csta Voice

Best Practices

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

Volume 9, Issue 1 March 2013

Inside This Issue

IN THE NEXT ISSUE OF THE *VOICE*

Professional Development

Authentic Learning and Computer Science

The Perfect Fit

Patrice Gans

Editor's note: *This is part one of a two-part series on authentic learning in the computer science (CS) classroom. Part one focuses on activities most appropriate for K–8 students.*

Research has consistently shown that the best learning does not occur when students are expected to absorb facts and figures. Instead, the *continued on page 2*

CSTA THANKS

Debbie Carter, Joe Kmoch, and Boots Cassel for their work on the CSTA Source Repository transition team.

csta.acm.org/WebRepository/WebRepository.html

AUTHENTIC LEARNING AND COMPUTER SCIENCE
continued from page 1

ideal learning environment is one where students are provided opportunities for active exploration connected to practical problems and events.

> *From the standpoint of the child, the great waste in school comes from his inability to utilize the experience he gets outside while on the other hand he is unable to apply in daily life what he is learning in school. That is the isolation of the school—its isolation from life.*
> —*John Dewey, 1916*

Today's educators look to authentic learning opportunities to bridge classroom learning with real-life experiences. As Ted Lai, Director of Information Technology for the Fullerton Elementary School District, notes, "At the heart of 21st Century Learning, in my opinion, is the piece on creating authentic projects and constructing knowledge… essentially making connections between learning and the real world."

So what exactly are authentic projects? Authentic learning typically focuses on concrete, complex problems and their solutions, using role-playing exercises, problem-based activities, case studies, and participation in virtual communities of practice. The key characteristics of authentic learning include:

- Learning is centered on authentic tasks that are of interest to the learners;
- Students are engaged in exploration and inquiry;
- Learning is often interdisciplinary;
- Students become engaged in complex tasks and higher-order thinking skills such as analyzing, synthesizing, designing, manipulating and evaluating information;
- Students produce a product that can be shared with an audience outside the classroom;
- Learning is student-driven with teachers, parents, and outside experts all assisting and coaching in the learning process;
- Students have opportunities for social discourse.

Authentic learning projects are a perfect fit for CS instruction. Integrating CS into core academic subjects via hands-on applications helps to solidify the learning experience. It also helps to reinforce the important role that computer technology plays in society. Understanding the connection between computer systems and business, scientific, or engineering problems goes a long way towards preparing our students for the future.

Many college CS curricula recognize the importance of authentic learning and, consequently, provide robust learning experiences which offer students with the opportunity to make exciting things happen (e.g., write programs that control their cell phones or robots, create physics and biology simulations, or compose music). Luckily, K–12 CS education is eagerly adopting similar curricula.

As a K–8 computer teacher, I recognized early on the importance of tapping my students' personal interests as a way to ensure their engagement in CS. At the beginning of the 2011 school year, I introduced them to the basic concepts of programming via Scratch. As the school year progressed, I decided to include

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.

other CS methodologies. With this goal in mind, I spent the last few weeks of the 2012 school year teaching HTML.

I was amazed at their enthusiasm. The fact that they could create a website and then share the information with their families, friends, and even virtual strangers via the Internet, was a very powerful motivator. They couldn't wait to get started. And even when they encountered challenges along the way, their sense of accomplishment never wavered. I used the Go Berserk material as a guide to teach my students the fundamentals of HTML. The authors are teachers from Northern Ireland. They were inspired to develop the curriculum and the corresponding book (and website) by teaching primary and secondary school children to make their own websites. I used their book with my fourth, fifth, and sixth grade students.

The after-school program called Project GUTS (Growing Up Thinking Scientifically) provides an excellent example of an authentic learning curriculum for middle school students. Based in New Mexico, Project GUTS invites middle school students to investigate topics of interest (e.g., epidemiology, bark beetle infestation, and traffic) within their local communities. The students, with the assistance of GUTS facilitators, use technology to explore these real-world problems. According to the GUTS website (*projectguts.org*), the emphasis of the program is to teach students to look at the world and ask questions, develop answers to the questions through scientific inquiry, and use critical thinking to assess which ideas are reasonable and which are not.

The program calls for students to investigate a problem, interview experts and community members, gather data, and run experiments and simulations with their computer models (using StarLogoTNG and Netlogo) to better understand the problem being studied. "We encourage the students to look at their projects with an eye to the complexity of the problem: in other words, instead of looking for a 'right answer' we direct them towards trying different initial conditions, building randomness into the variables, and observing many, many runs of their code." explains GUTS Investigator Betsy Frederick.

Opportunities abound for active exploration connected to practical problems and events for even the youngest CS students.

LEARN MORE:

Go Berserk
www.go-berserk.com/aboutUs.php

Meridan: A Middle School Computer Technologies Journal
www.ncsu.edu/meridian/win2003/authentic_learning/index.html

NetLogo
ccl.northwestern.edu/netlogo/

StarLogo TNG
education.mit.edu/projects/starlogo-tng

Blocks Programming

Michael Tempel

THE GROWING POPULARITY OF SCRATCH over the past six years has encouraged the widespread use of a particular style of visual programming known as blocks programming. Programs are created by snapping together graphical pieces, like putting together a jigsaw puzzle. Because of the ease and comfort in getting started, blocks programming makes computing accessible to a much wider audience.

The first blocks programming language was Logo Blocks. Developed in 1996 at the MIT Media Lab, this graphical version of Cricket Logo controlled programmable bricks, small microcontrollers that were used in robotics projects. Blocks programming is also found in Turtle Art, PICO Blocks, MIT App Inventor, StarLogo TNG, and Alice, among others.

Scratch programming features sprites, screen objects that may be animated and act as characters in a story or game. For example, we may have a cat sprite and a mouse sprite. The words of the script are displayed on the screen in what appear as snap-together blocks. A cat and mouse chase scene script could cause the cat to move around the screen following specific instructions for pointing and moving. A repeat block would *continued on page 4*

BLOCKS PROGRAMMING

continued from page 3

cause the action to be repeated a set number of times. If the cat touches the mouse during the repeat structure, a "gotcha" message could be displayed. If the mouse hasn't been caught after 10 repeats, the cat sprite says "You win!" and the script ends.

The blocks provide a visual representation of the program, which for many people is more understandable than text. A block's shape gives an indication of its purpose. The way in which the blocks are assembled shows the flow of the program.

Most blocks have a notch at the top and a tab at the bottom. These "stack blocks" may be put together, forming a sequence of instructions. The "repeat" block has a "C" shape, with the sequence of instructions to be repeated fitting into the "C". The "if" block has a similar shape enclosing instructions to be executed when a specific condition is met.

Some of these blocks, such as the "move" block, require inputs. We can click in the number field and type a number. For the "point in direction" block, rather than use a constant as the input, we can use a "reporter block" which returns a random number between 1 and 360. This block has a rounded rectangle shape so as to fit into the number field of the "point in direction" block. The "if" block needs a Boolean input, which is provided by the "touching?" block. It reports "true" if the cat touches the mouse and "false" if the two sprites are not touching. The hexagonal shape of this block allows it to fit properly into the "if" block.

A beneficial side effect of having the shapes of the blocks be related to their uses is that it is largely impossible to make syntax errors. If we try to put the "point in direction" block into the "if" block, it will not fit because it is the wrong shape. Similarly, we cannot put the "touching?" block into the "point in direction" block. In this way, most syntax errors are precluded.

The blocks are also grouped by color. Blocks that move sprites are blue. Control blocks like "repeat" and "if" are yellow, and so on. Categorization by color provides an additional visual clue to the actions of various components of a program.

There are limitations to blocks programming. Because of the graphical representation, a large program takes up a lot of screen real estate and can be hard to follow. But as with text-based programming languages, one can create procedures (functions) that allow a body of code to be collected and called by name. Procedures can be complex, and may include other user-created blocks as sub-procedures. The definition of a block may include the block being defined so as to create a recursive procedure.

> A beneficial side effect of having the shapes of the blocks be related to their uses is that it is largely impossible to make syntax errors.

One can go quite far with blocks programming. Snap!, an extended version of Scratch, is being used to teach computer science (CS) at the high school and college level, including some of the pilot sites for the new AP CS Principles course.

LEARN MORE:

Alice
www.alice.org/

App Inventor
appinventor.mit.edu/

LogoBlocks
research.microsoft.com/en-us/um/people/abegel/mit/begel-aup.pdf

PICO Blocks
www.picocricket.com/

Snap
snap.berkeley.edu/

Super Cricket
handyboard.com/cricket/software/

TurtleArt
turtleart.org/

Scratch
scratch.mit.edu

Scratch Arduino
seaside.citilab.eu/scratch/arduino

ScratchEd
scratched.media.mit.edu/

StarLogo
education.mit.edu/projects/starlogo-tng

From F1 to Mach 10

Programming in Real Life

Brooke Wehrmann

PROGRAMMING TOOLS that are not used in real-world work outside of the computer science (CS) classroom often require a difficult translation between what students see and do in the classroom to what they see and do on the AP exam, in college, or in the workplace. At National Instruments (NI) we have used our work with scientists, academics, and engineers to create a graphical programming language called LabVIEW™ that helps students make these critical connections.

Many exciting technological events occurred this past year including the SpaceX Dragon shuttle mission and the Red Bull Stratos space jump in which Felix Baumgartner reached an estimated speed of Mach 1.24. These events were made even more exciting for CS teachers when the news broadcasts highlighted the programming behind these record-breaking feats. From the mission control system at SpaceX, to the altitude chamber test before the space jump, CS was essential to their success. Both events relied heavily on LabVIEW.

At NI we know there are some pretty cool applications of LabVIEW programming and we get especially excited when LabVIEW and CS are front and center in the news. Our goal is to get these tools into the hands of students—the next generation computer scientists. Students can write, implement, and test similar code to create a mission control program and explore velocity. With LabVIEW, students see and do in the classroom what computer scientists see and do in the workplace; it helps answer the question, "When will I ever use this?"

More than 30,000 companies also use LabVIEW to accelerate productivity and innovation for projects such as:

- Formula One teams at the University of Texas at Austin measure automobile down force and design cars that will stay close to the ground;
- The National Federation for the Blind partnered with Virginia Tech to program a motor vehicle for drivers who are visually impaired;
- The startup company Windlift delivers mobile wind power to remote villages for post-conflict reconstruction, disaster relief, and third-world development.

LabVIEW engages students in problem solving, algorithmic thinking, and design and innovation while being both accessible for middle school students and rigorous for advanced students. The LabVIEW for Education website (*k12lab.com*) offers more career connections, videos, and lesson plans.

LEARN MORE:

K12Lab CS lessons
k12lab.com/lesson-plans/subjects/computer-science

F1 Cars Shaped by Wind Tunnel
www.kvue.com/home/F1-cars-shaped-by-wind-tunnel-testing-179825951.html

Red Bull Stratos altitude chamber
www.youtube.com/watch?v=39a7oXKK-Pk

Red Bull Stratos space jump
www.youtube.com/watch?v=lB1PTOp1aKk

SpaceX Dragon
k12lab.com/inspiration/spaceX

Congratulations CSTA Members

Barbara Boucher Owens, Southwestern University

Boots Cassel, Villanova University

Steve Cooper, CSTA Chair, Stanford University

As ACM Distinguished Educators for 2012 ***awards.acm.org/homepage.cfm?awd=157***

Meet the Authors

Darren Cambridge
American Institutes for Research
Darren is a senior consultant at the Institutes in Washington, D.C. He directs online social learning and analytics projects, including the U.S. Department of Education's Connected Educators initiative. He is the recipient of the MacArthur Foundation's Digital Media and Learning Faculty Writing Prize for E-Portfolios for Lifelong Learning and Assessment.

Patrice Gans
Newtown, CT
Patrice teaches technology to K–8 students at Fraser-Woods School. She also teaches Scratch in summer programs at Naugatuck Valley Community College. Her goal is to engage all students, beginning in kindergarten, to CS.

Leslie Keller
Apex, NC
Leslie teaches CS at Apex HS and for the North Carolina Virtual Public School. She has recently been involved in updating the CS curriculum for North Carolina. Leslie enjoys creating curriculum around innovative technologies and engaging a more diverse student population.

Michael Tempel
Logo Foundation
Michael is President of the Logo Foundation, a nonprofit organization he founded with MIT Professor Seymour Papert in 1991. He has been developing Logo software and educational programs for more than 30 years.

Brooke Wehrmann
National Instruments
Brooke manages STEM curriculum partnerships at National Instruments. She also mentors high school FIRST robotics teams.

Brenda Wilkerson
Chicago Public Schools
Brenda is the Program Manager of Information Technology in the Office of Pathways to College and Careers, Career and Technical Education, and Early College STEM Schools.

CS10K Update

Your Invitation to the New CS10K Community

Darren Cambridge

The National Science Foundation and the U.S. Department of Education are launching a new online community of practice for teachers of Exploring Computer Science (ECS) and Advanced Placement Computer Science Principles (AP CSP). The new community is designed to support both current and potential teachers of these two courses and computer science (CS) teachers are very excited about its potential.

The community is being developed by the American Institutes for Research (AIR) and led by 13 facilitators, each associated with one of more NSF or College Board funded professional development projects. With AIR's support, the facilitators are working to generate immediate value to participants by ensuring that the community site provides a wealth of teaching resources and prompt replies to posted questions. "The CS10K Community is one of the most exciting things to happen for me in my career. I have been doing computer science (CS) education, outreach, and training support in one form or another for nearly 20 years," says Jeff Sales, Community Facilitator

The community is part of the larger CS10K initiative, which is working to have 10,000 well-trained high school CS teachers in 10,000 high schools by 2015. And membership is open to all. "The creation of this online community of practice is a long awaited project," says Deepa Muralidhar, a teacher from Georgia and a member of CSTA's leadership cohort.

Gail Chapman, the national outreach coordinator for ECS at UCLA, notes that the ultimate goal is for the CS10K community to become a true community of practice rather than just one more resource repository. "We hope also to engage community members in more structured, sustained activities related to course content and pedagogy that generate tangible products or provide opportunities to take on leadership roles", says Chapman.

Teachers participating in one of the existing professional development projects will be able to interact through private project spaces, allowing them to build on the collaborative, reflective, and trusting nature of our face-to-face professional development opportunities.

Teachers are able to join groups discussing issues—such as the assessment of student work, student-centered pedagogy, and student recruitment—or to form their own group to focus on other topics or create teaching resources of particular use to them. For example, teachers might choose to work together to demonstrate how the course curriculum maps to state standards or to adapt materials for use with students with disabilities. "These activities will be a success as teachers gain a sense of empowerment and validation through their participation," says CSTA Chicago chapter president Don Yanek.

The community also is for building one-to-one relationships with peers and experts who can provide coaching and support. Members can indicate their skills, interests, and willingness to help others in their searchable profiles. Trishan De Lanerolle, facilitator from Trinity University, sees these relationships as essential because ECS and CSP tend to be taught by dedicated teachers with varied backgrounds and exposure to the field. "Relationships formed within the community lend support and provide a sounding board," says De Lanerolle.

All CSTA members are invited to participate in the community at *cs10kcommunity.org*, provide feedback, and join the conversation.

Curriculum in Action

Get Game: Attracting Students to the Next Level

Leslie Keller

Attracting students to high school computer science (CS) courses can be a challenge; getting students to sign up for a second course is often even a bigger challenge. In an effort to increase interest and enrollments in computing courses, a team of North Carolina high school computing teachers, along with Deborah Seehorn, NC Department of Public Instruction's Business, Finance, & IT Education Consultant, looked to new *Game Development with XNA: Semester 1* and *Game Development with Advanced XNA, Kinect, and Windows Phone 7: Semester 2* curricula to help restructure the Computer Programming II course.

Using Microsoft Visual C# with the XNA Game Studio Software Development Kit (SDK), students learn how to create Xbox and Windows-based games. The XNA curriculum, developed by Pat Yongpradit, Springbrook High School, MD, is comprised of two semesters of course work and uses Rob Miles' *Learn Programming Now! Microsoft XNA Game Studio 4.0*.

While *Semester 1* is deemed basic, it does assume students already have foundation programming skills such as looping, decision-making, and object-oriented design. *Semester 1* applies these skills and teaches additional concepts and skills specific to game development. *Semester 2* adds programmer-defined classes and offers a project for Microsoft's Imagine Cup competition. Six-week units on Windows Phone and Kinect development are also part of the second semester curriculum.

Programming using the XNA SDK is a different paradigm for students and it takes some adjustment. Within the XNA environment, code executes repeatedly, sixty times per second, unless told to stop. Learning to program within this "game loop" requires students to change the way they think about how their code executes. Students also learn various methods to interact with their games using a wired Xbox controller and/or keyboard as well as how to add additional content, such as music, sounds, and images, to their games. When students use the controllers to interact with their games, they are engaged and delighted with their accomplishments.

Our classes meet for 90 minutes, five days each week for approximately 90 days. In planning the course, we were not certain how long the content of *Semester 1* would take. After teaching the *Semester 1* material, we opted to add additional content to cover creating and using programmer-defined classes from *Semester 2*. The chapter on teaching classes is broken into three parts and requires creative and assiduous effort on the part of both teachers and students.

Classes are a challenge to students even if they have used class objects and methods throughout their programming coursework. There is a leap from creating an existing class object and modifying that object using the class methods to creating a new class and using it effectively. Adding derived classes and advanced topics such as polymorphism and abstraction challenges students to expand their programming skills.

Extending the project requirements by encouraging students to customize their projects, using their own images and sounds, also increases engagement. Students enjoy personalizing their projects and many will exceed the technical expectations. Working in teams using the pair-programming pedagogy also encourages students to bounce ideas off of their teammates and challenge each other to add the next great feature to their projects.

The last two weeks of our course were dedicated to creating a final project, again with a partner. I provided a rubric of technical requirements and feedback/reflection worksheets but gave students free reign as to the topics of their projects. The biggest challenge was keeping students from going "too big." The project encourages students to collaborate and push each other. They not only develop programming skills, but communication and planning skills as well. Most students successfully completed the program they planned; a few were too ambitious for the time constraint, and a few simply did not plan well.

Feedback from students was very positive; they liked using the controllers and programming what they felt was something "real." From a teacher's perspective, it is a fun and challenging course. The XNA materials challenge the students' programming skills but also challenge them to work in teams and to plan and communicate effectively. My goal to increase enrollment was achieved; enrollment for this class is up about 50% over the past year.

LEARN MORE:
Programming with C# and XNA 0.5: Jump Start
Game Development with XNA: Semester 1
Game Development Advanced XNA, Kinect, and Windows Phone 7: Semester 2
www.microsoft.com/education/facultyconnection/precollegiate

Spotlight

Chicago Early College STEM Schools

Chicago CSTA Chapter Making an Impact

Editor's note: *In this article, CSTA visits with Brenda Wilkerson, the Program Manager of the Early College STEM Schools in the Chicago Public Schools (CPS), about the new STEM schools and the role they will play in computer science (CS) education.*

CSTA: Recently, five new STEM high schools opened their doors to students in CPS. What was the thinking behind the creation of the schools and what makes them different from other high schools?
Wilkerson: The STEM high schools are different from all others in that they are the first Early College Schools in CPS geared toward building critical thinkers and focusing on technology careers. Students will experience accelerated education with opportunities to earn dual credits leading to two-year degrees and interviews with some of the top technology CPS-partner companies in the country. The program looks to support student development beyond year 12, fostering post-secondary credential attainment.

CSTA: What is your role in this program?
Wilkerson: My role is to coordinate the conversation between industry, institutions of higher education, CPS administrators, and educator communities, to foster the development of courses of study that create the atmosphere necessary to train, motivate, and incubate tomorrow's technologists. I work with industry and education leaders to create the pathways that will help our students understand the opportunities for innovative and critical thinkers, not only in technology-specific careers, but in careers in all sectors. I also lead the work to develop STEM courses for career pathways.

CSTA: How has the Chicago CSTA chapter been involved?
Wilkerson: The Chicago CSTA Chapter, led by Don Yanek, has been very supportive and proactive in helping CPS identify and train teachers in our quest to increase CS education in CPS. We have worked closely throughout to bring more awareness and opportunity to our students and the communities they represent, and we are thankful for Chicago CSTA's hard work and dedication.

CSTA: Which students would be most interested in a STEM school? What is the application process?
Wilkerson: We believe that these schools offer something to a wide variety of students. The application process is generally very open within neighborhood boundaries first, and then tiered across network and district boundaries. We want to offer equitable opportunities that allow students to explore rigorous and relevant STEM education that focuses on the "T," regardless of their prior educational experience or district location. Recognizing that acceleration in the diverse population we draw from will be key, these schools are designed to layer support and a variety of class experiences, as well as after-school activities, that give students opportunities to grow in areas in which they may have little or no prior experience.

CSTA: What kinds of programs are available to students specifically interested in studying CS?
Wilkerson: We believe that all of our programs are founded in sound CS principals. There are three strands of focus for CPS technology education: web development and programming, networking and securities, and database development and cloud computing. Each freshman student (year 9) experiences the Exploring Computer Science (ECS) course. The purpose of this course is to introduce key CS principles and to teach students to think differently about the work of analyzing, designing, and computing, as they collaborate to identify and solve problems. We believe that these are necessary skills for all students as they prepare for successful post-secondary study and careers.

CSTA: Will students who graduate from these schools benefit in ways that students attending more traditional high schools will not?
Wilkerson: Students will enjoy the support of a corporate mentor to help them draw the connections between their studies, career opportunities, and the challenging work we face as a nation. The opportunity to earn dual-credits as early as year 11 creates an additional relevance to the classes students take from the very first day of their high school education. And through opportunities to communicate with and visit partner colleges, we hope that students will develop a sense of connection between

continued on page 8

SHOW ME THE NUMBERS

CSTA GROWING IN NUMBERS AND INFLUENCE

	MEMBERSHIP	COUNTRIES REPRESENTED	CSTA CHAPTERS
June 2008	4,781	100	5
June 2009	6,374	103	17
June 2010	7,014	105	29
June 2011	9,366	115	35
June 2012	11,574	125	45
Goals 2013	14,000	130	55

Source: CSTA Membership Committee (January 2013)

csta Voice

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

We're on the Web.
Join our Facebook page!
csta.acm.org

CHICAGO EARLY COLLEGE STEM SCHOOLS
continued from page 7

high school and college that will foster an expectation of post high school education and college attendance. Overall, the culture we are planning for our students in these new STEM high schools should add a deeper sense of purpose to their high school experience.

CSTA: Is there a link between the STEM high schools initiative and the current program to implement ECS as a required course in CPS?
Wilkerson: It turns out that the timing of the opening of the STEM high schools actually accelerated our mission to add ECS to our district. We were on track to increase the offering throughout the district when this initiative came up. We are pleased that we had this connection with CSTA and were therefore poised to deploy the curriculum in the STEM high school project.

LEARN MORE:
CPS Early College STEM Schools
www.cps.edu/Spotlight/Pages/Spotlight288.aspx
Chicago CSTA
cstachicago.ning.com

MARK YOUR CALENDAR

SIGCSE 2013
March 6–9, 2013, Denver, Colorado
www.sigcse.org/sigcse2013

USA Computing Olympiad: March Contest
March 8–11, 2013
www.usaco.org

ACSL Contest #3
March 15, 2013
www.acsl.org

International Conference on Education, Training and Informatics: ICETI 2013
March 19–22, 2013, Orlando, Florida
www.iiis2013.org/icsit/website/default.asp?vc=31

Java Fundamentals Teacher Training
March 25–June 28, 2013, Denver, Colorado
academy.oracle.com/pages/prog_commit_inst_institute.htm

Consortium for Computing Sciences in Colleges (Mid-South)
April 5–6, 2013, Fort Smith, Arkansas
www.ccsc-ms.org

Consortium for Computing Sciences in Colleges (Southwestern)
April 5–6, 2013, San Marcos, California
www.ccsc.org/southwestern/index.php

USA Computing Olympiad: US Open
April 5–8, 2013
www.usaco.org

Consortium for Computing Sciences in Colleges (Central Plains)
April 12–13, 2013, Kansas City, Missouri
www.ccsc.org/centralplains

Consortium for Computing Sciences in Colleges (Northeastern)
April 12–13, 2013, Loudonville, New York
ccscne.org/ccscne2013

Java Fundamentals Teacher Training
April 15–July 26, 2013, San Mateo, California
academy.oracle.com/pages/prog_commit_inst_institute.htm

ACSL Contest #4
April 19, 2013
www.acsl.org

Consortium for Computing Sciences in Colleges (South Central)
April 19–20, 2013, Shreveport, Louisiana
www.ccsc.org/southcentral

Annual CSTA Conference (formerly CS & IT)
July 15–16, 2013, Quincy, Massachusetts
www.cstaconference.org