Creating AP® CS Principles: Let Many Flowers Bloom

• Marie desJardins •

Computer Science Principles (CSP) is a new curriculum framework that was designed collaboratively by Computer Science (CS) faculty, high school teachers, and the College Board. CSP was initially designed with support and leadership from the National Science Foundation (NSF), with the intent of creating rigorous, engaging, high-quality CS courses that would be available to all high school students in the United States. One of the primary goals of CSP is to broaden participation in high school and college computer science courses [4]. The curriculum is specifically designed to appeal to a more diverse population of students than are traditionally seen in CS courses, particularly female students and underrepresented minorities. Results from the CSP pilot offerings showed that significantly greater percentages of women and underrepresented minorities were attracted to the courses than are seen in traditional CS courses [5], and anecdotally, the instructors of the offerings described here are observing the same increased diversity. The professional development courses that train the teachers focus on active, inquiry-directed pedagogies, and the course has few prerequisites, reducing the barriers to entry compared to AP® CS A, and increasing the potential for attracting a diverse population of students.

The CSP curriculum framework emphasizes computational thinking, creativity, and the impact of computing and computer science. CSP goes beyond traditional programming-focused CS courses by including traditional topics of algorithms and programming as just two of the seven "big ideas" in computer science (the others being creativity, abstraction, data and information, the Internet, and global impact) [2]. Each of the nine courses described in this article provides a solid introduction to algorithm design and programming within this curriculum framework.

The College Board has made CSP an official Advanced Placement® course, with the first AP exam to be offered in May 2017. The CSP creation effort has gone hand-in-hand with NSF's "CS10K" program, whose goal is to train 10,000 high school teachers to teach CSP and other rigorous, academically challenging CS courses.

CSP is defined by a curriculum framework [3] that is structured around six *Computational Thinking Practices* and seven *Big Ideas of Computing*. As discussed later in the article, each Big Idea is expanded through *Essential Questions* (27 in all), *Enduring Understandings* (23), *Learning Objectives* (42), and *Essential Knowledge* (307) statements that articulate specific concepts and knowledge that students completing the course should be able to demonstrate. The AP exam will include two components: a *written examination* that assesses students' mastery of the Essential Knowledge, and electronically submitted student work that constitutes a *through-course assessment* including both col-

laboratively and individually produced work on two *performance tasks*: the Create task and the Explore task.[1]

Within this curriculum framework, there is broad scope for creating courses that are structured around different themes, use different pedagogical methods, present the required information in different ways, and use different programming languages and computational tools. Many different CSP courses have been developed and piloted, and professional development opportunities are available for many of these courses. To date, thousands of teachers have completed professional development in these different "flavors" of CSP.

This paper presents an overview of the CSP curriculum framework, then describes and compares nine of the different CSP courses that have been developed by universities and nonprofit organizations: Beauty and Joy of Computing (BJC), Code.org CS Principles (Code.org), CS50 for AP CSP (CS50), CS Matters in Maryland (CS Matters), CS Principles for High School (CSP4HS), Computing in Secondary Schools (CISS), Mobile CS Principles (Mobile CSP), Project Lead the Way Computer Science and Engineering (PLTW), and Thriving in Our Digital World (TODW). The College Board is developing a formal endorsement model for CSP projects that will include the BJC, Mobile CSP, TODW, Code.org, and PLTW courses. The other projects will likely be endorsed as the formal model expands. Moreover, there are other courses that already are in development or could be created in the future, so the options for delivering the course will continue to grow.

COMPUTER SCIENCE PRINCIPLES

The six Computational Thinking Practices appear in Sidebar 1. These practices are designed to emphasize the type of work that CS professionals engage in, and help students to understand the methods and techniques that are needed to perform the work required for the class and the AP exam. Each of the Learning Objectives (see below) is directly connected with one of the Computational Thinking Practices.

The seven Big Ideas appear in Sidebar 2. Each Big Idea is associated with several Essential Questions; for example, "How do people develop and test computer programs?" (Big Idea 5: Programming) and "How are vastly different kinds of data, physical phenomena, and mathematical concepts represented on a computer?" (Big Idea 2: Abstraction). The concepts and knowledge that students should understand by the end of the course are specified in a hierarchical structure that elaborates on each of the Big Ideas. Specifically, there are three or four *Enduring Understandings* associated with each Big Idea. Each of these Enduring Understandings is associated with one to three *Learning Objectives* that are assessed on the AP CSP performance tasks and written exam. The Learning Objectives each integrate a specific Computational Thinking Practice with particular course content, and specify how students will be asked to demonstrate their knowledge in that area. Finally, each Learning Objective is expanded by a list of one to seventeen *Essential Knowledge* statements that specify facts and content required in order to successfully demonstrate the student's knowledge of the corresponding Learning Objective. Some Learning Objectives and Essential Knowledge include *Exclusion Statements* to limit or constrain the scope of the expectations for student mastery.

Computational Thinking Practices

P1: Connecting Computing. *Students learn to understand the effects of computing on people and on society.*

P2: Creating Computational Artifacts. *Students create computational artifacts that represent creative solutions to problems.*

P3: Abstracting. *Students use abstractions to develop and analyze models and simulations.*

P4: Analyzing Problems and Artifacts. *Students evaluate, improve, and analyze the correctness of proposed solutions to problems.*

P5: Communicating. *Students describe, explain, and justify computational artifacts, behaviors, and results.*

P6: Collaborating. *Students work together in diverse teams to solve problems, produce*

Big Ideas of Computing

BI1: Creativity. *Students design creative solutions to problems, and learn how computation can enable creative expression of ideas.*

BI2: Abstraction. *Students learn how abstraction is used to represent data, to organize knowledge, and to design computational artifacts.*

BI3: Data and Information. *Students learn how data and information are collected, processed, visualized, and understood to solve problems and create knowledge.*

BI4: Algorithms. *Students study, design, analyze, and evaluate algorithms.*

BI5: Programming. *Students write and test programs to solve problems and express creative ideas.*

BI6: The Internet. *Students learn how the Internet works, how it is used to support communication and collaboration, and the importance of cybersecurity solutions for privacy and security.*

BI7: Global Impact. *Students explore how computing enables innovation, augments human interaction, and leads to beneficial and harmful*

[1] The information in this article about the curriculum framework and performance tasks is current as of September 2015. Small changes to the performance tasks may be made as part of the piloting process, but the curriculum framework itself is fixed.

An example of a "slice" through the CSP curriculum hierarchy shows the relationships between the different levels of curriculum design [3].

BI 4: Algorithms

EU 4.2: Algorithms can solve many, but not all, computational problems.

LO 4.2.2: Explain the difference between solvable and unsolvable problems in computer science. [P1]
Exclusion Statement (for LO 4.2.2): Determining whether a given problem is solvable or unsolvable is beyond the scope of this course and the AP Exam.

EK 4.2.2B: Heuristics may be helpful for finding an approximate solution more quickly when exact methods are too slow.
Exclusion Statement (for EK 4.2.2B): Specific heuristic solutions are beyond the scope of this course and the AP Exam.

CSP ASSESSMENT: AP EXAM

The AP CSP Exam will be administered during the two-week AP exam period in early May each year, and will include multiple-choice questions built on the learning objectives and essential knowledge statements of the CSP curriculum framework. The number of multiple-choice questions and the time allowed for completion of the exam are being finalized; current estimates indicate between 60-80 questions given in roughly two hours.

CSP ASSESSMENT: PERFORMANCE TASKS

AP CSP is unusual in that it will include a *through-course assessment* of student work as part of the assessment process. The purpose of the through-course assessment is to assess learning objectives that are difficult to assess with a fixed-response/multiple choice exam and to provide for student-centered work that can both broaden the appeal of a course with a high-stakes exam and provide valid results for generating college credit and/or placement. Each of the two Performance Tasks (PTs) in the portfolio covers multiple big ideas and learning objectives, and each requires the student both to create artifacts and to describe and analyze the creation process. The student work for the PT will be submitted online directly to the College Board for assessment as part of the AP exam.

To ensure that students can complete the PTs, guidelines are provided with each task that ask teachers to provide a specific number of in-class hours for work on the tasks. Teachers are allowed to provide guidance about the submission process and about expectations, including the rubrics that will be used to assess the PTs as part of the AP exam scoring process. Teachers may also offer examples of PT artifacts, and may help monitor progress on the tasks, but cannot select or assign topics to students, nor may they revise, correct, or evaluate students' work on the tasks. Each of the courses described here incorporate practice PTs that offer the students the opportunity to go through the creation process with supervision and feedback, before they independently undertake the actual PTs.

The Explore PT requires students to identify and research a computational innovation, responding to specific prompts by writing about the innovation, its beneficial and harmful effects, the data generated by or associated with the innovation, and other specific learning objectives related to the student-chosen innovation. Students also submit a computational artifact related to the innovation such as an infographic, video, or audio artifact and reflect on the purpose of the artifact and the process of creating it.

The Create PT requires students to design and create a program that demonstrates creativity or solves a problem the student has identified. The Create PT requires both collaborative (in pairs) and individual contributions to the work. In addition to the program itself, students must submit a video that shows how the program works, and must respond to prompts discussing the purpose of the program, its operation, the development process, and how collaboration was used in the creation of the program.

CSP COURSES

The following chart gives a high-level comparison of the nine CSP courses reviewed in this article.[2] The descriptions of the courses in the following sections were provided by the course development teams, and edited for consistency and clarity. The different emphases in the course descriptions reflect the variety of goals of the development teams. Each course has a distinctive flavor or unique characteristic that is summarized in the first column of the table. The courses use a variety of programming languages and computational tools, as indicated in the second column. All of the courses except PLTW have online course materials that are publicly accessible; the stage of development varies and is summarized in the third column. (Most courses distribute their course materials through a Creative Commons licensing agreement.) Most development teams are offering professional development (PD) to interested teachers, either online, in person, or in a hybrid format. Although this article does not review the PD approaches in depth, most of the courses emphasize active learning, student engagement, and inquiry-directed pedagogical methods. The availability of PD opportunities (and whether they are limited to specific partner schools or school systems) is indicated in the last column. Most PD opportunities are free, and some offer a stipend to participating teachers; exceptions are noted in the table.

[2] All information provided about these courses is current to the best of the author's knowledge as of the writing of this publication. All courses are undergoing continual development and the specifics are subject to change.

	Unique Characteristic	Programming Languages and Tools[3]	Availability	Professional Development
BJC (Beauty and Joy of Computing)	Advanced, rigorous programming, mobile apps, Internet APIs	Snap*!*; Python	Eight units available, with continued development planned	Free online support through wiki and Piazza; crowd-funded 6-week PD to be available in 2016
Code.org	Daily lesson plans, App Lab Widgets, Code Studio, discovery-based instruction	JavaScript, Internet Simulator, App Lab persistent data storage	Units 1 and 2 available; additional units to be rolled out in 2015-16	15-month in-person/online PD with teacher stipend in partner districts (matching funds required)
CISS (Computing in Secondary Schools)	Peer instruction pedagogy	Alice, Excel, Internet Simulator	Complete and available through website registration for teachers who have completed PD	Free in-person intensive training over 14 months or free online course on content and teaching methods
CS50	Advanced, rigorous programming with an emphasis on community	Scratch, C, PHP, SQL, JavaScript; Linux	2015-16 rollout in "real time"	Free two-day in-person workshop (with certificate from Harvard) and online community
CS Matters in Maryland	Daily lesson plans emphasizing data creation, analysis, and understanding	Python, Excel; *EarSketch, NetLogo, Bokeh, DataQuest.io*	All six units are available freely on the project website	Free hybrid online/in-person training for Maryland teachers
CSP4HS	Inquiry-based learning; scalable blended PD	Snap*!*	Six PD units available on MOOC portal	Six weeks of free online instruction for all teachers; one week face-to-face; year-long community of practice
Mobile CSP	Mobile app development	App Inventor	Complete and available through website registration	Free six-week online PD for all teachers
PLTW (Project Lead the Way)	Exposure to a wide range of professional tools and programming languages	Scratch/App Inventor, Python, PHP/SQL/HTML/CSS/JavaScript, Linux, NetLogo	Available to PLTW teachers and districts only	Intensive two-week in-person training for teachers in PLTW districts (fee required for districts)
TODW (Thriving in Our Digital World)	Project-based learning and blended delivery using online materials	Scratch, Processing	AP version in development; scheduled for completion in Summer 2016	PD for dual enrollment version offered annually; PD for AP CS with teacher stipend to be offered beginning Summer 2016

[3] Italicized languages and tools are included in optional course modules.

Beauty and Joy of Computing

http://bjc.berkeley.edu

Contacts: Dan Garcia (ddgarcia@berkeley.edu), Tiffany Barnes (tmbarnes@ncsu.edu)

Collaborator: Brian Harvey

Eight Units: Introduction to Computational Thinking; Developing Complex Programs (Fun Programming Project); Lists and Algorithms; Algorithmic Complexity (followed by the CSP Explore PT); Data; The Internet (followed by the CSP Create PT); Trees and other Fractals; Recursive and Higher-Order Functions

Languages/Tools: Snap*!*; optional module on Python

Beauty and Joy of Computing (BJC) uses the Snap*!* visual programming language, includes deep CS ideas (functional programming, recursion, and higher-order functions), and addresses the social implications of computing. The course "meets students where they are, but doesn't leave them there."

A transformative and empowering experience comes when one learns how to program a computer, to translate ideas into code. This course teaches students how to do exactly that, using Snap*!* (based on Scratch), a user-friendly "block" programming language that is purely graphical: programming involves simply dragging blocks around, and building bigger blocks out of smaller blocks. BJC's browser-based environment allows for cloud storage of projects, exploration of internet APIs, and the ability to use it on all mobile devices—one of the earliest examples shows how to make a "whack Alonzo" mobile app in 90 seconds.

BJC also focuses on CSP's "Big Ideas" of computing, including abstraction, design, concurrency, simulations, and the limits of computation. There are daily "computing in the news" discussions, and optimism about technology in general is balanced with a critical stance toward any particular technology. Throughout the course, relevance is emphasized: relevance to the student and to society. The overarching theme is to expose students to the "beauty and joy" of computing: to empower them to create meaningful projects, to see that code itself can be beautiful, and to have fun! BJC is especially focused on bringing computing (through this course) to traditionally underrepresented groups in comput-

ing, i.e., women and ethnic minorities. BJCx, a four-part year-long MOOC on edX, will be launched on Labor Day 2015; as of this writing, more than ten thousand learners have signed up for the first "MOOClet." All of BJC's course materials are freely available on BJC's website, under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International license.

Code.org

http://code.org/educate/csp

Contacts: Baker Franke (baker@code.org), Brook Osborne (brook@code.org), Pat Yongpradit (pat@code.org)

Five Units: Digital Information, The Internet, Programming, Data, Performance Tasks

Languages/Tools: Code Studio, App Lab (JavaScript IDE)

Code.org's CSP course is a year-long high school course with four core units of study and a fifth unit devoted solely to working on Performance Task projects. The curriculum contains lessons and projects that encourage students to construct their own mental models and computational artifacts that address the 7 Big Ideas of CSP. For example, students model the layers of the internet by developing their own sets of protocols to encode and send increasingly sophisticated types of information between computers.

A goal in constructing the course was to provide teachers with a rich set of instructional resources that includes daily lesson plans, computational widgets, videos, rubrics, assessments, a teacher dashboard, and an "answer viewer" for teachers to view student work. The "App Lab" JavaScript programming environment is a powerful learning tool that allows students to create interactive web applications, toggle back and forth between block and text-based modes while programming, easily create databases for persistent data storage, and use a drag-and-drop HTML/CSS editor for quickly constructing and styling the app's user interface.

Successful delivery of this course is not built around a teacher in the role of content expert. Instead, materials emphasize student discovery, and lessons include diverse support materials for teachers to choose from, based on what is appropriate for their students. For those teachers looking for more guidance with the curriculum, Code.org offers a 15-month long professional development program focused on the instructional practices and concepts that make up this course. Code.org's course materials are publicly available and distributed under a Creative Commons license.

Computing in Secondary Schools (CISS)

http://www.csedohio.org/

Contact: Nigamanth Sridhar (n.sridhar1@csuohio.edu)

Collaborators: Debbie Jackson, Santosh Misra, Karla Hamlen, Beth Simon

Four Units: Programming using Alice; developing complete programs with special emphasis on creative aspects of program design; exploring the internet and impact of computing; understanding the nature and use of data and information.

Languages/Tools: Alice, Excel, Internet Simulator

The CISS CS Principles curriculum is based on the course developed by Dr. Beth Simon at University of California at San Diego and the ComPASS project. CISS uses the Alice programming language to address topics of Abstraction, Algorithms, Creativity, and Programming. Students explore a variety of programming concepts and constructs in Alice, and complete the Create performance task by creating an original program in Alice. The course includes eleven Alice modules that provide broad coverage of programming topics, including sequencing, conditionals, looping, objects and methods. Students spend a significant amount of time on planning and storyboarding their programs, with a substantial emphasis on creative aspects of program design.

The course uses Microsoft Excel or Google spreadsheets as a platform to teach students how to understand and comprehend Data and Information. The course includes two modules on spreadsheets and data. Students use publicly available datasets from the internet that they analyze and use to produce infographics as part of the Explore performance task.

The course also uses a variety of "unplugged" activities that address the Global Impact of computing. Course materials are available through a Blackboard site that requires a free registration process, after completing free professional development modules.

In 2014-15, five teachers taught the CS Principles course across Ohio. In Summer 2015, two of the teachers from the first cohort returned as master teachers to work with the CISS team to train 18 more teachers, who will all be teaching CS Principles in their schools. At the end of the 2015-16 school year, all curriculum materials will be made available for free, in time for the AP course to be launched in 2016-17.

CS50 for AP CSP

https://cs50.harvard.edu/ap

Contact: Doug Lloyd (ap@cs50.harvard.edu)

32-36 Weeks: Topics include abstraction, algorithms, data structures, encapsulation, internet technologies, resource management, security, software engineering, and real-world impacts thereof.

Languages/Tools: Scratch, C, PHP, SQL, JavaScript, Linux

CS50 is Harvard University's introduction to the intellectual enterprises of computer science and the art of programming for students at all experience levels. "CS50 AP" is a free adaptation for high schools that satisfies the new AP CS Principles curriculum framework by translating CS50's thirteen-week college curriculum to a pace that is more suitable for high school audiences.

CS50 AP offers a rigorous experience, but is also designed to be accessible to all students, whether or not they have previous programming experience. Largely using programming as a vehicle, students are introduced to the "big ideas" of the CS Principles curriculum, along the way learning valuable problem-solving techniques. These techniques will help them not only when studying for the AP Exam, but also for their careers beyond, whether or not they continue on a professional computer science track. At the end of the course, students are challenged to create a project entirely of their own design, providing an opportunity to creatively explore what most interests them.

Students and teachers alike can leverage CS50's existing online communities to support their learning—in particular, teachers can

join a network of fellow CS50 AP educators. Course materials will be available online as they are rolled out during the 2015-16 school year. All materials are licensed as OpenCourseWare, and educators are encouraged to use, remix, and share the course's resources.

CS Matters in Maryland

http://csmatters.org

Contacts: Marie desJardins (mariedj@cs.umbc.edu), Jan Plane (jplane@cs.umbc.edu), Joe Greenawalt (jgreenawalt@ccboe.com)

Collaborators: Dianne O'Grady-Cunniff, Christina Morris, Jennifer Smith

Six Units: Your Virtual World, Developing Programs, Information and the Internet, Data Acquisition, Data Manipulation, Data Visualization

Languages/Tools: Python, Excel; optional modules for EarSketch, NetLogo, Bokeh, Dataquest.io

The CS Matters in Maryland CS Principles AP course incorporates a focus on active, inquiry-based learning. CS Matters includes detailed daily lesson plans and offers numerous extensions and adaptations to meet the needs of diverse learners. The curriculum was created in summer 2014 and is undergoing continued development and improvement. Fourteen master teachers worked with the CS Matters team to develop the course, using a collaborative curriculum development process within a novel web-based collaborative curriculum building software system that the CS Matters team is planning to make publicly available. The curriculum development process included ongoing team review of all course materials, focusing on alignment with the CSP learning objectives and essential knowledge, Common Core math and science standards, inclusive instructional practices, and active learning pedagogies.

The structure of the course is designed to meet all of the CSP learning objectives, to prepare the students for the two CSP Performance Tasks and written exam, and to spread out the work on these tasks over the course of the year. The overarching theme of the course is *data*: the nature and variety of data on the internet; algorithmic methods for processing and managing data; and ways in which data can be analyzed, visualized, and interpreted to increase human understanding and solve challenging real-world problems. Programming concepts are taught using Python. Units 1-3 are publicly available on the project website. Units 4-6 are in beta distribution: they are available by request, and will be made publicly available by Fall 2015.

CSP4HS

https://csp-cs4hs.appspot.com

Contact: Jeff Gray (gray@cs.ua.edu)

Collaborators: A+ College Ready/NMSI (Mary Boehm and Carol Crawford), CS4Alabama teachers

Six Units: Paving the Way; The Power of Bits and Bytes; Programming is a Snap!; Abstraction and Algorithms; The Internet Makes the World All Flat; The Big Deal about Data

Languages/Tools: Snap!

The CSP4HS course was developed in 2011 as a College Board Pilot CSP course at the University of Alabama, with a special focus on preparing pre-service Secondary Math Education students. The course was then adapted for CS4Alabama, an NSF CE21 project that is training 50 teachers across Alabama in multi-year professional development across several teacher cohorts. CSP4HS is driven by Teacher Leaders, who mentor and develop shared curriculum resources. With support from Google's CS4HS program, the training has evolved to a large-scale online professional development effort (training over 1,000 teachers from 47 states and 12 countries in 2015), freely available to anyone interested in teaching CSP.

CSP4HS provides a "gentle introduction" to CSP, with a focus on helping teachers and students understand the CSP Curriculum Framework and Performance Tasks. The target audience is high school teachers who may be considering the course as a first offering at their school, and who may not enter the training with the content knowledge needed to teach the course. Pedagogical suggestions for teaching the content are also offered throughout the course, with a particular emphasis on cooperative learning structures to help infuse diversity and improve student engagement.

The CSP4HS training materials include: (1) training modules for the CSP Curriculum Framework and Performance Tasks; (2) over 120 videos for teachers and students; (3) lesson plans developed by Alabama Teacher Leaders; (4) lesson slides; (5) quizzes and exams; (6) a pacing guide and syllabus; and (7) a Piazza-based community of practice for teachers participating in the course. Six states are using CSP4HS for their online content and providing their own face-to-face training based on the CSP4HS topics. Course materials are available to teachers through a web portal after a free registration process.

Mobile CSP

http://mobile-csp.org

Contact: Ralph Morelli (ralph.morelli@trincoll.edu)

Collaborators: Jen Rosato, Chery Takkunen, Chinma Uche

Eight Units: Preview and Setup (Pre Course); Mobile computers and Mobile Apps; Graphics and Drawing; Animation, Simulation, and Modeling; Algorithms and Procedural Abstraction; Lists, Databases, Data and Information; The Internet; AP CSP Exam Prep

Languages/Tools: App Inventor

Mobile CSP is organized around the Big Ideas of AP CSP. The course includes more than 30 detailed tutorials covering programming topics in App Inventor, and more than 30 lessons on computer science topics such as algorithms, binary numbers, and computer security. Each lesson includes interactive exercises with immediate feedback, ranging from simple quiz questions to live-coding exercises. Live-coding exercises in App Inventor reinforce understanding of basic programming concepts and improve problem solving skills. Readings from *Blown to Bits* [1] ask students to reflect on some of the major societal issues that characterize 21st century computing, such as privacy, security, and social networking.

Mobile CSP is a project-based course. Students complete two collaborative programming projects and an individual research and writing project on the impact of a recent, computing innovation

that appeals to the student. These projects conform to the Explore and Create Performance Tasks. Course materials are freely available following a free website registration process.

Project Lead The Way Computer Science and Software Engineering

https://www.pltw.org/our-programs/pltw-computer-science/pltw-computer-science-curriculum

Contact: PLTW School Support Team (schoolsupport@pltwo.org)

Four Units: Algorithms, Graphics, and Graphical User Interfaces; The Internet; Raining Reigning Data; and Intelligent Behavior

Languages/Tools: Scratch/App Inventor, Python, PHP/SQL/HTML/CSS/JavaScript, Linux, NetLogo

Project Lead The Way's CSE course focuses on computational thinking and real-world applications of computer science. Students learn Python as a primary tool and engage in projects and problems including app development, visualization of data, cybersecurity, and simulation. CSE also incorporates a wide range of additional tools and platforms to expose students to different modes of computational thinking. Using a fee-based partnership model with school systems, PLTW offers comprehensive professional development for teachers, day-to-day lesson plans and resources, and school and technical support. Unlike the other eight courses covered in this article, PLTW has a strict licensing agreement, and course materials are only available to teachers in partner districts who have paid for and completed the PLTW training.

Thriving in Our Digital World

http://www.cs.utexas.edu/~engage/

Contacts: Calvin Lin (lin@cs.utexas.edu), Bradley Beth (bbeth@cs.utexas.edu)

Six Units: Impact, Programming, Representation, Digital Manipulation, Big Data, Artificial Intelligence

Languages/Tools: Scratch, Processing

Thriving in Our Digital World (TODW) is rooted in the inquiry-based pedagogies of project-based learning. The inquiry-based, student-centered approach to instruction engages diverse student populations with computer science content that is both rigorous and relevant to their lives. Each of the six modular units combines an authentic problem or scenario, structured team collaboration, student-centered activities, and engaging multimedia and narratives. Teaching and learning are supported through a variety of scaffolds including rubrics, group contracts, and intermittent checkpoints for formative feedback.

The dual enrollment nature of TODW has helped to attract a more diverse student population than is typically found in AP courses, and has led to a close partnership between UT faculty and high school teachers. The dual enrollment version of the course has been offered since 2012, with accompanying professional development for teachers. TODW is now working with the UTeach Institute—responsible for helping 43 universities start UTeach STEM teacher preparation programs—to develop an AP version of the course and to train hundreds of teachers to offer the new course. TODW's course materials are under development and will be made available on the project website.

The widespread adoption and availability of CSP as an AP high school course has the potential to reach many more students than have previously had the opportunity and inclination to take CS at the high school level.

CONCLUSIONS

AP CSP represents an exciting new opportunity for high school students to be exposed to a broad view of computational thinking concepts and practices, and to apply these ideas in hands-on, collaborative, active learning environments. The widespread adoption and availability of CSP as an AP high school course has the potential to reach many more students than have previously had the opportunity and inclination to take CS at the high school level. The diversity of CSP courses surveyed here highlights the flexibility of the CSP curriculum framework to be met by different courses that meet the needs of diverse student populations. **Ir**

Acknowledgements

This work was supported by NSF award #1339265. Many thanks to Owen Astrachan, Bradley Beth, Bennett Brown, Baker Franke, Joe Greenawalt, Jeff Gray, Calvin Lin, Doug Lloyd, Ralph Morelli, and Nigamanth Sridhar for their inputs and comments on this paper.

References

[1] Abelson, Hal, Ken Ledeen, and Harry Lewis, *Blown to Bits: Your Life, Liberty, and Happiness After the Digital Explosion*. Addison-Wesley Professional, 2008; http://www.bitsbook.com/wp-content/uploads/2008/12/B2B_3.pdf. Accessed 2015 August 15.
[2] Astrachan, Owen, "The CS Principles Project." *ACM Inroads* 3, 2 (2012).
[3] College Board, AP Computer Science Principles Curriculum Framework 2016-2017; https://secure-media.collegeboard.org/digitalServices/pdf/ap/ap-computer-science-principles-curriculum-framework.pdf. Accessed 2015 August 15.
[4] Cuny, Jan, "Transforming High School Computing: A Call to Action." *ACM Inroads* 3, 2 (2012).
[5] Snyder, Lawrence, et al. "The First Five Computer Science Principles Pilots: Summary and Comparisons." *ACM Inroads* 3, 2 (2012).

MARIE DESJARDINS
Department of Computer Science and Electrical Engineering
University of Maryland, Baltimore County
1000 Hilltop Circle
Baltimore Maryland 21250 USA
mariedj@cs.umbc.edu

DOI: 10.1145/2835852