A Cultural Computing Curriculum

James Davis
Rensselaer Polytechnic Institute
Troy, New York
DavisJ20@rpi.edu

Michael Lachney
Michigan State University
East Lansing, Michigan
LachneyM@msu.edu

Zoe Zatz
Rensselaer Polytechnic Institute
Troy, New York
zatzz@rpi.edu

William Babbitt
Rensselaer Polytechnic Institute
Troy, New York
babbw2@rpi.edu

Ron Eglash
University of Michigan
Ann Arbor, Michigan
eglash@umich.edu

ABSTRACT

Broadening the participation of underrepresented students in computer science fields requires careful design and implementation of culturally responsive curricula and technologies. Culturally Situated Design Tools (CSDTs) address this by engaging students in historic, cultural, and meaningful design projects based on community practices. To date, CSDT research has only been conducted in short interventions outside of CS classrooms. This paper reports on the first semester-long introductory CS course based on CSDTs, which was piloted with 51 high school students during the 2017-2018 school year. The goal of this study was to examine if a culturally responsive computing curriculum could teach computer science principles and improve student engagement. Pre-post tests, field notes, weekly teacher meetings, formative assessments, and teacher and student interviews were analyzed to assess successes and failures during implementation. The results indicate students learned the conceptual material in 6 months rather than in the 9 months previously required by the teacher. Students were also able to apply these concepts afterward when programming in Python, implying knowledge transfer. However, student opinions about culture and computing didnâĂŹt improve, and student engagement was below initial expectations. Thus we explore some of the many challenges: keeping a fully integrated cultural curriculum while satisfying CS standards, maintaining student engagement, and building student agency and self-regulation. We end with a brief description for how we intend to address some of these challenges in the second iteration of this program, scheduled for fall 2018. After which a study is planned to compare this curriculum to others.

CCS CONCEPTS

• **Applied computing → Education**; *Interactive learning environments*; *Collaborative learning*;

KEYWORDS

Pilot, Mixed-method, Culture, Curriculum, High School, Introductory, Computer Science, Design, Culturally Situated Design Tools

ACM Reference Format:
James Davis, Michael Lachney, Zoe Zatz, William Babbitt, and Ron Eglash. 2019. A Cultural Computing Curriculum. In *Proceedings of the 50th ACM Technical Symposium on Computer Science Education (SIGCSE '19), February 27-March 2, 2019, Minneapolis, MN, USA.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3287324.3287439

SIGCSE '19, February 27-March 2, 2019, Minneapolis, MN, USA

ACM ISBN 978-1-4503-5890-3/19/02.
https://doi.org/10.1145/3287324.3287439

1 INTRODUCTION

In July of 2017, a high school computer science (CS) teacher lamented that when students in his introductory CS course used Interactive Python they were neither learning nor engaged, and worse yet did not experience the personal and creative satisfaction that programming offers. This teacher works in an urban high school in Upstate New York which included over 50% students underrepresented in CS. Counter to findings by Margolis et. al. [15] that reveal students of color are often faced with shallow CS offerings, this teacher provided students with a rigorous CS curriculum that would lay a foundation to take more advanced CS courses in high school and college. Yet, while the LOGO turtle had been reported by Seymour Papert [16] and colleagues as a motivating factor for young people's engagement with programming decades earlier, its implementation in Interactive Python proved, like the course in general, to be of little interest to his students. Indeed, in this school CS had become yet another course with content decontextualized from students' lives and passions.

To address these challenges, a team of social scientists, technologists, and this teacher collaborated to develop an equally rigorous curriculum called Cultural Computing. It used a suite of culturally responsive educational technologies called Culturally Situated Design Tools (CSDTs) to contextualize CS within African American and Native American cultural practices. The idea behind CSDTs is that computational thinking is already present in many cultural practices; and thus students can benefit by using tools that allow them to "translate" between indigenous or vernacular knowledge in cultural context, and equivalent concepts in the classroom. With CSDTs, students are asked to examine the history and personal significance of cultural practices like cornrow braiding, quilting, and science fiction, then to computationally duplicate existing designs before creating artifacts of their own. This contextualization is a critical component of culturally responsive education, in which

community, national, and political identities are connected in support of learning [13]. Unlike prior research with CSDTs in small scale workshops of a week or two, this paper reports on a semester long curriculum intended to be implemented in a CS course over several months.

2 BACKGROUND

CS undergraduate degree attainment is notoriously low in terms of gender (17.1% women) and ethnic (1.1% black) diversity [19]. This cannot be entirely accounted for by access. Black students have access to 23% fewer classes than white students, and 17% fewer computers [20], but there is a 1400% underrepresentation in CS degree attainment [19, 20]. Thus there may be other social, economic, and/or psychological barriers for why students do not enter CS in high school and college. To make matters worse, students in CS classes face many of the same challenges as in other classes: uninteresting lecture styles, regimented curricula, and poor scaffolding for future courses. There have been many broad criticisms of CS education in the U.S. as a result.

Some claim that classes are too didactic and do not support the development of formal thinking [16], which may hurt all students, but especially those resistant to authority. Others suggest that student disengagement may be from stereotype threat [6], where students in CS courses may feel, explicit or implicit, as though they cannot succeed because of false, though widespread, societal stereotypes about their race or gender [18]. There are additional claims that students simply do not engage because CS is not an obviously visible and representative part of their culture or community [4], or that teachers can have a genuine interest in their students but there is a cultural gap that can result in miscommunication and frustration [8].

Constructionism is one learning theory that has sought to use coding and design to overcome these issues via low-control exploration in an open and inviting environment [16]. For example, Scratch, a constructionist visual programming environment, aims to address the didactic nature of education by giving students more freedom to design their own programs. Another learning theory tackling these issues is culturally responsive teaching, which grounds teachers' classroom content in the unique cultural competencies of their students [10, 13]. Thus reducing or eliminating false impressions of cultural or racial limitations and supporting better communication with students of different backgrounds. Culturally responsive teaching treats community expertise and cultural competencies as important academic assets and culturally rich sources for student engagement. One offshoot of culturally responsive teaching is culturally responsive computing, which highlights the computational qualities of such community assets and expertise [12, 17].

CSDTs is a suite of culturally responsive computing applications that leverages constructionist design and implementation strategies. Like Scratch, CSDTs have students work within a visual programming environment to design and reverse engineer media but, unlike Scratch, grounds that media in computationally rich cultural artifacts (e.g. Native American quilts and African American braiding techniques). As part of this process, students conduct research on cultural and community practices in small groups to frame the historical, social, and/or political significance of the design they are working with. In the past, CSDTs have been shown to improve student engagement and knowledge acquisition in short term implementations [1, 7]. There has been no research on semester or year long curricula using CSDTs, and the research supporting their design and development has taken place outside of CS courses. As a result, major challenges remain for creating CSDTs and associated curriculum for CS classrooms.

3 CURRICULUM DEVELOPMENT

Based on Interactive Python, which this curriculum was designed to replace, students needed to explore critical CS concepts including: Parameters, Loops, Variables, Conditionals, and Functions while practicing program writing, reading, and debugging. This method of introducing various concepts is less constructionist than desired, but needed to maintain instructor support. To ground material and engage students up front, this material was put first. Then concepts were introduced in the order above because they were considered easiest to hardest. Students were then introduced to programming with tutorials which had previously been shown to be effective<cite pending paper>. Each new unit was introduced by the instructor, who provided a guided practice session, in which they demonstrated the relevant CS concepts while students simultaneously performed the same actions on their own computers. This too came from the instructors personal experience. Then students were provided with cultural artifacts to be created. These artifacts were based on prior work and experiences with various cultural groups. It was hypothesized that some of the scripts would be too challenging for students to create on their own, and so they were provided with parts that were already completed. Students were then asked to create their own designs using their new knowledge.

4 METHODS

A CSDT based curriculum called Cultural Computing was administered in four high school classes, attended by a total 51 students. To assess the curriculum a pre-post test was created. Of the 51 students, we were able to match 33 pre and post tests. Based on the pre-post test demographics, the class identified as, 48% white, 24% black or African American, 6% Hispanic, 3% Asian, and 18% multiracial. Furthermore 24 identified as male, 9 identified as female.

The pre-post assessment consisted of seven main parts, each designed by the research team in collaboration with two external evaluators. Section one of the assessment contained demographic information (e.g. gender and ethnicity). Section two contained four items designed to gather students' perception related to computing. The third section measured students' perceptions related to culture and section four contained students' perceptions as they related to the relationships between computing and culture. The above mentioned close-ended items used a series of six-point "Likert-type" [5] scale: 1=Strongly Disagree, 2=Disagree, 3=Slightly Disagree, 4=Slightly Agree, 5=Agree, 6=Strongly Agree. The fifth section of the assessment used eight vocabulary items with a word bank. The sixth section contained two items asking students to write pseudo code and a multiple choice code question. Seventh and finally, there was a section with three questions on geometric transformations.

While this was a CS course, we sought to measure students' understanding of transformational geometry because each of the CSDTs that students used to create cultural designs drew on cultural traditions in which the geometric transformations (i.e. rotation, translation, reflection, and dilation) are embedded. Thus the seventh section on this topic helped examine knowledge, transfer, and possible fringe mathematical benefits. This test, and the results, were reviewed and scored by external evaluators.

Additionally, researchers attended most sessions to take field notes and scheduled weekly meetings with the teacher to discuss students' progress and provide constructive criticism that was noted for future iterations of the curriculum. Field notes and teacher meetings focused on class activities and student engagement, with particular attention paid to students' difficulties interacting with the software and online platform. Formative assessments in which students were provided with an end goal, starting costumes, and a list of steps / outcomes and points for each new topic before being asked to complete the design without peer or instructor support. Ethnographic interviews were conducted with three students, which focused on their feelings and future plans as they related to CS and culture. Finally, interviews with the teacher both at the end of the curriculum and after moving to Interactive Python were conducted to assess his perspective on successes and failures of the Cultural Computing curriculum in preparing students to move to a text-based programming environment.

An initial pre-test was administered before students set up accounts and profiles on the CSDT online learning community (csdt .rpi.edu). On this site they learned about the political, social, and historical relevance of the cultural designs they were to spend the semester working with (African American cornrow braiding; Native American, African American, Appalachian quilting traditions; and feminist, indigenous, black and speculative science fiction). After their exploration of these topics they shared what they learned with the rest of the class in the form of Google presentations. They then took an introductory programming tutorial. This was followed by a sequence of units on "parameters," "loops," "variables," "conditionals," and "functions." As part of each unit, a guided practice session given by the teacher, in which he demonstrated the relevant CS concepts while students simultaneously performed the same actions on their own computers. After the guided practice, students were given a programming challenge in which they were to study and manipulate code to finish an incomplete design from one of the cultural areas. After the conditionals unit, but before functions, students were then asked to work in pairs to create their own designs and then physically fabricate them. After the completion of the Cultural Computing curriculum students took a post test before proceeding to an Interactive Python curriculum.

5 RESULTS

No significant differences were found when examining the mean differences within subscores for student perceptions on computing, culture or their relationship on the pre-post assessment (see Table 1). This is consistent with observational data. When students were informally surveyed at the end of the semester, less than one-third of them intended to continue with CS. It is also consistent with our interviews in which two of three students were not able to recognize the relationship between CS and history, politics, community, or culture originally intended to be integrated. One student didn't remember covering the material at all.

Table 1: Within groups attitude differences [11]

Subscore (n=33)	Pre	Post
Perceptions Related to Computing	16.09	16.03
Perceptions Related to Culture	12.91	12.79
Perceptions related to Computing and Culture	12.79	11.09

When looking at individual items in each of these three constructs some notable changes standout beyond interest in pursuing CS (see Table 2). This includes students confidence in explaining CS concepts to middle school students. In addition, while results show a negative change in students' interest in learning about their own culture, students' were more interested in cultural and artistic practices from other cultures. Consistent with these two positive changes, there was a slight positive change in students' confidence that they can use cultural art and designs to teach computer programming to people in their community. Students were also less likely to say CS supported cultural understanding, or cultural practices supported understanding CS. This further confirms the aforementioned observational and interview data. This may have been due to the fact that the teacher front loaded all of the political, historical, and social aspects of the CSDTs and exclusively focused on the CS aspects of the tools for the remainder of the course.

A paired-samples t test was calculated to compare the mean pretest score to the mean post test score for vocabulary: Conditionals, Recursion, Loops, Functions, Variable, Heritage Algorithm, Script and List. The mean on the pretest was 2.67 (sd=1.493) and the mean on the posttest was 4.03 (sd=1.649). A significant increase from pretest to posttest was found ($t(32)=-3.697$, $p<.05$ (see Table 3). These results are well couched in fact that the units of the Cultural Computing curriculum were largely designed around many of these terms (e.g. students spent a whole unit on loops, another on conditionals, and so on). What is more, the teacher taught and reinforced these CS concepts in the course through both lectures and guided practices.

Finally, a paired samples t test was calculated to compared the mean pretest score to the mean post test score for open ended questions around math and programming . The mean on the pretest was 3.70 (sd=2.568), and the mean on the post test was 7.55 (sd=3.193). A significant increase from pretest to post test was found ($t(32)=7.608$, $p<.001$ (Table 4).

This too was consistent with our formative assessments and observations. Some students during the first unit, parameters, challenges required instructor intervention to move images on the screen. In the final unit nearly all students were able to make their own designs that included the concepts from every unit. While the first three formative assessment showed several students completely copying from old projects, and many struggling, the final formative assessment showed most students were able to nearly complete a challenging design independently.

Table 2: Treatment Group Responses to Survey Items [11]

Statement	Pre % agree	Post % agree	Change
I look forward to learning more about how computer programs work.	64	58	-6
I feel confident explaining what computer programming is to someone in middle school.	24	42	+18
I am interested in learning about computer programming in college or in my future work.	52	49	-3
I am interested in learning computer programming on my own time, outside of school.	27	33	+6
I am interested in learning about different cultures and their artistic practices.	39	46	+16
I am interested in learning more about my own cultural heritage.	49	35	-14
Including knowledge of different cultures in school improves learning	46	48	+2
I think that knowing about cultural designs and art will make someone a better computer programmer	42	30	-12
I think that understanding computer programming will better help me understand cultural designs and art such as quilting, braiding and science fiction.	49	30	-19
I can use cultural design and art to teach computer programming to people in my community.	18	30	+12

Table 3: Mean Scores from Pre and Post on Vocabulary Questions [11]

Treatment Group	N	Mean	SD	Change pre to post
Pre	33	2.67	1.493	
Post	33	4.03	1.649	+1.46

Table 4: Paired samples statistics on open-ended math and computing questions [11]

Treatment Group	N	Mean	SD	Change pre to post
Pre	33	3.70	2.568	
Post	33	7.55	3.193	+3.60

However, the rate of this learning was extremely varied. The variables unit, intended for only 2 weeks, lasted 6, with the instructor repeating material and challenges constantly until satisfied students understood it. However, of conditionals and if-else loops the teacher said "I think they [students] immediately grasped it." The teacher frequently stressed that the curricula performed much better than Interactive Python, with students learning all the same material in 6 rather than 9 months. Furthermore, when students moved to text based programming, the teacher stated that they were able to transfer the relevant concepts.

6 DISCUSSION

It is important to note that this study had major limitations. Neither the attitude constructs nor the individual items of the survey were tested for validity or reliability. However, the survey, observations, and interviews indicate together may suggest that student attitude toward CS, culture, and their relationship did not improve. And perhaps equally importantly, students didn't seem to enjoy the class. The questions on the ability of students to program were similarly not validated, but again, agree with formative assessments, observation, and teacher perception that students did learn the material. This failure to engage students is particularly troubling as it's a core goal of both constructionism and culturally responsive teaching. This may be related to separate failures to both maintain a cultural connection and student agency.

Considering the emphasis on culturally responsive computing, the most dramatic issue for this implementation of the Cultural Computing curriculum was the inability to maintain the connections between culture and CS. After initially reading about the historical, political, and economic background sections associated with the CSDTs, the teacher did not address any of these social l dimensions of the cultural designs again. Moreover, the cultural context was constantly switched for different directed lessons, preventing any deeper exploration of the social aspects of the cultural designs by students. Consequently, while some students continued to see the embedded cultural connection, by the time students were working on physical renderings, some had also completely forgotten there was any cultural relevance.

The teacher did not explain why they were reluctant to revisit the social dimensions of the cultural material. Perhaps the fact that

it was a CS course and this was what the teacher is trained in, not social studies or history, reflects a lack of cultural competencies and an uncomfortableness in talking about cultural traditions with students in their courses. Like many other technical disciplines, CS often reproduces a social/technical dichotomy. Commonly understood, this dichotomy privileges what is seen as "technical"–the application of math or science–over what is deemed "social" or person-centered activities [9]. The field of science and technology studies has critiqued this dichotomy, arguing what is technical is also social and cultural and, conversely, what is social and cultural is also technical [3]. This multi-directional understanding is exactly what the CSDTs seek to make clear. To overcome this dichotomy, one strategy is to help teachers make strong connections between CS and culture by including and collaborating with those cultural expertise (e.g. an expert in African American braiding techniques) during a programming lesson [2, 12, 14]. While this requires extra effort on the part of the teacher, it has the added benefit of fostering classroom environments that are representative of and inviting for community experts, including people some students may be familiar with and even know personally.

The curriculum failed to instill student agency or allow self-regulation. Students were never allowed to pick their cultural context, as originally planned. Projects were limited or stripped down to make more time for guided lessons. Students were not allowed to set their own goals, overcome their own challenges, or asked to plan out work. Instead guided instruction, in which students performed tasks with the teacher, was so specific that even minor errors in entered values needed instructor intervention. Students were dropped into pre-existing programs, and many of them took longer to struggle to understand how a program worked than to complete a project (making their own script). This lack of control, choice, and relevancy may have influenced students' inattention or disinterest in the classroom.

Despite this, the success of students to learn material is some consolation. Furthermore, this has been extremely effective in collecting data for future improvements. Areas that have changed this coming year include improvements from small technical details, tasks, curricular details, and methodological details. The small technical details include defaulting to saving to the cloud and log in methods; tasks include giving a gradient in challenge difficulty, embedding cultural descriptions in the code, and starting from nothing rather than existing code; curricular details include eliminating lectures, including self-regulatory assignments, and asking students to reach out to their community; and methodological details include collecting more student interviews, structuring observations, and using validated tests where possible.

7 CONCLUSION

Cultural computing successfully improved student learning and performance over the school year. This is a strong indicator that CSDTs can be successfully used to form a year long curricula. However, the end goal of making this introductory high school programming course extremely engaging and facilitating rapid mastery of critical concepts like loops, variables, conditionals, and functions will require further iteration. As such the curriculum will be redesigned and deployed. It is hoped that a controlled study can be arranged.

ACKNOWLEDGMENTS

This work was supported by the National Science Foundation through NSF grant #1640014. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

REFERENCES

[1] William Babbitt, Michael Lachney, Enoch Bulley, and Ron Eglash. 2015. Adinkra Mathematics: A study of Ethnocomputing in Ghana. *Multidisciplinary Journal of Educational Research* 5, 2 (June 2015), 110–135. https://doi.org/10.17583/remie.2015.1399

[2] Audrey Bennett, Ron Eglash, Michael Lachney, and William Babbitt. 2016. Design Agency: Diversifying Computer Science at the Intersections of Creativity and Culture. *Revolutionizing Education through Web-Based Instruction* (2016), 35–56. https://doi.org/10.4018/978-1-4666-9932-8.ch003

[3] Wiebe E. Bijker, Thomas Parke Hughes, and Trevor J. Pinch. 1989. *The Social Construction of Technological Systems: New Directions in the Sociology and History of Technology.* MIT Press. Google-Books-ID: SUCtOwns7TEC.

[4] Sapna Cheryan, Allison Master, and Andrew N. Meltzoff. 2015. Cultural stereotypes as gatekeepers: increasing girls' interest in computer science and engineering by diversifying stereotypes. *Frontiers in Psychology* 6 (2015). https://doi.org/10.3389/fpsyg.2015.00049

[5] Dennis L Clason and Thomas J Dormody. 1994. Analyzing data measured by individual Likert-type items. *Journal of agricultural education* 35 (1994), 4.

[6] Ron Eglash, Juan E. Gilbert, and Ellen Foster. 2013. Toward Culturally Responsive Computing Education. *Commun. ACM* 56, 7 (July 2013), 33–36. https://doi.org/10.1145/2483852.2483864

[7] Ron Eglash, Mukkai Krishnamoorthy, Jason Sanchez, and Andrew Woodbridge. 2011. Fractal Simulations of African Design in Pre-College Computing Education. *Trans. Comput. Educ.* 11, 3 (Oct. 2011), 17:1–17:14. https://doi.org/10.1145/2037276.2037281

[8] Christopher Emdin. 2017. *For White Folks Who Teach in the Hood... and the Rest of Y'all Too: Reality Pedagogy and Urban Education* (reprint edition ed.). Beacon Press.

[9] Wendy Faulkner. 2000. Dualisms, Hierarchies and Gender in Engineering. *Social Studies of Science* 30, 5 (Oct. 2000), 759–792. https://doi.org/10.1177/030631200030005005

[10] Geneva Gay and James A. Banks. 2010. *Culturally Responsive Teaching: Theory, Research, and Practice* (2 edition ed.). Teachers College Press, New York.

[11] Z Score Inc. 2018. Evaluation Memo: RPI STEM+C Partnership, Summer 2018. (2018).

[12] Michael Lachney. 2017. Culturally responsive computing as brokerage: toward asset building with education-based social movements. *Learning, Media and Technology* 42, 4 (Oct. 2017), 420–439. https://doi.org/10.1080/17439884.2016.1211679

[13] Gloria Ladson-Billings. 2013. *The Dreamkeepers: Successful Teachers of African American Children.* John Wiley & Sons. Google-Books-ID: _GQMZsAX0igC.

[14] Dan Lyles, Michael Lachney, Ellen Foster, and Zoe Zatz. 2016. Generative Contexts: Generating value between community and educational settings. *Teknokultura* 13, 2 (2016), 613–637. https://dialnet.unirioja.es/servlet/articulo?codigo=5764112

[15] Jane Margolis, Rachel Estrella, Joanna Goode, Jennifer Jellison Holme, and Kim Nao. 2010. *Stuck in the Shallow End: Education, Race, and Computing.* The MIT Press, Cambridge, MA.

[16] Seymour Papert. 1980. *Mindstorms: Children, Computers, and Powerful Ideas.* Basic Books, Inc., New York, NY, USA.

[17] Kimberly A. Scott and Patricia Garcia. 2016. Techno-Social Change Agents: Fostering Activist Dispositions Among Girls of Color. *Meridians* 15, 1 (2016), 65–85. https://doi.org/10.2979/meridians.15.1.05

[18] Claude M. Steele. 2011. *Whistling Vivaldi: How Stereotypes Affect Us and What We Can Do* (reprint edition ed.). W. W. Norton & Company, New York.

[19] Stuart Zweben and Betsy Bizot. 2017. *Generation CS Continues to Produce Record Undergrad Enrollment; Graduate Degree Production Rises at both Master's and Doctoral Levels.* Technical Report. Computing Research Association. 3–51 pages. https://cra.org/crn/wp-content/uploads/sites/7/2017/05/2016-Taulbee-Survey.pdf

[20] US Census. 2017. *Race and Ethnicity.* Technical Report. U.S. Department of Commerce. https://www.census.gov/mso/www/training/pdf/race-ethnicity-onepager.pdf