csta Voice

Computational Thinking

The Voice of K–12 Computer Science Education and its Educators

Volume 7, Issue 2 May 2011

Inside This Issue

FEATURES

Is Computational Thinking the Fourth "R"?
Defining Computational Thinking for K–12
Building the Future of CS with Blocks

COLUMNS

Classroom Tools
Out and About the Community
Chapter Highlights
College Connection
Show Me the Numbers

INFO BRIEFS

CS & IT Conference
Staff
Membership Renewal
Contribute to the CSTA Voice
Contact Info
CSTA Thanks
Meet the Authors
Mark Your Calendar
Resources

IN THE NEXT ISSUE OF THE *VOICE*

Your CSTA Membership

Is Computational Thinking the Fourth "R"?

Joan Peckham

TO FUNCTION WELL AS CITIZENS and as scientists in a technology-driven world and workplace, we need to use, understand, and navigate a changing landscape. To this end, we have called for and embraced the building of a national cyberinfrastructure to enable us to compete on a global level and to carry out our lives in a secure and healthy manner. At the same time, many of us worry that one part of the national infrastructure has been overlooked—the thinking skills needed to thrive in this new environment.

In 2006, Jeannette Wing popularized computational thinking (CT) to promote the identification and inclusion of thinking skills that arise from the computing disciplines, but that are not currently a part of the "Three Rs"—the Fourth "R" if you will. Jeannette's assertion is that CT skills are needed by everyone, in all disciplines, and in all facets of our lives.

Informally we might define CT as a set of thinking or problem-solving strategies that are used to solve problems when working with computers, and that can also be applied to many problem types, even absent the computer. Since 2006, many of us provided preliminary definitions and examples of CT and then floated them in public. Indeed, NSF among others has funded workshops to determine the nature of CT and created several programs in which investigators are asked to identify CT constructs and test them in formal and informal experiments. The definition is emergent and it will take the participation of multiple communities, including computing researchers and teachers, teachers of all disciplines, cognitive experts, and educational researchers.

One example of CT comes from the efforts of social scientists to examine the multitasking in which we all engage in our daily lives and workplaces. Is multitasking making us more effective or ineffective? An article in the *Christian Science Monitor* in 2008 observed that computer scientists grappled with the problem of multitasking early in the development of operating systems. In particular, they had to develop software that was capable of managing the multiple processes that are running on a computer at any given time. Why not use the approaches these computer scientists employed to solve the human multitasking problem? Indeed computer scientists have long recognized the parallels between human and machine problem formulation and solution strategies. Here we have an example of how we might apply CT to a social science research problem!

Since 2006, there have been many efforts to provide a more precise definition of CT. The first major attempt was by the National Academies in two workshops on Computational Thinking for Everyone. The report of the first workshop provides a long list of ideas, rather disjointed, but to be expected when uncovering essential strategies from a relatively new and broad discipline. The second report is still pending. The current effort to develop a new Advanced *continued on page 2*

IS COMPUTATIONAL THINKING THE FOURTH "R"?

continued from page 1

Placement Computer Science (AP CS) exam has identified a framework needed for CS students and acknowledges that there is much more to computing than programming. Margolis and Goode have developed a first course in computing for high school students—*Exploring Computer Science* (*www.exploringcs.org*)—and they are now working to identify the core CT constructs presented in their course. ISTE and CSTA have convened a series of two workshops to define an operational definition of CT for K–12 (see article by Stephenson and Barr in this issue). The result, work by computer scientists and education experts, is perhaps the most accessible definition to date. Much of this information can be found on the NSF CE-21 portal (*www.computingportal.org/CE21*). CE-21 is a program at NSF to broaden participation in computing, to identify CT learning objectives, and to test and broadly deploy strategies in K–14.

With all of these new ideas, there have been doubters. After early formulations of CT, some were alarmed that we might be proposing to teach our children to think like computers. Instead, we are proposing that students learn the thinking skills required to solve the deepest and most pressing problems of our times. Computational thinkers are masters of computers and solvers of deep problems using skills with which we have yet to imbue computers.

Others, hearing of the abstraction and modeling skills thought to be a part of CT, have asserted that students of mathematics and science already learn these skills. However, many science, mathematics, and engineering abstraction skills are only taught in graduate school, thus are currently inaccessible to the general population. Through the introduction of computers, these thinking skills are now needed and are accessible to everyone.

Also, while abstraction is a fundamental human approach for managing complexity, a new class of abstraction skills have emerged that are particularly suited to problem solving in this age of computing. For example, object-oriented software design makes heavy use of abstraction to design large and complex software systems that are reusable and easy to maintain. The principal approach is to design the software as a system of interacting objects to be manipulated on different abstraction levels. The object-oriented design paradigm could be scaffolded and taught to very young students, and has broad application.

Computational thinkers are masters of computers and solvers of deep problems using skills with which we have yet to imbue computers.

This has been a general introduction to CT. In the next issue of the *Voice*, I will describe emerging definitions, connect them to a few concrete examples, and point to additional CT resources.

CSTA Voice ISSN: 1555-2128

CSTA Voice is a publication of the Computer Science Teachers Association.

CSTA Voice is a quarterly publication for members of the Computer Science Teachers Association. It provides analysis and commentary on issues relating to K–12 computer science education, resources for educators, and information for members. The publication supports CSTA's mission to promote the teaching of computer science and other computing disciplines.

Change of Address and Membership Questions: Contact Member Services via e-mail at *cstahelp@csta.acm.org*, or call 1-800-342-6626 (U.S. & Canada) or +1-212-626-0500 (Global).

Criteria for submitting articles: Potential writers for CSTA should send a brief description of the proposed article, estimated word count, statement of value to members, author's name & brief bio/background info, and suggested title to the editor at *cstapubs@csta.acm.org*. The final length, due date and title will be negotiated for chosen articles.

Notice to Authors Contributing to CSTA Newsletter: By submitting your article for distribution in this publication, you hereby grant to CSTA the following non-exclusive, perpetual, worldwide rights:

- to publish in print on condition of acceptance by the editor
- to digitize and post your article in the electronic version of this publication
- to allow users to copy and distribute the article for noncommercial, educational or research purposes

However, as a contributing author, you retain copyright to your article and CSTA will make every effort to refer requests for commercial use directly to you.

Defining Computational Thinking for K–12

Chris Stephenson and Valerie Barr

WHEN JEANETTE WING launched a discussion regarding the role of computational thinking (CT) across all disciplines, she ignited a profound engagement with the core questions of what computer science is and what it might contribute to solving problems across the spectrum of human inquiry. Wing argued that advances in computing allow researchers across all disciplines to envision new problem-solving strategies and to test new solutions in both the virtual and real world.

In the summer of 2009, the Computer Science Teachers Association (CSTA) and the International Society for Technology in Education (ISTE) began a multi-phase project supported by the National Science Foundation, aimed at developing an operational definition of CT for K–12.

Developing an operational definition of, or approach to, CT that is suitable for K–12 is especially challenging because there is, as yet, no widely agreed upon definition of CT. In addition, to be useful, this definition must ultimately be coupled with examples that demonstrate how CT can be incorporated in the classroom. The primary work of the project was therefore carried out during two workshops; the first focused on developing a shared understanding of CT, and the second on creating exemplar resources and strategies that would support the implementation of CT concepts and skills across grade levels and subject areas.

In attempting to define what distinguishes CT from other problem-solving methods, the educators involved in this project (more than 30 educators representing all grade levels and multiple subject areas) focused on the centrality of the computer and a set of concepts encompassed by CT.

"CT is an approach to solving problems in a way that can be implemented with a computer. Students become not merely tool users but tool builders. They use a set of concepts, such as abstraction, recursion, and iteration, to process and analyze data, and to create real and virtual artifacts. CT is a problem-solving methodology that can be automated and transferred and applied across subjects."

They also envisioned CT manifesting in the classroom through active problem solving. They saw students "engaged in using tools to solve problems", "comfortable with trial and error", and working in "an atmosphere of figuring things out together".

The project, while not yet complete, has already resulted in the creation of a number of useful artifacts, perhaps the most important being the following operational definition of CT.

CT is a problem-solving process that includes (but is not limited to) the following characteristics:

- *Formulating problems in a way that enables us to use a computer and other tools to help solve them;*
- *Logically organizing and analyzing data;*
- *Representing data through abstractions such as models and simulations;*
- *Automating solutions through algorithmic thinking (a series of ordered steps);*
- *Identifying, analyzing, and implementing possible solutions with the goal of achieving the most efficient and effective combination of steps and resources; and*
- *Generalizing and transferring this problem solving process to a wide variety of problems.*

It has also led to identification of a number of dispositions or attitudes that are essential dimensions of CT. These dispositions or attitudes include:

- Confidence in dealing with complexity;
- Persistence in working with difficult problems;
- Tolerance for ambiguity;
- The ability to deal with open-ended problems; and
- The ability to communicate and work with others to achieve a common goal or solution.

In addition to these definitions, the project team is creating a number of resources to support the implementation of CT in K–12. These include:

- A table identifying core CT concepts and capabilities and providing examples of how *continued on page 4*

Contribute to the CSTA *Voice*

The editorial board of the **CSTA** ***Voice*** is dedicated to ensuring that this publication reflects the interests, needs, and talents of the **CSTA** membership. Please consider sharing your expertise and love for computer science education by contributing newsletter content.

Potential writers for the **CSTA** ***Voice*** should send a brief description of the proposed article, estimated word count, statement of value to members, author's name and brief bio/background info, and suggested title to the editor at: *cstapubs@csta.acm.org*. The final length, due date, and title will be negotiated for chosen articles. Please share your knowledge.

Volunteer today!

The CSTA *Voice* welcomes your comments.

E-MAIL: *cstapubs@csta.acm.org*
PHONE: 1-608-436-3050
FAX: 1-928-855-4258

Letters to the Editor are limited to 200 words and may be edited for clarification.

CSTA thanks

The Web Repository Cataloging Volunteers

LEADERS:

Joe Kmoch
Debbie Carter

Dave Burkhart
Renee Ciezki
Myra Deister
Rebecca Dovi
Ria Galanos
Jacqueline Harper
Mindy Hart
Stephanie Hoeppner
Michelle Hutton
Debbie Klipp
Karen Lang
Eugene Lemon
Ron Martorelli
Tim McMichael
Daniel Moix
Deepa Muralidhar
Jill Pala
Margot Phillipps
Tammy Pirmann
Johanna Rivera
Cihan Taktak
Ronald Tenison
Angie Thorne
Jeannie Turner

DEFINING COMPUTATIONAL THINKING FOR K–12

continued from page 3

they might be embedded in activities across multiple disciplines;
- An implementation strategy matrix that identifies activities and outcomes by key stakeholders in the short term, midterm, and long term;
- A set of exemplar activities for embedding CT across grade bands and subject areas;
- A one-page flier describing CT; and
- Key articles written by members of the project team.

All of these resources will be made available on the CSTA website (*csta.acm.org*). We strongly encourage you to review these resources and join the conversation about how we can best enable all of our students to incorporate CT concepts and skills into their knowledge base.

This project is supported by the National Science Foundation under Grant Nos. 0964217 and 1030054.

Computational Thinking for Game Design

Alexander Repenning and Andri Ioannidou

HOW DO PEOPLE CONCEPTUALIZE COMPUTATION? What tools would allow people with no programming experience to build sophisticated simulations and games? For 15 years, supported by the National Science Foundation, we have researched these questions. The *AgentSheets* software authoring environment resulting from this research has been used worldwide by students in curricular and extracurricular contexts.

Scalable refers to the scope of applications, starting with simple game design in middle schools and advancing along a gentle learning slope, all the way to graduate school.

Scalable Game Design is our latest curriculum design initiative that enhances K–12 education with game design-based curriculum and teacher training aligned with standards. Scalable refers to the scope of applications, starting with simple game design in middle schools and advancing along a gentle learning slope, all the way to graduate school. We have started to use the notion of computational thinking (CT) tools as a combination of curriculum based on CT skills, such as *Scalable Game Design*, and technical authoring tools, such as *AgentSheets*.

Based on our experiences, we have identified requirements for effectively including computer science (CS) education into schools and we created a checklist to assess the effectiveness of CT teaching strategies.

› **Low Threshold:** To make CS inclusive to women and other underrepresented groups, an ideal strategy may be to make it an integral part of existing required courses, such as early exploratory topics. In this context, it is typically feasible to squeeze in a one-week (five hours) module. To be successful it must be possible for students to complete a project, such as the *Frogger* game, in this allowed time. If the project is hard to build and the game design activities become frustrating, little progress towards building CT skills is achieved. With *AgentSheets*, most students finish *Frogger* in the first two hours, and additional game creation activities follow.

› **High Ceiling:** If the students cannot make interesting, playable games, then their initial excitement quickly gives way to big disappointment. Students need ways to build games with artificial intelligence, math, and complex behavior. How can my characters find the shortest path in a maze? How can I make them collaborate or compete? This type of sophistication may seem out of the reach of middle school students but we have found ways to scaffold game design with CT patterns to the point where they can build games that, not too long ago, would have

required much more advanced CS skills.

› **Scaffolds Flow:** Optimal flow in game design requires balancing design challenges and developing skills by scaffolding the process with well-defined stepping stones (Csikszentmihalyi, 1990). Increasingly complex patterns and interactions are gradually introduced to students. These patterns can be used to sequence a number of games, starting with a simple *Frogger* game all the way up to *Sims*-like games and complex computational science applications. CT tools, coupled with curricula designed in anticipation of the next challenge, deliver a scaffolded gentle-slope learning trajectory.

› **Enables Transfer:** Teachers can ask their students, "Now that you can make *Space Invaders,* can you build a science simulation?" Perhaps, this question really gets to the core of CT. This is what educators believe learning should be able to achieve: transfer! How can game design skills transfer to model building? One way is to develop a higher-level CT pattern inventory. This inventory is a collection of conceptual patterns that are transferable between a number of applications. To support this kind of transfer, CT tools need to include functionality relevant to game design as well as to science simulation design. Examples of functionality include visualization tools such as plotters and tools to export data into other tools such as spreadsheets.

› **Supports Equity:** To be effective, tools must motivate and educate students across ethnicity and gender, in a variety of educational settings from elective classes to required courses. Formal studies such as an independent research study by the Stanford School of Education (Walter, 2007), concluded that both boys and girls express the same high levels of desire to continue with game design using *AgentSheets*. Teachers report: a) increased participation by girls in computer classes, and b) both boys and girls are so energized after using *AgentSheets* that they go to the counseling office to put computers as their first elective choice.

› **Systemic & Sustainable:** For computational tools to be successfully integrated in K–12 education, they need to be systemically adopted by schools and districts. Students are intrinsically motivated by game design activities to engage in problem solving, including accessing, compiling, and integrating information, outcomes that are consistent with those suggested by the *ACM K-12 CS Model Curriculum*. We have developed teacher training and curriculum aligned with ISTE NETS standards and have integrated *AgentSheets* into the middle school computer education curriculum of entire school districts. *AgentSheets* Inc. and the University of Colorado are currently collaborating on an NSF-funded project that implements *Scalable Game Design* in schools in diverse areas in Colorado (technology hub, inner-city, rural, and remote/tribal areas).

Visit the *Scalable Game Design* wiki (*scalablegamedesign.cs.colorado.edu*) and *AgentSheets* website (*www.agentsheets.com*) to see what students are doing, try some of the tutorials, and provide feedback.

References:

Csikszentmihalyi, M. (1990). Flow: The psychology of optimal experience. New York: Harper Collins Publishers.

Walter, S. Barron, B. Forssell, K. & Martin, C. (2007). Continuing Motivation for Game Design. *CHI*, 2007, 2735-2740.

CSTA Member in the News:
Evelyn Torres-Rangel
LA Lakers Teacher of the Month

Evelyn is a teacher at Gabrielino High School and president of the Southern California CSTA. She has led her school's award winning MESA (Math, Engineering, and Science Achievement) program for the past 25 years. Selection of an award recipient is based on exemplary teaching.

Meet the Authors

Valerie Barr
Union College, Schenectady, NY
Valerie is a Professor of CS. She is working on creating a campus-wide computation program that will engage all students in computing, regardless of their field of interest.

Andri Ioannidou
University of Colorado, Boulder
Andri is the Senior Project Manager of AgentSheets Inc. She teaches game design classes at all levels, nationally and internationally.

Joan Peckham
University of Rhode Island
Joan is a Professor of CS. She is currently on rotation at the National Science Foundation, where she is serving as a Program Director in the Office of Cyberinfrastructure.

Alex Repenning
University of Colorado, Boulder
Alex is a Professor of CS and creator of the *AgentSheets* simulation and game-authoring tool. He has been teaching game design courses in the U.S., Europe, and Asia.

Chris Stephenson
Executive Director, CSTA
Chris is a long-time advocate for K–12 CS education. She is the author of several textbooks, white papers, and scholarly articles on CS and adaptive technologies.

Audrey Van Norman
Marketing Specialist, iD Tech Camps
Audrey is responsible for making teachers and students aware of the many programs offered by iD Tech Camps' youth technology summer program, located at 60 prestigious universities in the U.S.

Elaine Kao
Google, Inc.
Elaine is on the Google education team managing several K–12 initiatives, including Exploring Computational Thinking. She has been in the industry for over 12 years and has played a variety of roles, from software engineer to product and program manager.

Classroom Tools

Exploring Computational Thinking at Google

Elaine Kao

Over the past year and a half, a group of California-credentialed teachers and some of our own Google engineers came together to discuss and explore ideas about how to incorporate computational thinking (CT) into the K–12 curriculum to enhance student learning and build this critical 21st Century skill in everyone.

While different sources define CT slightly differently, we define it as a set of skills that software engineers use to write the programs that underlie all computer applications.

- Decomposition: the ability to break down a problem into sub-problems.
- Pattern recognition: the ability to notice similarities, differences, properties, or trends in data.
- Pattern generalization: the ability to extract unnecessary details and generalize those that are necessary in order to define a concept or idea in general terms.
- Algorithm design: the ability to build a repeatable, step-by-step process to solve a particular problem.

Given the increasing prevalence of technology in our day-to-day lives and in most careers, we believe that it is important to raise this base level of understanding in everyone.

To this end, we've developed a program that is focused on exploring CT in and outside of the classroom. Similar to some of our other initiatives in education, including CS4HS (*www.cs4hs.com*) and Google Code University (*code.google.com/edu*), this program is committed to providing educators with access to our curriculum models, resources, and communities. Our goal is to help teachers learn more about CT, discuss it as a strategy for teaching and understanding core curriculum, and easily incorporate CT into their own curriculum in math, science, language, history, and beyond.

The lessons, examples, and programs developed by the team reflect both the teachers' expertise in pedagogy and K–12 curriculum, as well as our engineers' problem-solving techniques that are critical to our industry. Our goal in the coming months is to further expand our curriculum models to demonstrate how CT can work in all subjects, not just in STEM-related subjects, as well as in all grade levels.

Before launching this program, we reached out to several educators and classrooms and had them try our materials. We received valuable feedback that will be used into future versions.

- CT as a strategy for teaching and student learning works well with many subjects, and can easily be incorporated to support the existing K–12 curriculum.
- Models help to call out the specific CT techniques and provide more structure around the topics taught by educators, many of whom were already unknowingly applying CT in their classrooms.
- Including programming exercises in the classroom can significantly enrich a lesson by both challenging the advanced students and motivating the students who have fallen behind.
- Examples provide educators with a means of re-teaching topics that students have struggled with in the past, without simply going through the same lesson that frustrated them earlier.

To learn more about our program or access our CT curriculum materials and other resources, including our moderated discussion forums, visit *www.google.com/edu/ect*.

Out and About the Community

Keeping Kids Active in STEM Year 'Round

Audrey Van Norman

Because computers and technology never go on summer vacation, STEM (science, technology, engineering and mathematics) education should not be relegated to only nine months of the school year either. iD Tech Camps fill the summer void for students ages seven to eighteen with fun technology programs that combine the energy of traditional summer camps with the computer science (CS) skills necessary for success in today's job market.

During the weeklong programs, students learn in small specialized classes and apply new CS skills towards the creation of a final project. The camp environment makes STEM subjects more accessible to students by putting technology in the exciting contexts of robotics, mobile programming, web design, film production, sports and technology, and a variety of game design and development topics.

Learning in a camp environment is also effective because it builds on students' existing interests—gaming, for instance. When students see the jump from simply playing video games to designing them, their hobbies suddenly become the possible foundations for careers. Pete Ingram-Cauchi, CEO of iD Tech Camps, believes that encouraging STEM education early is the best way to prepare for today's job market. "When you are 13 years old and learning to program your own iPhone apps—that is just cool. It gets you ready for the future. It is relevant."

CS skills are increasingly important for all students. According to the Entertainment Software Association 2010 report, the video game software industry grew at an annual rate of 8.7% between 2005 and 2009. The U.S. Bureau of Labor Statistics estimates that employment of computer software engineers will grow by 32% through 2018, much faster than the average for all occupations.

The iD Tech Camps programs are held at 60 universities throughout the U.S., including MIT, Harvard, Princeton, Stanford, and Carnegie Mellon. Campers can participate in innovative programs including an introductory programming course for kids as young as seven, Teen Academies during which students build a professional portfolio, and iD 365, a monthly series of online workshops like *Android App Development*.

For more information visit: *www.internalDrive.com*.

Chapter Highlights

Great Job CSTA-San Diego!

CSTA's San Diego Chapter is at the heart of a new partnership and a recent symposium dedicated to reintroducing computer science (CS) curricula into all San Diego County schools in the coming years.

The partnership consists of the San Diego Chapter of CSTA, the University of California San Diego, and the University's San Diego Supercomputer Center (SDSC) Education Team, led by Director Diane Baxter. SDSC hosted the symposium where:

- SDSC Director, Michael Norman, explained the relationship of CS with astrophysics. For example, "seeing" the earliest time in the existence of the universe requires virtual imaging since it is inaccessible through the visible or infrared spectrum.

- Gail Chapman provided an update on the Los Angeles Unified School District *Exploring Computer Science* introductory course (*www.exploringcs.org*). Gail, who co-authored the curriculum, has worked to have it included in the district curriculum and was candid concerning both the accomplishments and the challenges faced.
- Beth Simon, Lecturer with UCSD's Computer Science and Engineering Department and newly appointed Director of the Center for Teaching Development, described the new *Computer Science Principles* course at UCSD. Beth, who is teaching one of five national pilot curricula for the proposed *AP CS Principles* course (*csprinciples.org*), discussed her curriculum, pedagogical approach, and outcomes.

The three largest public school districts in San Diego County were represented by board trustees, administrators, and curriculum personnel, who contributed to a discussion of the shared SDSC and CSTA goals and the potential incorporation of the *Exploring CS* and the *AP CS Principles* courses in the San Diego area.

Future meetings will involve dialogues and planning with district leadership to lay the foundation for district CS curricula. The partnership will provide the resources and staff training and districts will work at the local level by encouraging staff, analyzing student needs, and examining systemic elements such as scheduling. Learn more at: *sandiegocsta.org*.

College Connection

Rochester Institute of Technology

Editor's note: *This dialogue with Paul Tymann, Chair of the Department of Computer Science at the Rochester Institute of Technology, is a continuation of our series of interviews with CSTA institutional members. Please share these details about the CS programs at RIT (*www.rit.edu*) with your students.*

The Rochester Institute of Technology (RIT) is one of the world's leading technological institutions. Enrolling approximately 17,000 students and located in Rochester, NY, RIT is comprised of nine colleges, and has more than a dozen graduate and undergraduate programs in computing and computing-related disciplines. The hub of computing education at RIT is the B. Thomas Golisano College of Computing and Information Sciences, which enrolls approximately 2,500 undergraduates and 600 graduate students, making it one of the largest producers of computing graduates in the world. The college offers graduate and undergraduate programs including a Ph.D. in Computing and Information Sciences.

CSTA: What draws students to your program and what keeps them there?
Tymann: RIT has a reputation as an outstanding academic institution with faculty committed to teaching and state-of-the art facilities. Additionally, RIT emphasizes experiential education, especially cooperative education where students gain paid, practical experience in their field before graduation. Students graduate with as much as one year of paid work experience in their field.

CSTA: Tell us about innovative majors or programs of study.
Tymann: In addition to the more traditional programs like CS and IT, the B. Thomas Golisano College of Computing and Information Sciences offers programs in software engineering, game design and development, information security and forensics, medical informatics, and new media interactive development. Innovative programs in other colleges include bioinformatics and computer animation.

CSTA: What cool careers are your graduates prepared for?
Tymann: The sky is really the limit for CS students at RIT. There is not a discipline around that does not use computers. Our graduates work in a wide variety of fields including traditional software development, game development, and computer effects for movies. Our students are also well-prepared to pursue graduate education and advanced degrees in computing.

CSTA: What distinguishes your program from others?
Tymann: In addition to career-focused technological education, RIT faculty are educators first and researchers second. There are many opportunities for students to get involved in a wide variety of projects outside of the classroom. RIT has such a breadth of disciplines across its nine colleges that students enjoy interactions in and out of the classroom with students from diverse backgrounds and academic interests.

CSTA: Tell us a bit about the social environment.
Tymann: When most people think of CS majors they instantly envision of the classical stereotype of the lone CS student sitting in front of a terminal and coding. In reality, very few software systems are written by individuals; they are built by teams of computer scientists. Accordingly, at RIT our students often work in teams. In our second course for majors, for example, students work in teams of three to write the logic for a board game. At the end of the term we host the "Battle Royale" that pits the programs written by one team against another.

Outside of the classroom, the Computer Science Community (CSC) serves as a community for learning, support, friendship, social activity, and mentorship. CSC hosts workshops on new computing technologies, information sessions with various employers, and pizza nights that feature board games or laser tag.

Among the residence halls, Computer Science House, founded in 1976, is one of the oldest and most popular Special Interest Houses at RIT. Located on the third floor of Nathaniel Rochester Hall, CSH provides an innovative living environment for over fifty students and a gathering place for many more that live off-floor. Students develop special projects as part of this experience. Projects have included the very first internet-based soda machine, a drink delivering robot, and a multi-touch display.

SHOW ME THE NUMBERS

VOLUME OF COMPUTATIONAL THINKING REFERENCES FOUND WITH AN INTERNET SEARCH ENGINE

Computational thinking (CT)	3,440,000
CT examples	2,580,000
Teaching CT	2,090,000
CT and university	1,920,000
CT definition	652,000
CT and K–12	49,600

csta Voice
We're on the Web! *csta.acm.org*

Computer Science Teachers Association
2 Penn Plaza, Suite 701
New York, NY 10121-0701

MARK YOUR CALENDAR

NCWIT Aspiration in Computing
Various local and state award event dates
www.ncwit.org/work.awards.aspiration.find.html

Alabama Computer Camps
June 6–24, 2011 in Tuscaloosa, Alabama
www.cs.ua.edu/outreach/camps

CS & IT Conference
July 11–13, 2011 in New York City
www.csitsymposium.org

CH4HS: Carnegie Mellon
July 6–8, 2011 in Pittsburgh, Pennsylvania
www.cs.cmu.edu/cs4hs

Microsoft Innovative Educator Institutes
June 21–22 and 23–24, 2011 in Rochester, New York
www.microsoft.com/education/training/microsoftinstitute/teachers/Innovative_Educator_Program/default.aspx

Consortium for Computing Sciences in Colleges (CCSC: Midwestern)
September 23–24, 2011 in Huntington, Indiana
www.ccsc.org/midwest/Conference

Consortium for Computing Sciences in Colleges (CCSC: Northwestern)
October 7–8, 2011 in Richland, Washington
www.ccsc.org/northwest

Consortium for Computing Sciences in Colleges (CCSC: Rocky Mountain)
October 14–15, 2011 in Orem, Utah
www.ccsc.org/rockymt

Consortium for Computing Sciences in Colleges (CCSC: Eastern)
October 14–15, 2011 in Arlington, Virginia
www.ccsc-e2011.org

Grace Hopper Celebration of Women in Computing
November 9–12, 2011 in Portland, Oregon
gracehopper.org/2011

Consortium for Computing Sciences in Colleges (CCSC: Southeastern)
November 11–12, 2011 in Greenville, South Carolina
www.ccscse.org

RESOURCES

Here's more information on topics covered in this issue of the *CSTA Voice*.

Page 1: Computer Science Unplugged *csunplugged.org*
Page 1: Discussions: Jon Udell with Jeannette Wing *itc.conversationsnetwork.org/shows/detail1844.html*
Page 1: Discussions: Jon Udell with Joan Peckham *www.conversationsnetwork.org/shows/detail4094.html*
Page 2: CE-21 Portal *www.computingportal.org/CE21*
Page 2: CS & IT Conference *www.csitsymposium.org*
Page 3: CSTA *csta.acm.org*
Page 4: Agent Sheets *www.agentsheets.com*
Page 4: CSTA Source Web Repository *csta.acm.org/WebRepository/WebRepository.html*
Page 5: LA Lakers Teacher of the Month *www.nba.com/lakers/community/1011seom_february.html*
Page 5: Gabrielino High School *www.gabrielino.sgusd.k12.ca.us*
Page 6: Google Exploring Computational Thinking *www.google.com/edu/ect*
Page 6: iD Tech Camps *www.internalDrive.com*
Page 6: CSTA San Deigo *sandiegocsta.org*
Page 6: San Diego Supercomputer Center *education.sdsc.edu/teachertech*
Page 7: CS Principles *csprinciples.org*
Page 7: Exploring CS *www.exploringcs.org*
Page 7: CSTA Chapters *www.csta.acm.org/About/sub/CSTAChapters.html*
Page 7: Rochester Institute of Technology *www.rit.edu*