By Bill Heldman

Where's the C in STEM?

U.S. high schools must create robust computer science programs if they want students to be competitive in a global economy that is increasingly driven by technology. Here are a few tips to get them started.

With few exceptions, students interact with technology in one way or another every day. And yet, in most U.S. schools, the term *computer science* (CS) refers only to generic skills classes, such as keyboarding and computer applications. Even most Web programming classes usually teach students only how to use conventional graphical user interface (GUI) tools instead of HTML and CSS, perhaps because they are often led by teachers with no background in CS. Even worse, many U.S. high schools have dropped CS programs altogether.

The end result of this lack of focus on CS is that most U.S. students graduate into an increasingly tech-driven world with little knowledge of how it all works or any chance of contributing to the field. In fact, only 1–2% of students who apply for U.S. colleges major in CS.

Until schools bring CS out of math's backroom and give it a place of prominence as a STEM discipline (changing the acronym to SCTEM, using the same pronunciation, but with a much different meaning), the United States will produce fewer CS majors, and CS work will continue to be outsourced to those countries that do have the *C* correctly placed.

Create a Strong CS Program

I'm a former IT worker with a long background in the industry. Five years ago, I turned to teaching 11th and 12th grade computer science and game programming. The combination of my private and public sector IT experience, coupled with teaching high school, has given me the insight to recognize the roots of the issue and how it can be fixed with enough participation and effort.

CS needs to take its place alongside English language arts, science, and math as a fundamental academic necessity and be addressed as such in state standards, academic content, curricula, and educator licensing requirements.

A big part of the problem is that few educators really understand what computer science is. Administrators may be able to identify the tip of the CS iceberg, but they are ignoring the burgeoning mass beneath. It will take some detailed, hard work for school stakeholders to define the subject within the context of the overall curriculum.

To begin with, CS isn't treated as an academic standard, but it should be. CS needs to take its place alongside English language arts, science, and math as a fundamental academic necessity and be addressed as such in state standards, academic content, curricula, and educator licensing requirements. Teaching CS actually fosters better knowledge of other academic subjects, because when it is taught properly, it drives students to want to know more about math, science, and reading so they can learn more about CS. The subjects feed off of one another.

Once administrators have a clear understanding of the complete context of CS within the curriculum, the next hurdles are creating a standalone CS program, bringing it up to today's technological standards, and supporting it with a continuous improvement process. Because of the pace of technological change, schools must revise CS curriculum on an annual or even semi-annual basis. Administrators must be visionary leaders or have visionaries on staff who recognize and understand those changes so they can create new curricula that parallels current technologies and anticipates emerging technologies.

Why go through all this trouble? Because CS is an important career choice in U.S. society. According to a 2009 news release from the Bureau of Labor Statistics, 5 of the 30 fastest growing jobs in the United States are CS jobs, and they all require bachelor's degrees or higher. Many other surveys as well as recent breakthroughs in graphical and holographic arenas point to an even broader increase in the need for CS.

It is hard work to flesh out a CS program. The good news is that schools don't have to go it entirely alone, and they don't have to build their CS programs in a vacuum.

The school should integrate computer science industry and business leaders into its advisory board and give them a voice in the CS program. This will help teachers keep up with the pace of change and shape class content around industry expectations regarding CS career paths. Students should learn, for example, what game programmers do during a workday. Do they play games? Do they write code all day long, and if so, what kind of code do they write? How do they get the art into the game? How does the game story come about? The industry tie-in is not only relevant, it's key to student engagement and success.

Forging strong articulation agreements with community colleges will also strengthen a school's ability to help students interested in CS understand what they need to know to prepare for college. Community colleges can allow students who have uncompleted prerequisites or who are

ISTOCKPHOTO.COM/TRACK5

intimidated by the college/university setting to gently integrate into the system. This is important because postsecondary teaching environments are night-and-day different from secondary environments, and a student's comfort level translates to better academic performance.

The next step is recruiting and/or training qualified CS teachers. It's simply not good enough for a marketing teacher to offer a Dreamweaver class and teach a little HTML, or for a math teacher with an extra free section to teach AP Computer Science. CS has much deeper tentacles and a far longer reach than that. The teachers have to understand the nuances of the subject at a deep level.

Conventional teachers who do not have a CS background can teach CS as long as they meet the educational and occupational requirements (see the Resources section at the end of this article for the URL to a white paper the Computer Science Teachers Association released on this topic). But it will take extensive and ongoing professional development to bring non-CS teachers up to speed technically and keep them there. Principals can't simply assign the program, toss a how-to book in the door, and expect teachers to be successful. That leads to content that is elementary and unsatisfying, and students won't get anything out of the course.

ISTOCKPHOTO.COM/MBORTOLINO

A Model CS Curriculum

Building on the lessons of the past and the needs of the present and the future, the Computer Science Teachers Association (CSTA) proposes a four-level model curriculum for K–12 computer science that focuses on fundamental concepts. This model includes:

Level I—Foundations of computer science (recommended for grades K–8). Instruction at this level should provide elementary school students with foundational concepts in computer science by integrating basic skills in technology with simple ideas about algorithmic thinking. This can be accomplished by adding short modules to existing science, mathematics, and social studies units or through specific computing courses.

Level II—Computer science in the modern world (recommended for grade 9 or 10). Students at this level should acquire a coherent and broad understanding of the principles, methodologies, and applications of computer science in the modern world. This course focuses on the foundational concepts underlying the science and on knowledge and skills all students require. It can best be offered as a one-year course accessible to all students, whether they are college bound or workplace bound.

Level III—Computer science as analysis and design (recommended for grade 10 or 11). Students who wish to study more computer science may enroll in a one-year elective that would earn a curriculum credit (e.g., math or science). This course continues the study begun at Level II, but it places particular emphasis on the scientific and engineering aspects of computer science—mathematical principles, algorithmic problem-solving and programming, software and hardware design, networks, and social impact. Students will elect this course to explore their interest and aptitude for computer science as a profession.

Level IV—Topics in computer science (recommended for grade 11 or 12). This elective provides depth of study in one particular area of computer science. This may be, for example, an AP Computer Science course that focuses on computer science principles or programming and data structures. Alternatively, this offering may be a projects-based course in multimedia design or a vendor-supplied course that leads to professional certification.

This modified text was excerpted from CSTA's "A Model Curriculum for K–12 Computer Science: Final Report of the ACM K–12 Task Force Curriculum Committee."

Collaborate for Curriculum

Conventional CS can be a boring subject. Students will have a hard time getting interested in subjects such as binary trees and linked lists, classes and objects, TCP/IP, and the OSI model without some colorful, fun hook by which to teach them. Game programming turns out to be one such hook, but there are other ideas for teaching CS that work equally well. High-touch and hands-on labs and assessments linked to real-world project outcomes are the elements that make students want to learn.

It is vital that administrators and teachers understand that CS is more than the Web and more than programming. The CS whole is the sum of the parts, but not just computer parts; it's also made up of math, art, science, and English language arts, particularly when it comes to today's interactive media environments, including social networking sites and games.

For one thing, CS is an intensely graphical world that requires students to foster a healthy knowledge of both 2D and 3D graphics as well as animation environments. It's also a world of story and literature. So it's not enough for a teacher to show a student how to write a program. She must also help the student flesh

It's no longer just about English language arts or history or art or mathematics. In this case, it's about using CS to integrate a story into a program, game, or social networking site by using the tools of the trade in addition to all of the underlying academics that make the story complete.

out program concepts in a graphical environment in a way that is pleasing to the eye, easy to understand and manipulate, and rich in allegory and metaphor. In fact, in many cases, students must use the complete literary construct, including a protagonist, antagonist, plot, conflict, increasing stakes, and so forth.

Schools that integrate CS at the highest levels of their mission statements find that professional learning communities and work teams naturally assemble to discuss amalgamated curriculum and assessments. They want their students to be successful, and they soon find out that CS students will need more than just the black box of programming to find success in the industry. Good school leaders set out to find and assemble teams that are able to play off of one another's strengths to get to the greater goal of a holistic student. It's no longer just about English language arts or history or art or mathematics. In this case, it's about using CS to integrate a story into a program, game, or social networking site by using the tools of the trade in addition to all of the underlying academics that make the story complete.

I have seen this firsthand in my game programming classroom at Warren Tech, a career and technical education high school in Lakewood, Colorado. Students learn how to program in C++, C#, and Adobe's ActionScript 3.0. At the same time, however, we work on the idea of story. They learn that they must put a good story into a console-based (DOS-like) application to create a game that is both fun and playable. I stress that story is not just one element of a game, but the most important element.

Once they have thoroughly polished a story and made sure that it is interesting and consists of the proper constructs, we start to think about how to describe the story in a graphical world. They learn to think about color, 2D versus 3D, speech prompts, visual hints and cues, and Easter eggs. They're still dealing with the same old story, but now they've brought it into a visually satisfying place. And, along the way, I've had an opportunity to help the students learn how to manipulate the tools they need to turn their visual stories into games.

Other subjects also come into play. Suppose, for example, the story talks about a cannon firing. To create an accurate scene, students need to understand the equation of the ballistics of a cannon and scientific ideas, such as the coefficient of friction and velocity versus acceleration. The desire to make the story right drives students to want to know more about the math and physics of the story. A student recreating the board game Battleship needs to understand matrix math. Another student working on a whirlwind wants to know about the Coriolis effect, and so on. The same applies to the historical accuracy of a subject as well as possibly its anthropology, sociology, and ethnology.

Can I personally teach all this stuff? No! I rely on coaches and other teachers to help me round out the teaching model. For example, the mathematics coach might come in and give a half-hour miniclass on logarithms. It is this notion of teamwork and a team attitude toward teaching CS that fosters great advances in student achievement across a plethora of subjects—not just CS.

Schools that seriously integrate CS at the important, vital level it needs to be find that students want it, are good at it, and can be successful at it. On the flip side, if schools fail to introduce full and robust CS courses in high schools, they will do a disservice to their students, colleges and universities, and industry. With that in mind, we need to begin to work together to take back the ground that CS has lost and move forward into new terrain as collaborative educators.

Resources

"A Model Curriculum for K–12 Computer Science: Final Report of the ACM K–12 Task Force Curriculum Committee" (2003) by the Computer Science Teachers Association (CSTA): http://csta.acm.org/Curriculum/sub/CurrFiles/K-12ModelCurr2ndEd.pdf

"Ensuring Exemplary Teaching in an Essential Discipline" by CSTA: http:/csta.acm.org/ComputerScienceTeacherCertification/sub/TeacherCertificationRequi.html

"The 30 Fastest Growing Occupations Covered in the 2008–2009 Occupational Outlook Handbook," U.S. Bureau of Labor Statistics: www.bls.gov/news.release/ooh.t01.htm

Warren Tech High School: www.warrentech.org

Bill Heldman is a former longtime IT professional as well as the author of more than a dozen technical books and numerous articles for technical publications. These days, he teaches teenagers how to make video games, writes curriculum, and collaborates to better the public secondary teaching environment.

Help ISTE revise its Secondary Computer Science Standards!

Give us your feedback through the survey at www.iste.org/compsci-refresh.