csta Voice

K–8 CS Education

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

Volume 8, Issue 1 March 2012

Inside This Issue

IN THE NEXT ISSUE OF THE *VOICE*

Professional Development

Computer Science as an Art Form

K–5 Fine Arts

Christopher Michaud

COMPUTER SCIENCE (CS) equips students with unique skill sets similar to music and dance. The skills to model reality through numbers, methods, and functions give CS a unique place alongside music, dance, and drama as a way for humans to express creativity.

How do we teach CS to younger students?

continued on page 2

csta Computer Science Teachers Association

You are Invited!
CS & IT Conference 2012

Hyatt Regency Hotel Irvine, CA

July 9: Hands-on Workshops

July 10: Keynotes and Breakouts

The CSTA annual Computer Science & Information Technology Conference provides professional development opportunities for high school and middle school computer science and computer applications teachers who want practical, relevant information to help them prepare their students for the future.

See our terrific line-up of speakers and sessions
www.cstaconference.org

Register today!
www.softconf.com/c/csta2012

CS & IT 2012 is generously sponsored by the Anita Borg Institute, Google, and Microsoft Research.

CSTA THANKS

Baker Franke
Daniel Moix
Deepa Muralidhar
Kelly Powers
Padmaja Bandaru
Stephanie Hoeppner
Tammy Pirmann

for their work with the Leadership Cohort Advocacy Participation Program.

Executive Officers

Stephen Cooper
Chair
cooper@cs.stanford.edu

Michelle Friend
Past Chair
mfh@pobox.com

Staff

Dr. Chris Stephenson
CSTA Executive Director
Phone: 1-800-401-1799
Fax: 1-541-687-1840
cstephenson@csta.acm.org

Pat Phillips
Editor
Phone: 1-608-436-3050
Fax: 1-928-855-4258
cstapubs@csta.acm.org

Committees

Certification
cstacertification@csta.acm.org

Curriculum
cstacurriculum@csta.acm.org

Funding Development
cstagrants@csta.acm.org

Membership
cstahelp@csta.acm.org

Professional Development
cstapd@csta.acm.org

Research
cstaresearch@csta.acm.org

COMPUTER SCIENCE AS AN ART FORM

continued from page 1

We deliver concepts and content sequentially through culturally relevant activities to create and play games, tell stories, and run simulations that equip students with the tools for digital self-expression.

Kindergarten, first-, and second-grade students experience learning through kinesthetic actives of art, dance, music, drama, and sports. TuxPaint (*tuxpaint.org*) or other simple painting programs allow students to express ideas in images, colors, lines, and forms. These activities develop user interface skills, lead students to think about objects in the computer as similar to objects in their drawings, and encourage students to see technology as an expressive tool. I use painting, drawing, and text manipulation through standard office software to teach third-graders the skills of manipulating text, creating and editing images, navigating a file system, and saving and retrieving their work. I teach CS concepts to fourth- and fifth-grade students using Scratch, Lego NXT Robotics, and Python.

CS is taught in the domains of linear and event-driven sequences, methods, and models which are introduced sequentially and built one upon the other. First, students learn to arrange directions or commands in a linear sequence. After students have mastered commands to animate, draw, and drive a robot, we begin to label command sequences that we use over and over again.

Labeling sequences, or creating methods ("methoding"), is the second domain. Creating methods that can be reused represents an important step in understanding how to create a computer program.

The third domain is modeling objects. Through the use of data structures, students create virtual models of objects that "live" inside the program. The picture represents the character in a game or an object in a simulation.

Scratch (*scratch.mit.edu*) provides a graphic programming environment that allows students to explore, create, and remix graphics and sounds through animations, games, and simulations. Students extend the music activities of singing, dancing, and drama onto the screen. Examples include interactive artwork and storytelling. Students use Scratch to model movement with musical selections and use colors and speed of movement to reflect the musical ideas in a selection.

Students further explore modeling concepts by creating virtual representations of musical instruments. We use Scratch to create virtual instruments such as xylophones and drums (*www.nebomusic.net/scratchxylophone.html*). Students use Lego NXT Robotics to perform dance routines and play instruments to take their sequences and programs out of the "screen" and into the real world.

I use the Python programming language to transition my fifth-grade students from Scratch to text-based programming. The JES environment by Georgia Tech (*code.google.com/p/mediacomp-jes*) provides a platform for students to explore Python programming by manipulating pictures and sounds along with "driving" Turtle graphic programming (*www.nebomusic.net/techlesson11-5StudentDirections.html*).

CS is a powerful tool to empower elementary students in the fine arts.

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).

Reproduction Rights Information: No part of this publication may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopy, recording or information storage and retrieval system, without permission in writing from the publisher. Exception: permission to photocopy [individual] items for internal or personal use is hereby granted by CSTA.

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.

Thinking Myself

Kiki Prottsman

THERE IS NO COLLOQUIALISM that says, "If at first you don't succeed, give up and have a sandwich," and it is unlikely that such a phrase will ever become popular, as long as we continue to value innovation and discovery. The idea of "working until you succeed" is embedded in the concepts of scientific methodology, entrepreneurship, and of course, computer science (CS). Unfortunately, the emphasis on graded assessments has made it difficult to encourage children to learn through failure.

Thinking Myself (*games.thinkingmyself.com*) was created to encourage the try-and-try-again model. The site is an educational adventure that encourages children to navigate individually through lessons, while reading and playing games as they go. The freedom of trial-and-error learning is encouraged and the pressure to be perfect is removed from the process. It was developed using preferences and learning styles found to be desirable to girls over eight years of age.

The lessons of *Thinking Myself* encourage learning through explorations of computational thinking concepts. The concepts are simplified and condensed so that young children can easily absorb the lessons without adult explication. After an initial warm-up activity in which students explore an undirected challenge called the "Machine without Instructions," they are guided through exercises on decomposition, patterns, abstraction, and algorithms.

The exercises use computational thinking terminology. Though it may seem inconsistent to simplify the lesson while maintaining the vocabulary, the decision was made to reduce the fear of words so that children would not automatically associate complex words with difficult concepts. *Thinking Myself* aims to tame words that many adults find cringe-worthy.

The lessons of *Thinking Myself* encourage learning through explorations of computational thinking concepts.

In the first lesson, "Decompose," students learn the word decomposition and work to break problems into smaller pieces. After some instruction and an example, students decompose the problem in a sorting game.

In the "Patterns" lesson, students learn about patterns while searching through triangles in a tangram game.

The "Abstraction" lesson introduces ideas from CS, including inputs, outputs, and variables. The game in the lesson combines user input to create a specified output.

In the "Algorithms" lesson, students explore an algorithm in the form of a recipe and are then challenged to follow directions for a pirate treasure adventure.

While *Thinking Myself* is entertaining as a digital resource, all of the concepts can be presented in the classroom with the free and reproducible materials at: *games.thinkingmyself.com/analog*.

Congratulations CSTA Members
Richard Pattis and Marie desJardins
ACM Distinguished Members 2011

Pattis (University of California, Irvine) and desJardins (University of Maryland, Baltimore County) have been identified as Distinguished Members by ACM. The award recognizes members with at least 15 years of professional experience and five years of continuous professional membership and who have made significant impact on the computing field.

www.acm.org/press-room/news-releases/2011/distinguished-2011

csta
Computer Science Teachers Association

Let us know if your contact information changes.

cstephenson@csta.acm.org

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.

ACM founded CSTA as part of its commitment to K–12 computer science education.

MAKE YOUR VOICE HEARD

VOTE!

CSTA Board of Directors Election

Open online April 4

Close May 2

Results May 16

More details at CSTA

csta.acm.org

Teaching Java to Fifth Graders

Vic Wintriss and Aaron VonderHaar

CONVENTIONAL WISDOM holds that ten-year-olds have not yet developed abstract thinking abilities sufficiently to enable them to write computer programs—especially in Java. Our experience has been quite different. At Wintriss Technical Schools, a small, non-profit, after-school program, Java professionals use kid-appropriate terminology, training aids, and curriculum to teach children to write computer games.

Our teaching methods contribute to student success. Unlike classes that try to teach with lectures and boring exercises, our students write complete games from the very first day. Ten volunteer professionals mentor the students to create games such as Hi-Lo, Pong, Tic-Tac-Toe, Memory, and Asteroids. An autonomous robot competition further tests the skills and imagination of students.

Java professionals are willing to volunteer to teach students when they learn that the teaching environment is not a classroom of thirty students. Groups of about three students meet for two hours, once a week, in a classroom with four Apple iMacs. Student-teacher relationship building is important and teachers find that the strong relationships formed in this setting are very rewarding and an important feature of the program. (All of the volunteers are screened via LiveScan.)

My job is to teach the teachers kid-appropriate teaching methods. Our volunteers avoid using scary, technical terminology as much as possible before beginning students learn that they can actually write a sequence of code that produces fun output. Teachers describe how to make a blueprint (Java Class) "real" by using the Java keyword "new", instead of talking about instantiating a Class to create an object of the Class. Teachers also help students conceptualize programming concepts; methods have a "mouth" (the space between the parentheses in a method header), and you feed input parameters into a method's mouth. Methods also have "guts" (the area between the opening and closing curly brace in a method body) where the code for this method goes. The method return type specifies what comes out of the guts!

We also introduce students to real-world contexts for many of the math concepts they are learning in school by using the x-y coordinate system, trigonometry, and simple physics simulations in their games. Because our teachers are Java professionals, they are not limited to pre-scripted projects. Their programming expertise gives them the flexibility to adjust the games on-the-fly.

Five years of teaching Java to students from fifth grade through high school, has enabled us to develop an extensive collection of projects. We use NetBeans as the IDE and Subversion for the source control program. All student work is saved on a school server and accessible from home.

So far, four students have passed the Advanced Placement Computer Science (AP CS) exam; one in the eighth grade, one in the ninth grade, and two in the tenth grade. Students who pass the exam have had the incredible opportunity of being placed in paid, summer intern jobs. Our oldest student graduated from high school this year and is now attending Stanford as a CS major. Most importantly, the students seem to genuinely enjoy the new-found freedom of expression that comes with writing computer programs.

Highlights of CS Survey

State-by-State Analysis

Debbie Carter

IN AN EFFORT TO MEASURE schools' responses to the changing needs of students, CSTA surveys high school computer science (CS) teachers every two years. Results from the 2011 National Secondary Computer Science Survey were posted on the CSTA website last summer, and Duncan Buehl highlighted some of the trends in his article in the November 2011 issue of the CSTA Voice.

In addition to the consolidated results for the United States, the website also

provides summarized results for each of the 29 states from which 15 or more teachers responded. While many states' results are similar to those of the consolidated U. S. figures for many questions, some interesting variations are present.

Introductory (or Pre-AP) CS Courses

In the United States as a whole, introductory CS courses are offered in 69% of the respondents' schools. However, 91% of Maryland's respondents reported that their schools offer introductory CS, while only 50% of Oklahoma's respondents' schools offer such a course. Other states with high percentages included Illinois, Iowa, and New Hampshire (82-84%), while only 55% of respondents from Nebraska and North Carolina reported that their schools offer introductory CS.

Technology (38%) and Computing (36%) are the most common types of credit given for introductory CS courses, followed by Business (25%), but 69% of Georgia schools give Business credit, 52% of Virginia schools give Math credit, and 58% of Colorado schools give Elective credit for introductory CS courses.

CS educators have a variety of programming languages from which to choose for introductory CS courses. Nationally, Java is easily the most common language (38%), followed by Alice (29%) and Visual Basic (25%). However, Alice is the most prevalent language in Georgia and New Hampshire, and Visual Basic is the most common language in Indiana, Iowa, Kansas, Michigan, New Jersey, Ohio, Pennsylvania, Washington, and Wisconsin. Alice and Visual Basic are equally popular in Connecticut. Scratch is most common in Colorado's introductory CS classes, and JavaScript is the language of choice in Alabama and North Carolina.

Advanced Placement Courses

Nationwide, 36% of responding teachers said that their schools offer AP CS courses. Again, however, individual states vary widely. Maryland's CS teachers reported that 77% of their schools offer AP CS, followed by Florida (54%) and New Jersey (50%). At the other end of the scale were Kansas (9%), Oklahoma (17%), and South Carolina (18%).

Challenges of Teaching CS

Despite the need for students to have the types of skills that are taught in CS courses and a plentiful job market for graduates with CS-related skills, U. S. teachers continue to encounter low enrollments and other challenges in teaching CS. Eighty-one percent of respondents judged that their schools had "students who should be taking or would like to take" a CS course but are not doing so. In Washington, 100% of respondents reported this phenomenon; 95% of respondents in Indiana, Maryland, and North Carolina agreed. Even the states at the lower end of the scale were all above 50%: Kansas (53%), Nebraska (59%), and Alabama (63%).

What other challenges do CS teachers face? Presented with a list of eight specific challenges, teachers rated three of them significantly higher than the rest: they most commonly (39%) classified "Lack of support/interest by school staff" as a "great" challenge; in addition, 35% classified it as a "moderate" challenge. Two other challenges, "Lack of student interest/enrollment" and "Rapidly-changing technology," were classified as "great" challenges by 34% of respondents and as "moderate" challenges by 45%.

Most states' top two challenges were among those three challenges, but there were some interesting exceptions. "Lack of hardware/software resources" was one of the top two challenges for respondents in Alabama, Kansas, Michigan, and Oklahoma, while "Lack of curriculum resources" was critical for teachers in Indiana and Washington. "Difficult subject matter" was listed as the second greatest challenge by Texas respondents.

Read the "State-by-State Survey Results" in the CSTA Advocate blog: *blog.acm.org/csta*. Full survey results, including details of individual states' results (accessible through a clickable map), are available on the CSTA website at *csta.acm.org/Research/sub/CSTAResearch.html*.

Meet the Authors

Debbie Carter
CS Educator, PA
Debbie is a College Board consultant for AP CS and has served on the CSTA Board of Directors. She has been instrumental in developing the CSTA web repository, The Source.

Mark Dorling
Langley Grammar School, UK
Mark established and led a cross phase (primary-to-secondary) transition project called the Digital Schoolhouse. He is a Computing at School Board Member and a lecturer at Brunel University.

Patrice Gans
Newtown, CT
Patrice teaches K–8 technology at the Fraser-Woods School. She also teaches Scratch in summer programs at Naugatuck Valley Community Technical College.

Lynn Langit
Teaching Kids Programming
Lynn teaches, consults, and writes on cloud data, business intelligence, and big data solutions. Her latest book is *Smart Business Intelligence Solutions with SQL Server 2008.*

Christopher Michaud
Paulding County, GA
Christopher teaches K–5 music and CS at Nebo Elementary School. He also coaches the First Lego League Robotics Teams and teaches workshops and camps with the Georgia Institute of Technology.

Mylene Padolina
CSCP, EdLab Group
Mylene currently serves as the Project Manager for the Computer Science Collaboration Project – Engaging Youth with Disabilities.

Kiki Prottsman
University of Oregon
Kiki founded Thinkersmith, a non-profit organization focused on teaching CS as a creative passion. She is enthusiastic about the possibilities for bringing computational thinking to mainstream education.

Aaron VonderHaar
Wintriss Technical Schools
Aaron is a Java consultant and volunteer teacher in San Diego, CA.

Vic Wintriss
Wintriss Technical Schools
Vic an electrical engineer and founder of Wintriss Technical Schools, an after-school program in San Diego, CA.

Classroom Tools

Teaching Kids Programming

Lynn Langit

Teaching Kids Programming (TKP) is a non-profit organization of volunteer programmers who have developed a framework designed specifically for introducing basic programming to children ages 10 and older. The only prerequisite is that the children have basic keyboarding skills. The framework consists of lessons ("recipes") and video-based guidelines for teachers. All materials are free at: www.TeachingKidsProgramming.org.

TKP teaching is based on the "Intentional Method." The TKP Intentional Method means teaching by guiding pairs of children to translate English comments—the intention—into executable code. TKP uses Microsoft SmallBasic and our open source extensions called SmallBasicFun. We have also developed Intentional teaching materials for Java, T-SQL, and Microsoft Kodu (visual programming). Additionally, TKP methods are based on Agile programming techniques (in particular XP). These include the use of core Agile practices in the delivery of TKP courses. TKP practices include the following:

- Pair Programming: both the students and the teachers work in pairs to learn and teach TKP material.
- No Big Upfront: teachers are advised how to guide the student pairs to write and execute their first program within five minutes of the start of each TKP class.
- Test-driven Development: courseware is written so that students can be guided to translate one line of English into one line of code and then to execute the result. This is a type of visual test-driven development.
- Sustainable Pace: careful attention is paid to the pace of the class. Students (within each pair) rotate tasks either on task completion or on a regular time interval (such as five minutes). Pairs are also switched at the end of each lesson.
- Rapid Feedback: in addition to the immediate visual feedback that the students get after they run each line of code in the recipe, we use proctors in the classroom to keep the pairs on pace. In addition to live proctors, the courseware includes a Virtual Proctor, which provides visual feedback (screenshot of each successful execute).
- Craftsmanship: each recipe contains several sections so that students can progressively master concepts, APIs, and tools before being introduced to new information.

TKP coursework is designed to be modular. Each recipe is created to teach up to three core programming concepts. Examples of such concepts are objects, properties, keywords, for-loops, and events. There are up to five sections to each recipe. Each section takes from 15-30 minutes to teach. Sections include:

- Recipe: pairs are guided to create an output starting with Logo-like drawings.
- Recap: instructors re-do (recode) the recipe so the students have time to understand what they have learned.
- Variation: students do refactoring and then change their code.
- Quiz: pairs complete a short quiz in the IDE using the concepts just learned.
- Enrichment: pairs work to further reinforce concepts learned.

The courseware is designed to appeal to both genders and is currently being used in 15 U.S. states and 10 countries.

For more information and to view a teacher training video visit: *www.TeachingKidsProgramming.org*.

Curriculum in Action

Introductory CS in the K-3 Classroom

Patrice Gans

Where is it written that computer science (CS) is a learning experience reserved for middle and high school students? Fortunately, some schools across the country have made CS a priority and are currently nurturing the art and joy of CS in primary school students. The view of students as "creators," rather than just "consumers," is firmly rooted in my curriculum at Fraser-Woods in Newtown, CT.

Over 26 years ago, the book *Mindstorms: Children, Computers and Powerful Ideas,* by MIT Professor Seymour Papert, introduced the educational community to the concept that computers could serve as an environment for "thinking about thinking" and, simultaneously, developed the first educational programming language, Logo, designed specifically for elementary students. Papert envisioned a learning environment where children would program computers to acquire "a sense of mastery over a piece of the most modern and powerful technology." Students would become masters of technology instead of just users.

As the technology boom continues to invade our homes and schools, it is increasingly important for teachers to embrace Papert's vision. I introduce my students to rudimentary CS principles beginning in kindergarten with the application Ladybug Mazes and Ladybug Leaf from the Library of Virtual Manipulatives. The applications are similar to Logo; students create instructions (programs) to navigate a ladybug across the computer screen by selecting the correct icon to move the ladybug in the required direction. Steps are recorded in a plan window so that the students can visualize how each step contributes to the ladybug's overall movement.

After the students became proficient with the movement commands, they begin work with Ladybug Leaf, in which they use manipulatives to create geometric shapes. They start by creating programs to draw squares and rectangles and advance to creating parallelograms, triangles, rhombuses, octagons, and hexagons. Finally, the students write programs to create their own designs.

The Ladybug Leaf and Ladybug Maze applications provide an informal introduction to CS without any of the complications of a computer language or syntax. Initially, my students tackle the challenge of navigating the maze by trial and error, mixed with a fair amount of frustration. However, as we work together to understand the commands, the students are able to efficiently move the ladybug through the maze. At this point their frustration turns to elation. My second- and third-graders have similar experiences with Ladybug Leaf. By the end of the unit, they are developing programs which enable them to create beautiful geometric patterns. Developing code is no longer an enigma. They are proud to call themselves computer scientists.

CS Around the World

10 Year-old Detectives

Mark Dorling

The Digital Schoolhouse (DSH) is a transition project established by Langley Grammar School (LGS) in the United Kingdom to offer predominately Year 6 (10 year old) pupils from local primary schools the opportunity to visit LGS for a day of learning in a dedicated Information and Communications Technology (ICT) and computing environment.

To accommodate the different learning styles, the DSH has developed a variety of audio, visual, and kinesthetic teaching activities. For example, pupils perform a human database role play and use SQL syntax to structure questions. Higher-order questioning is also used to relate concepts to students' "real world" understanding. This enables pupils to better grasp the skills and concepts being taught, and maximizes learning.

This year, the DSH has introduced a new lesson called *Database Detectives,* based on the book *Certain Death* by Tanya Landman. The aim of the lesson is to highlight the role of ICT and computing in the logging and analysis of the data generated in the crime scene investigation lab and to illustrate how detectives use the clues to solve crimes. Before completing the lesson, the teacher is encouraged to read the book (except the last chapter!) and complete a series of numeracy puzzles loosely based on the book. The puzzle answers provide pupils with the clues to identify the murderer.

Pupils use cloud computing technology, such as Google Documents spreadsheets, to collaboratively input data about the suspects from profile cards based on the book. Pupils then perform verification on their neighbour's data entry before downloading the spreadsheet and importing it into Microsoft Access. After importing the data, pupils use the filter tool to solve the murder using the answers from the numeracy challenges and then create a report for the Court based on a query identifying the murderer.

DSH pupils work with a primary-trained teacher, who also has secondary ICT teaching and industry experience, to learn new skills and concepts, with a focus on how they are deployed in secondary education, the world of work, and business. The DSH approach features a creative curriculum based on the proposed changes to the British national curriculum. The creative curriculum links have been developed through collaboration with school subject specialist teachers and the library resource centre. The DSH is open two days per week. The day starts at around 10:00 am and finishes at 2:30 pm. The DSH lesson is split into three sessions lasting approximately 60 to 90 minutes.

The techniques in the Digital Schoolhouse could have future applications in teaching pupils with special educational needs or who have missed out on good ICT teaching in their earlier schooling and could also be extended into post-secondary education.

If you would like to discuss the concept of the DSH in more detail, please contact Mark Dorling at *dsh@lgs.slough.sch*.uk. Further reading about the project is also available from the DSH website (*www.digitalschoolhouse.org.uk*).

In the Spotlight

Collaborative Mini-grant Projects Announced

Obtaining guidance and engaging in career exploration activities can enable youth with disabilities to overcome potential obstacles in transitioning from school to work.

The National Girls Collaborative Project and CSTA are working together to increase accessibility and engagement. According to the Bureau of Labor Statistics, in August 2011, the unemployment rate for individuals with disabilities was 16.1% as compared to 8.8% for people without disabilities. Of the overall number of computer scientists, 16.8% of women and 7.5% of men have disabilities.

In an effort to highlight and support transition efforts occurring throughout the U.S., the Computer Science Collaboration Project (CSCP) works to identify and bring together programs and organizations to increase the participation and engagement of underrepresented youth, including youth with disabilities, in CS opportunities and activities. CSCP places an emphasis in reaching youth with disabilities to impact future employment rates, along with ensuring diversity and inclusion in the CS field.

Project activities are designed to bring organizations together, facilitate connections, encourage and support collaborative projects, and provide targeted professional development. Specific project activities include the funding of mini-grants as an incentive for collaborative projects. In January, CSCP announced that mini-grants were awarded to nine projects engaging youth with disabilities in computer science. The projects are:

- CS Fun Day
- Engaging Youth with Disabilities with Microsoft KODU, Basic Computer Game Development
- High School/High Tech Internships
- North Central Florida High School/High Tech CS Exploration
- Project E.S.T.E.E.M. – Experience Science, Technology, Engineering, Electronics & Math
- SMARTer Board: Girls Resolve Accessibility Issues
- Teaching Science through Robotics
- Project S.T.E.A.D.Y (Science, Technology, Engineering Academy for Disabled Youth)
- Tech Vision Quest

These projects offer youth with disabilities a variety of opportunities including hands-on programming training, introduction to available assistive technologies, exposure to higher education environments, interactions with CS professionals, and paid work assignments. Two of the projects also incorporate the participation of parents as a way to generate more awareness and reinforce messaging. Some projects are one day events while others are ongoing programs lasting several weeks.

CSCP is an NSF funded project that works to build collaborations between CSCP participants from K–12 settings, community-based organizations, higher education, and industry to increase participation and engagement of underrepresented youth in CS; to maximize access to shared resources among representatives from these same groups that are interested in expanding and broadening participation; and to strengthen the capacity of existing and evolving K–12 formal and informal programs in CS by supporting the use of exemplary practices.

More detailed information about the individual projects can be found at *www.cscproject.org*. In supporting and promoting these projects, CSCP hopes to emphasize the power of collaboration and the positive impact of including youth with disabilities in engaging and accessible CS programs.

SHOW ME THE NUMBERS

CSTA Website Growing in Use

	2010	2011	CHANGE
Site Visits:	68,287	74,706	+ 9.4%
Total Page Views:	164,739	178,577	+ 8.4%
Unique Visitors:	47,888	49,938	+ 4.3%
New Visitors:	32,674	32,249	- 1.3%
Repeat Visitors:	15,214	17,689	+16.3%

Source: *CSTA Google Analytics*

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

27th Annual International Technology and Persons with Disabilities Conference
February 27–March 3, 2012, in San Diego, California
www.csun.edu/cod/conference/sessions

SIGCSE 2012
February 29–March 3, 2012, in Raleigh, North Carolina
www.sigcse.org/sigcse2012

Consortium for Computing Sciences in Colleges (CCSC: Southwestern)
March 23–24, 2012, in Stockton, California
www.ccsc.org/southwestern

Consortium for Computing Sciences in Colleges (CCSC: Central Plains)
March 30–31, 2012, in Springfield, Missouri
www.ccsc.org/centralplains

Consortium for Computing Sciences in Colleges (CCSC: Mid-South)
March 30–31, 2012, in Jackson, Tennessee
www.ccsc-ms.org

CSTA Elections Open
April 4, 2012
An election ballot will be sent to all members via e-mail.

Webinar: CSTA Services and Resources: Supporting CS Education Continuum
April 5, 2012, 2:00 PM EST
www.cscproject.org

Consortium for Computing Sciences in Colleges (CCSC: South Central)
April 20–21, 2012, in Canyon, Texas
www.sci.tamucc.edu/ccsc

Consortium for Computing Sciences in Colleges (CCSC: Northeastern)
April 27–28, 2012, in Hamden, Connecticut
www.ccscne.org/2012

CSTA Elections Close
May 2, 2012
An election ballot will be sent to all members via e-mail.

CAS Teacher Conference
June 14–15, 2012, in Birmingham, UK
www.computingatschool.org.uk

International Autonomous Robot Competition (iARoC)
June 23–24, 2012, in San Diego, California
sites.google.com/a/wintrisstech.org/iaroc

Israeli National Conference for CS Teachers
July 3, 2012, in Haifa, Israel
URL unavailable

Innovation and Technology in CS Education (ITiCSE)
July 3–5, 2012, in Haifa, Israel
www.iticse12.org.il

CSTA Annual CS & IT Conference
July 9–10, 2012, in Irvine, California
www.cstaconference.org

Grace Hopper Celebration of Women in Computing
October 3–6, 2012, in Baltimore, Maryland
gracehopper.org/2012

RESOURCES

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

Page 1: CSTA Annual CS & IT Conference *www.cstaconference.org*
Page 2: How to Think Like a Computer Scientist *openbookproject.net/thinkcs/python/english2e*
Page 2: TuxPaint *tuxpaint.org*
Page 2: Nebo Music *www.nebomusic.net*
Page 2: Scratch *scratch.mit.edu*
Page 3: Thinking Myself *games.thinkingmyself.com*
Page 3: Computational Thinking and Women in CS *csta.acm.org/Research/sub/Projects/ResearchFiles/Prottsman_Thesis.pdf*
Page 3: ACM Distinguished Members *www.acm.org/press-room/news-releases/2011/distinguished-2011*
Page 4: Wintriss Technical Schools *www.wintrisstech.org*
Page 4: 2011 National Secondary CS Survey *csta.acm.org/Research/sub/CSTAResearch.html*
Page 6: Teaching Kids Programming *www.teachingkidsprogramming.org*
Page 6: Agile Programming *agilemanifesto.org*
Page 6: National Library of Virtual Manipulatives *nlvm.usu.edu/en*
Page 6: Digital Schoolhouse *www.digitalschoolhouse.org.uk*
Page 7: Computer Science Collaboration Project *www.cscproject.org*
Page 7: Access STEM *www.washington.edu/doit/Stem/promising.html*

Plan for CS & IT 2012

July 9–10
Irvine, CA
www.cstaconference.org