ASEE 2022 ANNUAL CONFERENCE
Excellence Through Diversity
MINNEAPOLIS, MINNESOTA, JUNE 26TH-29TH, 2022 ASEE
Paper ID #38420

Engaging Female High School Students in the Frontiers of Computing

Gordon Stein

Gordon Stein is currently a PhD student at Vanderbilt University. Previously, he served as a Senior Lecturer at Lawrence Technological University, helping to improve introductory Computer Science courses and integrate emerging technologies into the curriculum. At Vanderbilt's Institute for Software Integrated Systems, he has worked on projects combining accessible, block-based programming with robots and mixed reality platforms for educational use. Gordon also has experience bringing educational robotics into K-12 classrooms and summer programs. He is very excited to help make STEM education more fun and engaging for students worldwide.

Isabella Gransbury

Devin Jean

Lauren Alvarez

Marnie Hill

Marnie Hill (Program Manager, former HS CS Teacher), has her M.Ed in Technology Education with 8 years of teaching experience and 8 years experience in leading teacher professional development. She has several years of experience in developing and maintaining effective relationships with teacher professional development programs, and site coordinators, recruiting teachers and coordinating and training facilitators for PD workshops.

Veronica M Catete

Shuchi Grover

Tiffany Barnes (Distinguished Professor)

Brian Broll

Brian Broll is a Research Scientist at the Institute for Software Integrated Systems at Vanderbilt University. He holds a Ph.D. from Vanderbilt University in Computer Science and a B.Sc. from Buena Vista University, majoring in mathematics education. His research interests include computer science education and model-integrated computing.

Akos Ledeczi

Akos Ledeczi is a professor of computer science at Vanderbilt University. His research interests include wireless sensor networks, cyber physical systems and computer science education.

Engaging Female High School Students in the Frontiers of Computing

abstract

Creating pathways that stimulate high school learners' interest in advanced topics with the goal of building a diverse, gender-balanced, future-ready workforce is crucial. To this end, we present the curriculum of a new, high school computer science course under development called Computer Science Frontiers (CSF). Building on the foundations set by the AP Computer Science Principles course, we seek to dramatically expand access, especially for high school girls, to the most exciting and emerging frontiers of computing, such as distributed computation, the internet of things (IoT), cybersecurity, and machine learning. The modular, open-access, hands-on curriculum provides an engaging introduction to these advanced topics in high school because currently they are accessible only to CS majors in college. It also focuses on other 21st century skills required to productively leverage computational methods and tools in virtually every profession. To address the dire gender disparity in computing, the curriculum was designed to engage female students by focusing on real world application domains, such as climate change and health, by including social applications and by emphasizing collaboration and teamwork.

Our paper describes the design of curricular modules on Distributed Computing, IoT/Cybersecurity, and AI/Machine Learning. All project-based activities are designed to be collaborative, situated in contexts that are engaging to high school students, and often involve real-world world data. We piloted these modules in teacher PD workshops with 8 teachers from North Carolina, Tennessee, Massachusetts, Pennsylvania, and New York who then facilitated virtual summer camps with high school students in 2020 and 2021. Findings from teacher PD workshops as well as student camps indicate high levels of engagement in and enthusiasm for the curricular activities and topics. Post-intervention surveys suggest that these experiences generate student interest exploring these ideas further and connections to areas of interest to students.

introduction & motivation

Computing has finally come of age as advances in the field drive the transformation of work, commerce and everyday life. Distributed and cloud computing, artificial intelligence and machine learning, autonomous systems, the internet of things and cybersecurity are some of the new frontiers of computing that are fundamentally transforming how and where people work,

collaborate, communicate, shop, eat, bank, travel, consume news and entertainment and, quite simply, live [1]. Yet, learning experiences that engage and expose high school students, and especially those from historically marginalized groups, to these advanced computing concepts and practices in interdisciplinary contexts are not available to teens. The AP Computer Science Principles (CSP) high school course introduces students to computer science and programming through a novice-friendly curriculum that appeals to learners from diverse backgrounds. What should motivated students study after successful completion of AP CSP? The AP CSA class is centered on syntax-heavy learning of Java programming and it has traditionally not attracted students from underrepresented groups.

Building on the foundation that AP CSP provides, our new course 'CSFrontiers' seeks to dramatically expand access to the most interesting and exciting frontiers of computing and the types of collaboration and 21st century skills (such as data literacy) required to productively leverage computational methods and tools in virtually every profession. To address the dire gender disparity in computing, the curriculum is designed to engage female students by focusing on real world application domains and issues relatable to high school students (and especially girls) and affording exploration of topics such as social justice, healthcare, and climate change, and emphasize project-based learning, collaboration and teamwork. The novice-friendly NetsBlox programming environment accesses publicly available data sources to allow exploration of these cutting-edge computing ideas. This paper describes our research and development efforts to develop a new modular, project-based course comprised of four 9-week modules—"CSFrontiers" (CSF)—that seeks to dramatically expand access to the most interesting and exciting frontiers of computing and 21st century skills (such as collaboration and data literacy). The curriculum is designed with the goals of fostering student interest, and in turn, the development of a diverse workforce with skills in high demand today.

theoretical framing & curriculum pedagogy

Drawing on a theoretical framework centered on project-based learning and expansive framing, this project places collaboration, creativity and social relevance at the forefront of the curriculum design, and integrates strategies for successful recruitment of girls. The curriculum also provides students with female industry professional role-models to foster girls' self-perception, social encouragement, and belonging within STEM/ICT careers. Targeted content and role models are important, but not sufficient [2]. Our curricular activities connect to students lives, cutting-edge industry practices, and issues local to students' communities. Students engage in several projects to demonstrate the applications of their learned skills to real-world problems. Our mechanisms to measure the impact of the curriculum are aligned with the PBL design philosophy (described below).

Project-based Learning. The tenets of project-based learning (PBL) align well with our vision for a curriculum that engages all students and especially girls. PBL provides a meaningful project-oriented context to engage learners in conceptual ideas through hands-on knowledge building [3], [4]. PBL approaches draw on learning theory, especially cognitive apprenticeship, in which learners apply and learn disciplinary ideas and skills to investigate and solve meaningful problems [5]. Project-based science, for example, has been shown to engage learners in authentic

disciplinary learning as well as practices such as argumentation, explanation, scientific modeling, and engineering design in science classrooms [6], [7]. We hypothesize that, as in science, rich problems that situate complex computing ideas and skills can build connections between students' knowledge of advanced CS concepts and their understanding of everyday computing experiences. Our approach to PBL will focus on rigorous treatment of learning goals, while also supporting pedagogical approaches to make CS learning more meaningful to learners through authentic projects that can result in heightened motivation and interest [8].

Expansive Framing. Engle [9] proposed the idea of expansive framing. It is a strategy for engagement and preparation for future learning (PFL) that explicitly fosters an expectation that students will continue to use what they learn. Framing curricular content as having the potential for transforming students every- day experiences initiates a series of processes of cognitive encoding of the learning that eventually lead to greater motivation and transfer. Grover [10] successfully built on this work [9] to address middle school students' perceptions of computing and motivate transfer and preparation for future learning in an introductory CS course [11].

This design philosophy will be supported by the use of the NetsBlox [12]. programming environment (described below) that also affords features for an easy introduction to the advanced ideas in computing covered in CSFrontiers [13], [14].

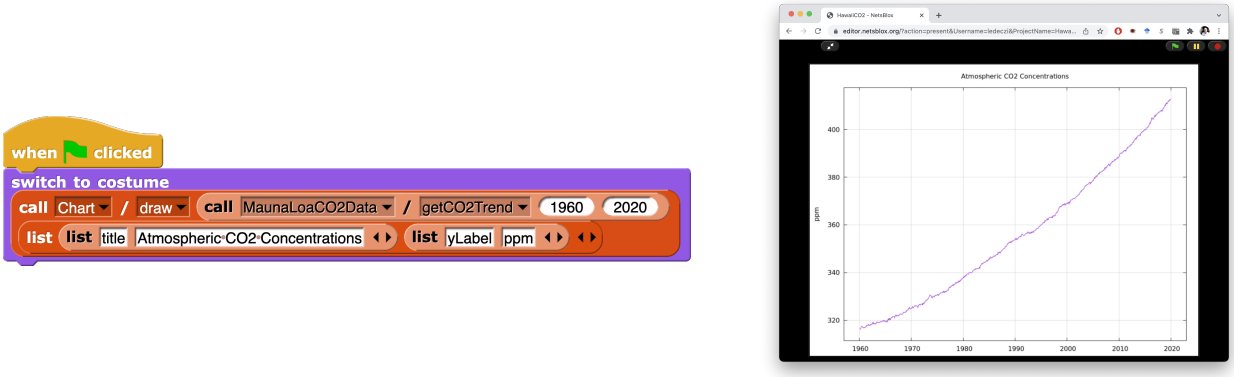
programming tools to support advanced ideas in computing

Most block-based programming environments provide only limited support to access the network. However, in order to support distributed computation and other modern computing technologies, it is important to provide flexible access to the internet. To this end, NetsBlox extends Snap! with two intuitive abstractions that allow students to create truly distributed applications [13], [14].

The first abstraction, *Remote Procedure Calls (RPC)* provide access to a set of selected online data sources and web services. RPCs allow users to invoke functions running remotely on the NetsBlox server and provide results as return values. Related RPCs are grouped into *Services*. Examples are Google Maps, Weather, Earthquakes, the Movie Database, and many others. Additional services that run directly on the NetsBlox server, and do not require third party support, include a Gnuplot-based chart service and a hierarchical key-value store called Cloud Variables. Importantly, RPCs use a ***single block*** called "call" which is self documenting. It has two pull-down menus, one for the service and one for the RPC. When a service is selected, the second menu reconfigures itself to show the RPCs available within the selected service. When an RPC is selected, slots for the required input arguments appear along with their names. Context sensitive help explains what the given RPC does and what the required input arguments are.

To illustrate how easy it is to start with NetsBlox and RPCs, consider the following 7-block program that display atmospheric carbon dioxide concentrations as measured by the NOAAA on Mauna Loa (Figure 1). The call to the *getCO2Trend* RPC of the *MaunaLoaCO2Data* Service returns a 2-column matrix: each row is a pair of time and measured CO2 in parts per million (ppm). This can be directly passed to the *draw* RPC of the *Chart* Service that returns an image

with the plot. The optional second argument to the draw RPC is a list of property-value pairs that modify the appearance of the plot. The returned image can be passed to the *switch to costume* block that changes the appearance of the sprite as shown.



(a) (b)

Figure 1: Displaying CO2 concentrations from Hawaii by NOAA. Code (a) and resulting plot (b).

The "call" block is a powerful abstraction. Using a single generic block that configures itself according to context removes the cognitive load of learning a new set of blocks for every service. It also eliminates palettes full of new and unfamiliar blocks that would require searching for just the right one. Furthermore, the call block returns built-in data types (numbers, text, lists, multi-dimensional arrays or images, etc.). Users do not need to de-serialize the data, parse text, or a process a JSON data structure to extract useful information from results, unlike with HTTP calls available in other tools.

The second distributed computing abstraction built into NetsBlox is message passing. Messages in NetsBlox are very similar to events in Scratch and Snap*!*. However, unlike events, messages can carry data and they do not have to stay within the project; they can travel to any other NetsBlox project running anywhere on the internet at the time of sending. Messages have types, defined by a name and the data the message is to carry (i.e., an ordered set of input slot names). Message type definition is done similarly to how one defines a custom block header in Snap*!*.

Figure 2: Simple texting app

Only two blocks are needed for message passing: one for sending and one for receiving. Selecting a message type in the "send" block pull down menu reconfigures it to show the corresponding input slots with their names provided. Similarly, selecting a message type in the "when I receive" receiver hat block shows the same fields as variables, just like a custom block

definition does. Figure 2 shows a simple texting app. If two or more users run the same app, they can send and receive messages to and from each other.

NetsBlox supports both synchronous and asynchronous collaboration. It allows users to issue and accept invitations to collaborate on a project. Collaborators can then work on the same project simultaneously. Concurrent editing operations show up on everyone's screen. The server resolves conflicting changes by approving the first one, and rejecting subsequent ones. However, since the typical latency is under 100 milliseconds, this rarely happens. Students can also work asynchronously. This is similar to how popular collaborative editing tools such as Google Docs work. However, there is a conceptual difference between static documents and continuously executing block-based code. The latter has a state: variable values and the appearance of the sprites and the stage. Since each user's computer executes the code independently, the program state would be hard, if not impossible, to synchronize. So NetsBlox only keeps the program itself in sync across collaborators. The scripts will be the same, but the stage and the variable values will typically be different across users.

Collaboration support enables pair programming, team projects, remote tutoring, and remote collaboration. The latter two have been especially important with online learning during the pandemic.

module descriptions

The CSF curriculum comprises four 9-week modules–Distributed Computing (DC), Internet of Things (IoT) & Cybersecurity, Artificial Intelligence & Machine Learning, (AI/ML), and Software Engineering (SE). This section described the first three that have been designed and piloted. The design of the last module, Software Engineering, is currently in progress.

distributed computing

The first module has multiple objectives. First, it needs to introduce the NetsBlox programming environment that is the primary tool used across the entire curriculum. At the same time, it serves as a refresher for the most important programming concepts students learned in previous classes. Finally, it covers the two distributed computing abstractions NetsBlox brings to the table: RPCs and message passing. These three objectives are tackled simultaneously through a sequence of increasing more complex hands-on projects.

The first project is a weather app with a fully interactive map background using the RPCs of the *Google Maps* service. Wherever the user clicks, the current weather conditions are displayed. The project reacquaints students with variables, if-statements and custom blocks (i.e., functions) and introduces RPCs such as *getMap* that returns an image with the map of the specified area, *getXFromLongitude* that converts longitude to screen coordinates, or *getTemperature* RPC of the *Weather* Service that returns the temperature in Fahrenheit at the desired location. The instruction is interactive: after the teacher introduces a concept and demonstrates its usage, the students need to add similar functionally on their own. For example, after demonstrating how to implement zooming in, students have to work on zooming out by themselves. Similarly, after showing how

to pan East, students have to program panning West, North and South on their own.

The next lesson is on lists using the *MovieDB* service. The program asks for a movie title and displays photos of the three leading cast members. This lesson is followed by accessing and visualizing climate change related data from the NOAA (see Figure 3). The focus in the first part of the module is on using online services to access STEAM data sources and services to create engaging projects.

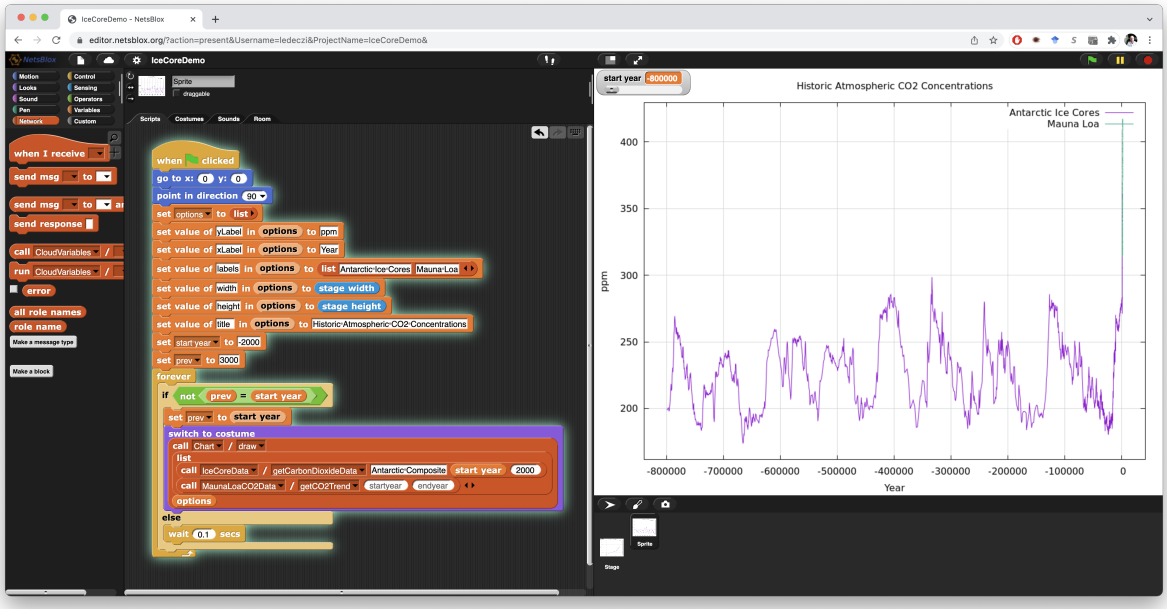


Figure 3: Plotting atmospheric CO2 concentrations for the past 800,000 years

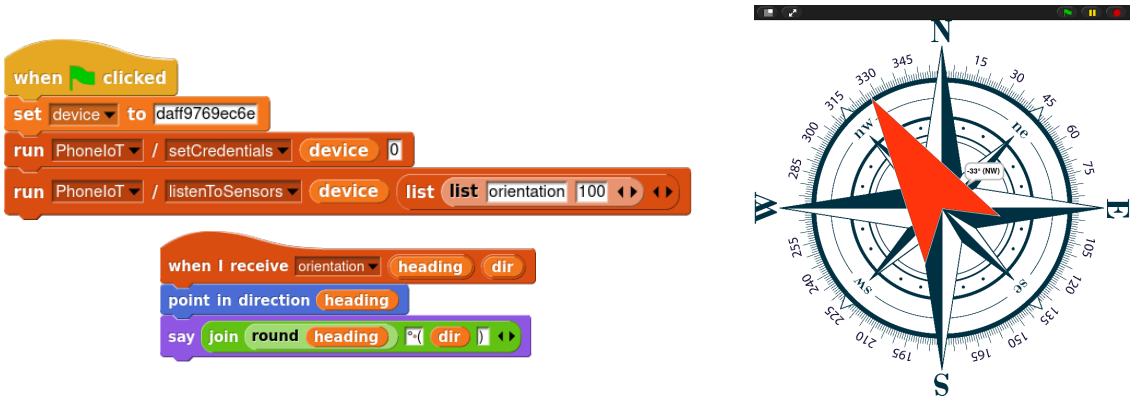
The second half of the module focuses on social and collaborative projects utilizing message passing. After introducing how to send and receive messages by creating a distributed “Hello World” example, the first project is an animation of a running dog as it seamlessly jumps from one computer screen to another. This unit is followed by creating a shared whiteboard as two students can write on each others’ stages by sending a list of pen coordinates back and forth. Other lessons include a chatroom where all classmates can post messages for everybody else and a mesh networking simulation. The final lesson is a simple distributed turn-based multi-player game such as Tic Tac Toe, 21 pebbles or Ghost. These lessons teach how to create various message types and design simple communication protocols.

The last week of the module is set aside for an individual or team based project of the students own choosing. Learners can pick one or more of the many services of NetsBlox to access data sources or service that were not covered and make a project utilizing it. They can even bring their own data. NetsBlox has a dedicated service that takes a CSV file and turns it into a user-defined service automatically. Or students can focus on message passing instead and implement their favorite distributed game.

internet of things (iot) & cybersecurity

As more and more devices are connected to the internet, IoT has become a pervasive trend in manufacturing, transportation, energy generation, smart buildings, and homes, among other areas. The NetsBlox environment supports accessing a number of open source sensors, allowing students to process and visualize sensor data. To introduce students to IoT, NetsBlox supports accessing ThingSpeak [15], which is a broad collection of primarily user-owned devices with a common access protocol. This allows students to search for certain types of devices—for instance, weather sensors—and download their recent data for processing and visualization in NetsBlox through tools such as the Chart service, which lets students easily generate and display graphs of their data.

To bring IoT content closer to students, NetsBlox also supports accessing the sensors and graphical displays of students' own smartphones through a mobile app called PhoneIoT [16]. By using their own hardware, this allows students to physically manipulate the devices they connect to and see live changes from direct interaction. For instance, one of the first projects involving PhoneIoT has students use live sensor data to turn their device into an accelerometer-based controller for manipulating sprites on the NetsBlox stage. That is, by tilting their phone every which way, they can control how the sprite moves on their computer screen. Another easy beginner project is a compass app using the phone's orientation sensor (see Figure 4). It is important to note that the student's program runs in the browser on their computer and not on the phone. The program interacts with the phone using the RPCs of the PhoneIoT Service and messages the phone sends to the user's program.



(a) (b)

Figure 4: PhoneIoT code (a) in NetsBlox for a simple compass app on the stage (b).

PhoneIoT provides a simple introduction to two common IoT access paradigms: polling and streaming. RPC return values are used for polling and message passing for streaming. For example, the compass app above uses the *listenToSensors* RPC to request orientation data from the phone every 100 milliseconds. In turn, the phone starts to send 10 "orientation" messages per second as shown in the figure.

Later, students are introduced to PhoneIoT's graphical components, which enables creating custom widgets on the display using yet another set of RPCs and receiving asynchronous

event-based user interactions via message passing. For example, students use this to implement a simple fitness tracker app that accesses the location sensor of the device and various RPCs in the Google Maps Service to plot the user's location on a map and update the phone's display with a mirror of the NetsBlox stage and other information such as the estimated total distance covered. Again, all the logic is implemented by the student's program in NetsBlox that in turn, interacts with the phone.

Thus far, students have been introduced to IoT primarily in terms of sensors, be they "proper" sensors like an accelerometer or GPS, or sensors representing user interaction. However, IoT also includes actuators, which allow programs to interact with the physical world. To this end, students are introduced to RoboScape, a NetsBlox service for writing programs that control wireless mobile robots [17]. Because the cost of hardware can be a barrier to some schools, NetsBlox also supports a virtual robotics environment called RoboScape Online [18], which is compatible with the existing RoboScape service. With RoboScape, students can access their physical or virtual robot's onboard sensors to implement manual behaviors like remote control driving with the keyboard or through PhoneIoT widgets, or autonomous behaviors such as lidar-based collision prevention. As opposed to many existing educational robotics platforms, RoboScape robots have a fixed program which accepts text-based commands over the network; this is used to introduce students to several cybersecurity topics. Initially, all communication between students' NetsBlox programs and the wireless robots is unencrypted, allowing students to explore how to eavesdrop on messages and even how to falsify data and take control of others' robots. To address this issue, students are introduced to a few simple encryption techniques supported by the robots (e.g., Caesar ciphers), as well as how to potentially crack these ciphers and circumvent other students' security measures. Later, students are introduced to Denial of Service (DoS) attacks, as well as rate limiting as a means to counter it. Proceeding in this fashion, students are gradually introduced to a variety of cybersecurity concepts including secure key exchange, replay attacks, and others. At several points along the way, students are tasked with designing their own suite of safeguard measures and programming their robots to manually or automatically complete some objective while other students attempt to sabotage them with their own suite of attacks.

artificial intelligence & machine learning

Artificial intelligence and machine learning have become increasingly important in daily life, as more systems are relying on these powerful algorithms and data to make decisions. Guided by the emergent AI4K12 framework [19], we have developed some new activities and adapted activities from the AI-4-All curriculum [20] for the AI/ML module. The AI-4-All Open Learning curriculum evolved from a 2015 summer camp called SAILORS designed in 2015 to meet three goals: increase interest in AI, contextualize AI through social impact, and address barriers for girls in CS [21].

In the first AI/ML unit, students use the already-familiar block-based NetsBlox programming environment to explore data features from several demo Twitter accounts to see how that affects the presence of Twitter bots in clusters. Once familiar with classification, students build simple classifiers using real datasets of hundreds of Twitter accounts. In the second unit, students complete an interactive map activity where they learn the inner workings of breadth first search

and find the shortest paths between major cities in the U.S.. In the third unit, students learn to use the Genius API of natural language processing techniques for sentiment analysis on two different forms of media (tweets and song lyrics). In the fourth unit, students take a deeper look at machine learning techniques. Students work with neural networks and imitation learning concepts to modify training sets to see how that influences the effectiveness of algorithms used. In the fifth unit, students make modifications to ML algorithm parameters, and eventually design their own machine learning apps. As part of each unit, students explore ethics in ML and biases perpetuated from pre-existing datasets on which algorithms are trained, and discuss how these issues arise in articles and stories from the news.

This module introduces AI and ML through classifiers, search, and sentiment analysis activities, and also emphasizes social applications and ethics of AI. In the NetsBlox Twitter activity, students use information such as number of followers, tweets, and retweets, to develop their own classification rules. For the Sentiment Analysis activity, students use Python with the Genius API library to compare the sentiment of song lyrics and song titles and develop hypotheses for trends based on their results. The topics of AI & Environment and AI & Criminal Justice from the AI4All Open Learning curriculum [20] help students connect the AI and ML they are learning to real world applications.

pilot studies: teacher pd & student camps

summer 2020, distributed computing pd

Table 1: CSF:DC PD phases and key activities

Week 1 (Teacher Training)	**a.** Intro to NetsBlox; **b.** Intro to Distributed Computing; **c.** Coding & training on RPCs and message passing broken down into a series of projects; **d.** Key pedagogies for CS teaching
Week 2 (Student Camp)	Teachers in groups of 2 or 3 work with students on a 1-week "camp" involving RPCs and message passing projects
Week 3 (Co-Design)	Teachers work in 2 groups to create/co-design 7 lesson plans for the CSF:DC Module

5 Female and 2 male high school CS teachers from North Carolina(4), Tennessee (2), and Massachusetts (1) were invited to participate in the Summer program. The teachers represented a diverse racial mix– the 5 female teachers were African-American (2), White(2), and Indian-American (1), and both male teachers were White. The 3-week summer PD was designed in 3 key phases each mapping to one week (see Table 1). For week 1, which was the most intense time of teacher training alongside the researchers, we planned (a) An introduction to the broader CSF project and research; (b) An introduction to Distributed Computing (CSF:DC) and its key concepts; (c) A curricular sequence of NetsBlox projects to bring RPCs and message passing to life; (d) Sessions on pedagogy (growth mindset, pair programming, PBL, real-world connections, culturally relevant pedagogy, student identity and intersectionality).

Each day of PD ended with teachers using the last half an hour to reflect, debrief, and respond to these survey questions: *What went well today? What could be improved as we move forward?*

What was one success you had today? Is there anything else you'd like to share?. At the end of the 3 weeks, we administered a summative feedback survey with mostly Likert scale items to get feedback on various elements of the overall PD and learning experience. In addition, we also asked teachers how likely they are to use the learning of CSF:DC with NetsBlox in their classes in the coming year and how they planned to use the materials. We also asked teachers a few open-ended questions about what they liked or what could be improved for the future.

teacher feedback

Based on the student projects in the summer camp in week 2 and co-designed lesson plans in week 3, we believe the summer program was a resounding success. Not only did the teachers learn from the experience as evidenced by both outcomes, they brought their teaching experience to make key value-additions to the lesson plans in terms of identifying a need to articulate prior knowledge and the spiral curriculum nature of the DC projects. Furthermore, they also incorporated ideas to round out the units with non-programming activities such as student research projects on understanding networking more generally. Such activities tied well to the pedagogical aspects of a project-based curriculum that connected ideas to the real world.

Table 2: Mean scores in summative survey (out of 5)

Field	**Mean (SD)**
I can use this training to positively impact the achievement of my students.	4.67 (0.47)
The content of the professional development is relevant to my professional responsibilities.	4.33 (0.75)
The facilitators helped me understand how to implement my learning.	4.67 (0.47)
This professional development will extend my knowledge, skills, and performances.	4.67 (0.47)
This professional development was tailored to meet my needs as a learner.	4.83 (0.37)
The agenda and plan were appropriate for the activities.	4.83 (0.37)
The agenda and plan were conducive to learning.	4.83 (0.37)
New practices were modeled and thoroughly explained.	4.67 (0.47)
Sufficient time was provided for guided practice and tasks.	4.50 (0.5)
The facilitators were knowledgeable and helpful.	5.0 (0.0)
The facilitators were well prepared.	5.0 (0.0)
The instructional techniques used facilitated my learning.	4.67 (0.47)
The materials used were accessible and enhanced my learning.	4.67 (0.47)
The PD activities were carefully planned and well organized.	4.83 (0.37)
The PD goals and objectives were clearly specified.	4.67 (0.47)
The PD included a variety of learning activities relevant to the topic.	4.67 (0.47)
Time was used efficiently and effectively.	4.67 (0.47)

Teacher feedback on the summer experience was overwhelmingly positive. The following are mean scores on aspects of PD from the summative survey.

summer 2021, teacher pd & student camp

In summer 2021, six of the seven 2020 teachers and two experienced high school CS teachers were recruited to participate in PD. The PD lasted 5 days, with 4 hours planned for synchronous work, and the expectation that teachers would spend the remaining 4 hours of each day planning as needed. We started with a 1.5 day introduction to refresh NetsBlox, programming, and problem based learning pedagogies. On the second day, the 8 teachers were divided into 4 pairs, two pairs that would learn and teach AI/ML, and two pairs that would learn and teach IoT. We then provided two parallel 3.5-day professional development workshops for the separate AI/ML and IoT groups. For each topic, we provided teachers with student-facing materials and brief teacher guides. Teachers worked through the curricula in pairs, and instead of always working each activity together synchronously as in summer 2020, we instead provided videos of solutions for each activity each day, so that teachers could use their asynchronous time to work on each activity at their own pace.

For the two weeks following the teacher PD, teachers worked in pairs to facilitate a summer camp on the topic they'd just learned. The AI/ML and IoT summer camps were facilitated through NC State University's Engineering Place, and held from 9am-12pm Eastern online each day. Over the course of 10 days, students explored each of the two modules in 30 hours, culminating in final presentations of a topic of their choice on the last day. Each afternoon after a 1-hour break, teachers met with the project PIs for 45 minutes to debrief by discussing how the camp went, reading student feedback, asking questions, and planning for the next day as needed.

teacher feedback

Each day after each camp session, we debriefed with the teachers to get their perspectives on trying out the materials with participants. In all cases, the facilitators reported a positive view of the curriculum and disclosed that participants were engaged in the activities. Conversations with the PD facilitators expressed some types of activities were more successful than others. Successful activities included the NetsBlox activities, the AI4ALL AI Bytes units, and the final project presentations. Teachers stated that the NetsBlox activities (Twitter Bot Classification and Sentiment Analysis) allowed participants to focus on the AI concepts being discussed without participants being concerned with optimal coding concepts. As the participants began the Python activities, a facilitator commented "Students seemed to enjoy working in Python. My students even said that it was fun." We also received positive comments about the AI4ALL AI Bytes presentations and the topics discussed. One teacher said, "[I] really really appreciated the absolutely excellent slide shows!", and another said, "Campers had some good insights about some of the issues with facial recognition AI, as well as some of the potential beneficial uses". This statement also expresses that participants were not only learning about the subject of AI and ML, but engaging in the subject matter. Finally, the culmination of the camps was the paired/group final projects and presentations. The day before the final presentation showcase (where parents were able to attend with their children to view the final project demos) a facilitator remarked "All groups made excellent progress on their projects today. More than half the groups were able to finish and think about extensions to their projects. The other groups are confident that they'll be able to finish in time tomorrow." Despite the differences in programming

backgrounds, all participants completed a final project and presented a slide presentation on them with a demo of their Python code.

student feedback

To gain perspectives from our female participants, we look at open-ended questions from the post-survey on participants' experience in the camp and their interest in a career in computer science. The female participant data showed the overall response to the question "What other feedback, comments, or suggestions do you have after your experience with the [CSF] Camp?" was very positive. One female participant commented on recommending the program, "I learned a lot and would recommend it to others." Others included, "I like how they were trying to make the students very energetic and more involving with the lesson" and "I had fun at this camp." Four female participants commented on the impact of the facilitators on their experience which is another factor for female participant engagement [22]. Additional feedback illustrates how facilitation is a valuable aspect of curriculum development and the importance of the facilitators' impact on the participants. A female participant commented, "I really enjoyed the camp and the support my teachers provided", "The information was very thorough and I really liked the step by step coding instructions. The Genius API activity was also something I've never seen before (in a good way).", "I liked how knowledgeable all the counselors were and the amount of effort that was put into the presentations and activity planning was evident".

Our results indicated that while the participants were challenged, they were not discouraged by the difficulty or the new concepts they were learning as represented in the following comment "I really enjoyed this camp! A few of the assignments towards the end of the week were confusing but once we worked through them it felt good that we could understand it."

In the final part of the survey, respondents were asked "Do you think you could be a computer scientist?" 6 female participants answered yes and 4 answered maybe (due to being interested in other STEM areas, which was specified in the survey). The results of this question, coupled with Likert scale survey questions indicate that the participants left more confident in knowing what being a computer scientist is like and whether they could visualize themselves as one. This was likely impacted by the balance of teacher leaders, invited speakers, and graduate student camp facilitators being largely female, providing them with many female role models.

next steps & conclusion

In the future we will be developing, piloting, and implementing our 4th and final module on Software Engineering and Games. This module carries 3 main themes across its learning activities including software engineering processes (prototyping, testing, teamwork), human computer interaction (HCI) (usability), and ethics (accessibility, security, etc). To situate this this lesson in a meaningful and engaging context, we explore both casual, collaborative games as well as serious games, or games with a purpose. These lessons will build on each other and students will end the module with a final capstone project that incorporates many of the learning objectives they've experienced earlier in the course. This unit also capitalizes on videos and guest speakers to bring in female role models from industry to highlight their innovative and collaborative work in the

field. We will train the CS Frontiers teachers on these materials in summer 2022, before they pilot in virtual summer camps prior to implementing the units in their academic courses.

acknowledgements

This material is based upon work supported by the National Science Foundation under Grants 1949472, 1949492 and 1949488. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

References

[1] S. Grover and R. Pea, "Computational thinking: A competency whose time has come," *Computer Science Education: Perspectives on teaching and learning in school. London: Bloomsbury Academic*, pp. 19–37, 2018.

[2] J. P. Cohoon, J. M. Cohoon, and L. G. Cintron, "Teaching teachers to teach diverse students in computer science," in *2016 IEEE Frontiers in Education Conference (FIE)*. IEEE, 2016, pp. 1–2.

[3] R. A. Duschl, H. A. Schweingruber, and A. W. Shouse, *Taking science to school: Learning and teaching science in grades K-8*. Washington, DC: National Academies Press, 2007, vol. 49.

[4] R. Lehrer and L. Schauble, *Cultivating model-based reasoning in science education.* Cambridge University Press, 2006.

[5] A. Collins, J. S. Brown, S. Newman, and L. Resnick, *Knowing, learning, and instruction: Essays in honor of Robert Glaser. Cognitive apprenticeship: Teaching the craft of reading, writing, and mathematics*. Psychology Press, 1989.

[6] J. L. Kolodner, P. J. Camp, D. Crismond, B. Fasse, J. Gray, J. Holbrook, S. Puntambekar, and M. Ryan, "Problem-based learning meets case-based reasoning in the middle-school science classroom: Putting learning by design (tm) into practice," *The journal of the learning sciences*, vol. 12, no. 4, pp. 495–547, 2003.

[7] K. L. McNeill, D. J. Lizotte, J. Krajcik, and R. W. Marx, "Supporting students' construction of scientific explanations by fading scaffolds in instructional materials," *The journal of the Learning Sciences*, vol. 15, no. 2, pp. 153–191, 2006.

[8] D. A. Fields, Y. Kafai, T. Nakajima, J. Goode, and J. Margolis, "Putting making into high school computer science classrooms: Promoting equity in teaching and learning with electronic textiles in exploring computer science," *Equity & Excellence in Education*, vol. 51, no. 1, pp. 21–35, 2018.

[9] R. A. Engle, D. P. Lam, X. S. Meyer, and S. E. Nix, "How does expansive framing promote transfer? several proposed explanations and a research agenda for investigating them," *Educational Psychologist*, vol. 47, no. 3, pp. 215–231, 2012.

[10] S. Grover, S. Cooper, and R. Pea, "Assessing computational learning in K-12," in *Proceedings of the 2014 conference on Innovation & technology in computer science education*. ACM, 2014, pp. 57–62.

[11] S. Grover, "Teaching and assessing for transfer from block-to-text programming in middle school computer science," in *Transfer of Learning*. Springer, 2021, pp. 251–276.

[12] "NetsBlox website," https://netsblox.org, 2021, cited 2021 December 1.

[13] B. Broll, Á. Lédeczi, P. Volgyesi, J. Sallai, M. Maroti, A. Carrillo, S. L. Weeden-Wright, C. Vanags, J. D. Swartz, and M. Lu, "A visual programming environment for learning distributed programming," in *Proceedings of the 2017 ACM SIGCSE technical symposium on computer science education*, 2017, pp. 81–86.

[14] B. Broll, Á. Lédeczi, G. Stein, D. Jean, C. Brady, S. Grover, V. Catete, and T. Barnes, "Removing the walls around visual educational programming environments," in *2021 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*. IEEE Computer Society, 2021, pp. 1–9.

[15] M. A. G. Maureira, D. Oldenhof, and L. Teernstra, "Thingspeak–an api and web service for the internet of things," *World Wide Web*, 2011.

[16] D. Jean, B. Broll, G. Stein, and Á. Lédeczi, "Your phone as a sensor: Making iot accessible for novice programmers," in *2021 IEEE Frontiers in Education Conference (FIE)*. IEEE, 2021, pp. 1–5.

[17] Á. Lédeczi, M. Maróti, H. Zare, B. Yett, N. Hutchins, B. Broll, P. Völgyesi, M. B. Smith, T. Darrah, M. Metelko *et al.*, "Teaching cybersecurity with networked robots," in *Proceedings of the 50th ACM Technical Symposium on Computer Science Education*, 2019, pp. 885–891.

[18] G. Stein and Á. Lédeczi, "Enabling collaborative distance robotics education for novice programmers," in *2021 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*. IEEE, 2021, pp. 1–5.

[19] D. Touretzky, C. Gardner-McCune, F. Martin, and D. Seehorn, "Envisioning ai for k-12: What should every child know about ai?" *Proceedings of the AAAI conference on artificial intelligence*, vol. 33, no. 01, pp. 9795–9799, 2019.

[20] "AI-4-All website," https://ai-4-all.org/, 2022, cited 2022 February 7.

[21] M. E. Vachovsky, G. Wu, S. Chaturapruek, O. Russakovsky, R. Sommer, and L. Fei-Fei, "Toward more gender diversity in cs through an artificial intelligence summer program for high school girls," in *Proceedings of the 47th ACM Technical Symposium on Computing Science Education*, 2016, pp. 303–308.

[22] A. Fisher and J. Margolis, "Unlocking the clubhouse: the carnegie mellon experience," *ACM SIGCSE Bulletin*, vol. 34, no. 2, pp. 79–83, 2002.