and-money costs.

Motivation

Life computes – perhaps no other two-word sound bite better captures the spirit and challenge of modern biology. Life's computation spans orders of magnitude that dwarf those of our artificial machines. Its sophistication and intrinsic value offer a promise to which computer science can, at its best, contribute both insight and intellectual resources.

Tomorrow's Biology and CS curricula will be even more intimately interconnected than today's. Poised to support such efforts are embodied computational artifacts: after all, *unscripted* environmental interaction is what distinguishes life's cogitation from that of our computers. This paper proposes and reports on how embodied computation – in the form of a scalable, Arduino-based electronic tangible – has supported hands-on laboratory exploration of the union of the fields of biology and computer science.

Here we contribute the curricular context and the design of that physical platform, depicted in Figures 3 and 4, used by students in three projects:

- an exploration of circuit design and phototaxis,
- the physical modeling of bacterial motion, and
- an investigation of the phenotypic reinforcement of genotype co-evolution in predator/prey relationships.

The classroom deployment of these materials is currently underway; we report on our evaluation methodology and interim results. Although these three projects are carefully circumscribed, the possibilities for Figure 1's platform are broad. We conclude with a description of some of the other applications that such a scalable and accessible system could support.

Background

The context for these three Arduino-based labs is an integrated introductory curriculum deployed at Harvey Mudd College (HMC) in the fall of 2009. The course fulfills both the CS 1 and Biology 1 requirements that all HMC students must complete by the end of their sophomore year. Twenty-eight students were selected from over fifty applicants for this pilot offering.

An introduction to biology *and* computer science

There is already a decade-long history introductory CS courses using computational biology as motivating context (LeBlanc and Dyer, 2004). Such efforts continue to expand and mature (Soh et al., 2009). In addition, there are wide-ranging examples of introductory biology curricula that leverage computational tools and increase students' savvy with them (Burhans and Skuse, 2004; Khuri, 2008; McGuffee, 2007).

To our knowledge, however, there do not exist other introductory courses that interweave biology and computer science topics at a depth and breadth sufficient *to fully support majors of either discipline* -- either separately or in combination. Figure 1 presents an overview of such a course's material and its integrated presentation.

A biologist presents the first lecture of each week; the second lecture motivates and connects a fundamental

computational idea from that previous meeting's biological basis. For homework by the end of the week students apply that computational idea in order solve, model, or explore two to five biological challenges. Python is the computational currency of most of these student homework assignments. To support this integrated approach we use both a popular biology text (Sadava et al.) and an in-house CS monograph, *CS for Scientists and Engineers*.

Example assignments

Figure 1 also summarizes three concrete examples of the interweaving of computational and biological material from the course. The full year of course assignments, lecture slides, and supporting materials are freely available from the course URL noted below.

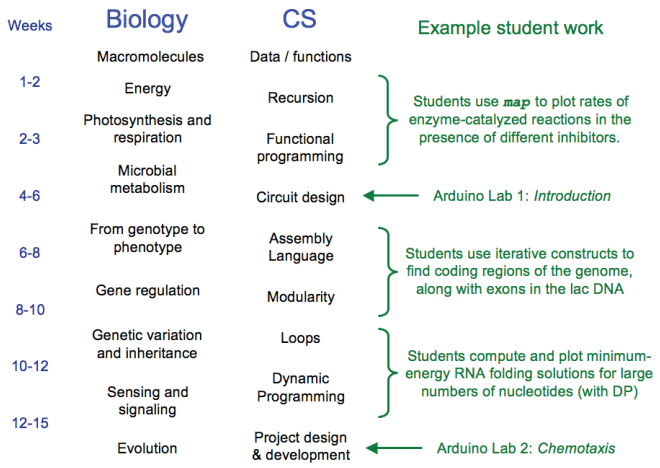

Figure 1. Summary of fall-semester Bio/CS topics. The third Arduino-based lab comes during the course's spring term.

Lab sessions

The course offers students a regularly scheduled, guided laboratory session. Most weeks, this offers a low-pressure setting in which to gain confidence with the Python required to investigate the week's homework. Because of the overhead associated with wet labs, we plan to offer only one: a gel-electrophoresis task in the spring semester in which students write Python-based image processing programs in order to interpret the results. Yet even in our dry labs we have sought hands-on *computational* metaphors through which each lab can reinforce the curriculum's primary theme: *life computes*.

We saw three opportunities for which hands-on, physically embodied computation seemed particularly appropriate:

- To reinforce the ideas of modularity and composition in both biological and artificial computation, we wanted students to build simple physical circuits from logic gates -- and then put them to use in a light-seeking task, contrasting with biological phototaxis.

- When presenting the chemical basis for single-celled organisms' volition, we wanted students to physically model the directed random walks produced by the choices in the direction of flagellar rotation.

- We also wanted students to investigate the interrelationship between predator/prey genotypes, as played out by their phenotypes. The genotypes (software) co-evolve into a pair of increasingly fit phenotypes (hardware) through a pair of mutually recursive feedback loops. This project dovetails with the course's coverage of evolutionary ecology.

Certainly we could have found a separate hardware solution to support each of these three projects. Yet bringing in new support materials for each investigation suffers from several drawbacks. First, each distinct hardware platform comes with its own, often significant, learning curve. Second, multiple platforms increase expense and, more worryingly, demand additional time for curricular and software support and maintenance.

Perhaps most importantly, a separate-tool-for-each-project approach unnaturally hides a fundamental insight common to real biological and real computational systems: the remarkable number of layers of abstraction that make such complex systems possible. Although one might argue their relative sophistication and efficiency, the hierarchy that creates ecosystems out of elements and the hierarchy that builds Google from gates share all the conceptual challenges of modularity, interdependence, and staggering depth.

The platform

Thus, we opted to explore single electronic tangibles that might scale across each of the hands-on projects we hoped to support. Seeking simplicity, ease of programmability, and low cost, we settled on the materials shown in Figures 3 and 5, below.

Lab 1: the building blocks

The foundation of the platform is the popular and powerful Arduino interface, whose open-source hardware and software runs across operating systems. The first lab starts with basic series circuitry (integrated circuits, LEDs, resistors, and wire) and then adds with sensing and actuation: a photoresistor and continuous-rotation servomotor.

In our very typical computer-room lab space, each of the 14 pairs of students used the bill of materials detailed in Figure 2, for a total cost of about $60 per pair. The size of the components allows all of the building to fit between the workstations at which each of a pair of students sits.

After creating a one-bit adder from gates to reinforce the universality of the {AND, XOR, 0, 1} set of logical primitives, students in Lab 1 use Python to manipulate more extensive circuits programmatically. They hook up a sequence of LEDs and design their own "hypnotizer," a light-display pattern they show off to colleagues and instructors. After their mesmerizing display works, students add light *sensing* via a photoresistor and actuation through a continuous-rotation servomotor. Students excited about logic design have the option of building a 2-bit adder, although Figure 4 testifies to its complexity.

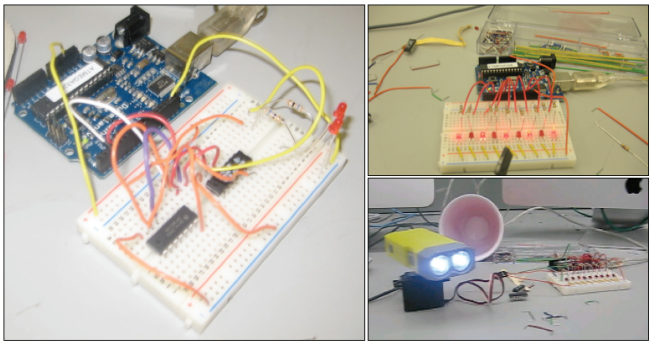

Figure 2. Completed light-lab subprojects highlighting the components used, including circuit composition in the one-bit adder at left, LED outputs in the "hypnotizer," top right, sensing via a photoresistor, and actuation through a continuous-rotation servomotor (lower right). The support library and student-written code are in Python, but could be in any language at all.

Materials List
for 2 students together

Arduino	$30
USB cable	$ 5
Protoboard	$ 5
Servo motor	$12
Logic Gates	$ 1
Photoresistor	$ 1
Wire and LEDs	$ 3
total	$57

Additional Materials
for the chemotaxis-modeling robot

+ servo motor	$12
+ photoresistor	$ 1
and when untethered	
+ batteries	$ 5
+ battery holder	$ 2
total	$20

Figure 3. A list of parts and prices of Figure 2's materials. Beside those are the additional components necessary to support building and programming the biased-random-walk project modeling bacterial chemotaxis, tethered or untethered.

The final product of the lab is a Python program that combines the motor, sensor, and LEDs into a light meter and a model of photophilic behavior. They attach a penlight to the servomotor and set it in motion. Its output impinges on the photoresistor. The LEDs then indicate brightness on a scale from zero-to-eight. Figure 2 shows some of the student teams' final systems.

Students control the motion, process the sensor data, and light the LEDs through Python programs they build from scratch during the lab. We found that two hours sufficed for about half of the ten pairs to complete the lab. The other half successfully created a program that integrated LED display and light-sensing, but without the servomotor. Future runs will streamline the introduction to ensure that actuation fits into the two-hour timeframe. We were heartened, however, that students wanted to (and did) borrow the materials to complete the lab in their rooms.

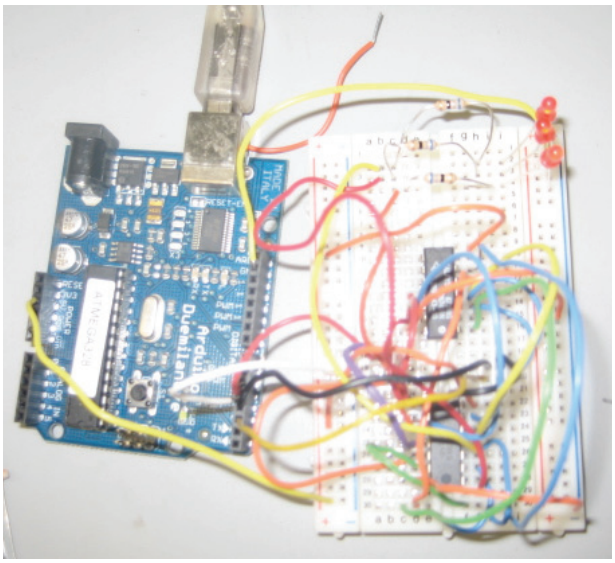

Figure 4. Although a one-bit adder is more suitable for a 2-hour lab, the construction of a two-bit adder (above) provides hard-won appreciation of the complexity that both biological organisms and today's computational devices embody!

Pedagogically this lab offers the students practice in the careful composition of disparate software components. Although they had practiced functional decomposition and programming for five weeks, this lab provided the first connection between their programs and human-scale physical interactions. Because this activity overlaps with the course's coverage of photosynthesis, we used the opportunity to contrast the sensing and energy use in both biological and artificial physical systems.

Lab 2: a robot "cell"

The introductory lab comes a bit before midterm; the follow-up is one of the end-of-term final projects. Building on the light-sensing and rotational actuation of the Light Lab, Lab 2 challenges students to create and program a physical model of bacterial chemotaxis in the form of a small wheeled robot.

An advantage to the first lab's materials is that building a mobile robot requires very little else. As the example robot in Figure 5 attests, the breadboard can act as chassis, scavenged plastic cup parts as wheels, and a hairclip can serve as a stabilizing caster.

The backstory motivating this lab is the chemical signaling that biases bacterial random walks. E. coli, for instance, are too short to measure the spatial gradients of attractant and repellant molecules directly. Rather, they estimate the spatial gradient by time-averaging those molecules' density.

These bacteria have only two actuation options: to move forward at their current heading or to rotate in place to a random orientation. Thus, they respond to favorable gradients by predominantly choosing to move forward. Unfavorable gradients cause a spike in random reorientation, followed by shorter forward steps in an attempt to escape.

The students implement precisely this behavior on their robots. They use light gradients in lieu of chemical gradients and can contrast direct spatial estimation, as done by Braitenberg vehicles, with the time-integrated signal that bacteria compute. The actuation is a natural extension of Lab 1's turret to two motors supporting differential drive. Figure 5 shows one such result.

Lab 3: Modeling predator-prey co-evolution

The second semester of this introduction to biology and CS concludes with an even larger project. We chose to build a capstone experience that investigates the co-evolution of predator-prey genotypes. The interdependence of species' genotype evolution has long been a cornerstone of evolutionary ecology. This project builds from the robotic modeling of the predator-prey co-evolution in (Floreano and Nolfi, 1997) in a way that reinforces both its biological and computational underpinnings.

The robots from the bacterial-modeling lab (Lab 2) can be used unchanged. Following (Floreano et al, 2001), we have them act as predator and prey by simply playing tag -- as long as they know each other's relative position, which can be provided by an overhead camera or in a software simulation. Predators move faster but with a limited field of view; the prey have 360-degree sensing, but move less quickly.

The genotype of each species is modeled by a short bit sequence. That sequence encodes coefficients relating motor strength with the relative pose of the opposing agent. When the predator succeeds in contacting, or "tagging," the prey, its genotype's fitness is increased and the prey's is decreased. When a prey phenotype evades a predator, that reinforcement reverses. A genetic algorithm then uses those fitness values to create a new pool of genotypes, whose corresponding phenotypes continue the cycle.

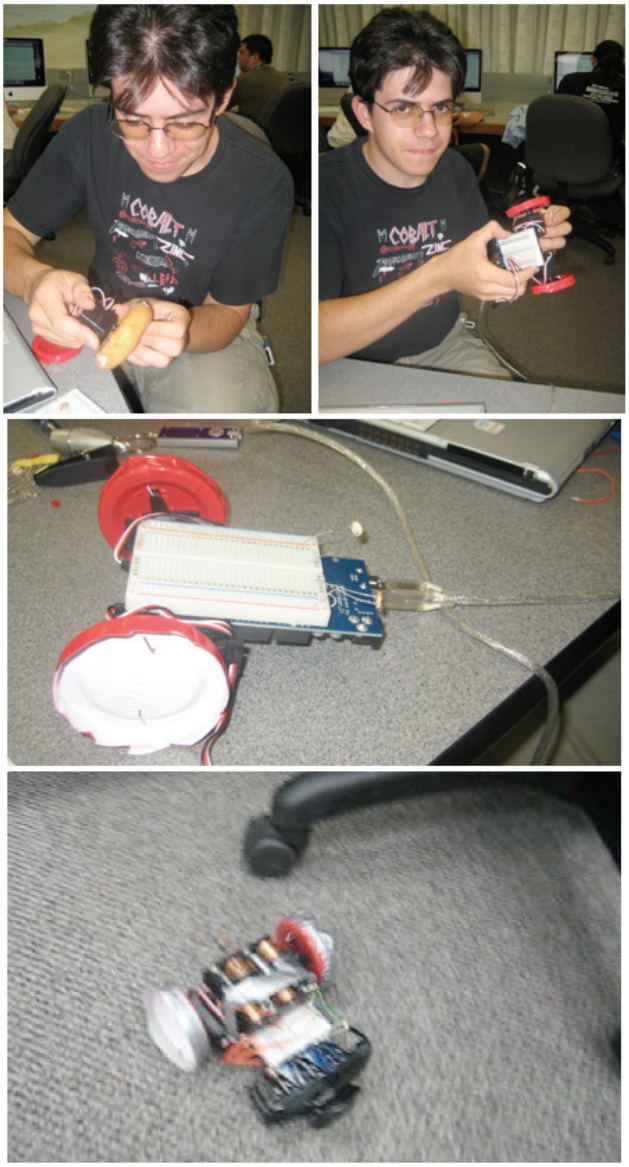

Figure 5. A student-built robot that completed the biased-random-walk project. At top, wheel designs are considered: plastic cups superceded doughnuts, which were eaten instead. The robot in the middle image runs tethered to the laptop computer, with the USB bus providing all power and control needed. As the bottom frame indicates, the transition to untethered operation is not difficult. This robot and its light-avoiding control program were both designed and built in the time available in our two-hour lab session by a single student who had previously worked through the light lab's introduction.

This summary describes an ongoing course. As such, this predator-prey project has yet to be finalized; it will be deployed in April, 2010. Even so, it is the software support that will make or break this curricular experiment, and that

software has already been thoroughly vetted and tested.

Leveraging few external libraries Inspired by Myro's *Pythons all the way down* philosophy (Blank, 2006), we have built a lightweight library for interaction with the Arduino in general and specifically for visualizing the state of the students' Arduinobots. Both require nothing beyond a standard Python 2.5.x or 2.6.x install and the **`pyserial`** library; they work under Windows, Mac OS X, and Linux.

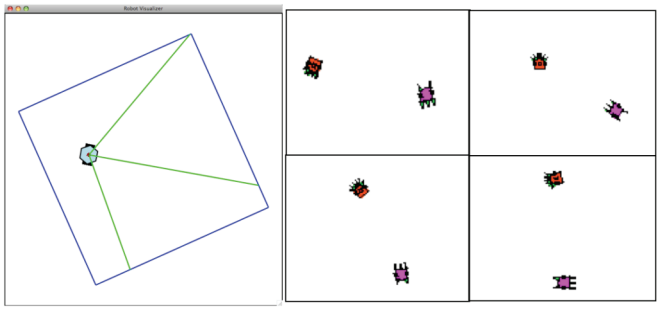

Figure 6. Snapshots from the robots' Python visualizer. The two robots featured in the right side's frames are predator and prey. They interact under the direction of their genetically-determined mappings from sensors to motors. Those software genomes are preserved in proportion to their fitness and mutated randomly. Surprisingly capable predator- and prey-behaviors result.

The graphical interface is an adaptation of John Zelle's wonderful **`graphics.py`** module with added support for arbitrary affine transformations and simplified separate-window GUI building. Because it uses the default Tkinter interface, no packages or libraries beyond Python itself are needed. It also runs as happily from within the IDLE development environment as from the command line. This was a deliberate consideration in its design, in order to make the learning curve as gentle as possible for students new to programming.

Our curriculum and software is freely available from the same source from which student lab participants download it: **https://www.cs.hmc.edu/twiki/bin/view/CS6/Arduino**.

Results

As of this writing, we have created and student-prototyped the three projects that build upon this Arduino-based platform. In addition, we have run the full class's 28 students through the circuit-building and light-seeking lab (Lab 1). Also, selected students have twice completed the bacterial-modeling lab (Lab 2) as preparation for full-class deployment in December. The predator-prey lab (Lab 3) has undergone thorough software testing, but its hardware implementation remains to be completed.

Student responses from that lab experience have been positive, both *per se* and relative to the lab and course experiences that do not employ electronic tangibles. Figure 7 shows the distribution of the *worthwhileness* and *difficulty* reported by students, two questions we have asked for years about introductory CS assignments.

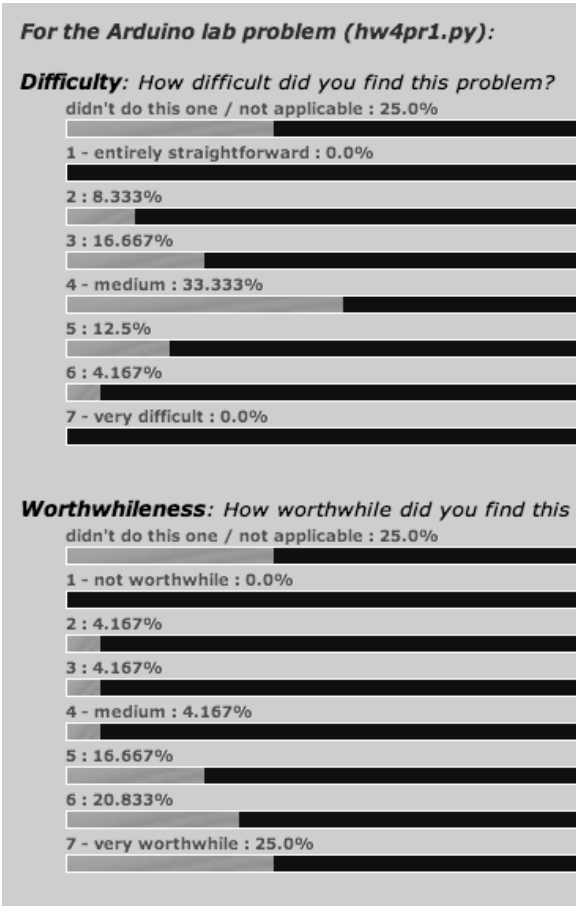

Figure 7. Student feedback from the "light" lab suggests a suitable difficulty for a valuable learning experience.

We were heartened to see that this lab earned the highest percentage of maximum *worthwhileness* scores to date.

Although head-to-head comparisons with other activities are still being analyzed, we point out that our goal is not that these physical computing labs replace or improve upon any existing activity. Rather, our hope was to broaden both the set of skills that students exercise and the comfort level they achieve throughout the Bio/CS experience. End-of-term feedback will provide data for measuring the extent to which these materials helped in those efforts.

Anecdotally, however, the hardware lab was enjoyable because it offered so different an interface to computation from previous weeks. Formally submitted responses to the lab include "I really liked the lab this week." and "2-bit

adder. YEAH!" Informal feedback reinforces these positive sentiments. Even so, a particularly thoughtful response pointed out that there is still considerable room for improvement:

> Lab was awesome, just seemed like a very broad topic being crammed into a tad too little time. Perhaps splitting it into two different days, with the first having more emphasis on getting the 'xor' and 'and' gates to work, and the second involving the python programming of the arduino would be better suited?

Certainly we look forward to refining all of these labs for future offerings. In fact, we hope to expand at least the introductory Arduino lab to *all* 200+ students who take CS1 each fall, not only those who opt for Bio/CS.

Perspective

Computational interaction among physical systems is fundamental to both biology and computer science. Inexpensive interfaces, such as the Arduino, are a robust and accessible foundation for building curricular links between those two fields in an engaging and hands-on manner. What is most exciting, in our opinion, is the *generality* of the toolset presented here.

This paper's labs only hint at the possibilities. These materials offer the most accessible bridge we have encountered between computation and the physical world. What's more, they scale well pedagogically. By adding other off-the-shelf circuit components, we use the same software and hardware to control indoor and outdoor fleets of robots used by students in elective classes, robot competitions, and research projects. Other departments at our institution are joining us on the bandwagon: an Arduinobot-based engineering elective focusing on feul-cell creation and power-autonomy will run in Fall 2010.

Electronic tangibles, in short, open a vast space of opportunities for deep, cross-disciplinary student engagement. We look forward to further exploring that space -- both within our own curriculum and in concert with the broader educational community.

Acknowledgments

The authors thank the anonymous reviewers for their insights and suggestions; we also acknowledge the generous support of funding from HHMI award #52006301, NSF DUE CCLI #0536173, and Harvey Mudd College.

References

Blank, D. 2006. Robots make computer science personal, *Communications of the ACM*, 49(12), pp. 25-27, ACM Press.

Burhans, D and Skuse, G. 2004. The Role of computer science in undergraduate bioinformatics education, *SIGCSE '04*, pp. 417-421, Norfolk, VA. ACM Press.

Floreano, D. and Nolfi, S. 1997. God Save the Red Queen! Competition in Co-Evolutionary Robotics, *Proc. Genetic Programming '97*, pp. 398-406, Morgan Kaufmann.

Floreano, D., Nolfi, S., and Mondada, F. 2001. Co-Evolution and Ontogenetic Change in Competing Robots, in M. Patel, V. Honavar, and K. Balakrishnan (eds.) *Advances in the Evolutionary Synthesis of Intelligent Agents*, MIT Press.

Khuri, Sami. 2008. A bioinformatics track in computer science, *SIGCSE '08*, pp. 508-512, Portland, OR. ACM Press.

LeBlanc, M. and Dyer, B. 2004. Bioinformatics and computing curricula 2001: why computer science is well positioned in a post-genomic world, *ITiCSE-WGR '04: Working group reports from ITiCSE on Innovation and technology in computer science education*, pp. 64-68, Leeds, U.K., ACM Press.

McGuffee, J. 2007. Programming languages and the biological sciences, *Journal of Computing in Small Colleges*, 22(4), pp.178-183, CCSC Press.

Sadava, D. Heller, H., Orians, G., Purves, W., and Hillis, D. Life: The Science of Biology, W. H. Freeman and Co. NY, NY.

Soh, Leen-Kiat, Samal, Ashok, Scott, Stephen, Ramsay, Stephen, Moriyama, Etsuko, Meyer, George, Moore, Brian, Thomas, William G. and Shell, Duane F. 2009. Renaissance computing: an initiative for promoting student participation in computing, *SIGCSE '09*, pp. 59-63, Chattanooga, TN, ACM Press.