

Scratch Encore: The Design and Pilot of a Culturally-Relevant Intermediate Scratch Curriculum

Diana Franklin*, David Weintrop†, Jennifer Palmer*, Merijke Coenraad‡, Melissa Cobian‡,

Kristan Beck‡, Andrew Rasmussen‡, Sue Krause*, Max White*, Marco Anaya*, Zachary Crenshaw*

*University of Chicago, Chicago, IL, USA

† University of Maryland, College Park, College Park, MD, USA

‡ Chicago Public Schools, Chicago, IL, USA

{dmfranklin,jenpalmer,sgkrause,mnwhite,manaya,zcrenshaw}@uchicago.edu;{weintrop,mcoenraa}@umd.edu;

{mcobian1,klbeck1,arasmussen}@cps.edu

ABSTRACT

While several introductory computer science curricula exist for children in K-8, there are few options that go beyond sequence, loops, and basic conditionals. The goal of this project is to not only fill this gap with a high-quality curriculum supported by complete instructional materials, but to also do so with an equity-balanced curriculum. That is, a curriculum that values advancing equity equally with student learning outcomes. In this paper, we introduce barriers to equity in public school classrooms, pedagogical approaches to culturally-relevant curricula, and how our Scratch Encore curriculum is designed to support equity-balanced learning. Finally, we present results of our pilot year, including early evidence of students taking advantage of the culturally-relevant design aspects.

CCS CONCEPTS

• **Social and professional topics** → **Computer science education; Computational thinking;**

KEYWORDS

Computational Thinking; Scratch; K-12 education; Culturally-Relevant Instruction

ACM Reference Format:

Diana Franklin*, David Weintrop†, Jennifer Palmer*, Merijke Coenraad‡, Melissa Cobian‡, Kristan Beck‡, Andrew Rasmussen‡, Sue Krause*, Max White*, Marco Anaya*, Zachary Crenshaw*. . Scratch Encore: The Design and Pilot of a Culturally-Relevant Intermediate Scratch Curriculum. In *The 51st ACM Technical Symposium on Computer Science Education (SIGCSE '20)*, March 11–14, 2020, Portland, OR, USA. ACM, New York, NY, USA, 7 pages. <https://doi.org/10.1145/3328778.3366912>

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
 SIGCSE '20, March 11–14, 2020, Portland, OR, USA
 © 2020 Association for Computing Machinery.
 ACM ISBN 978-1-4503-6793-6/20/03...\$15.00
<https://doi.org/10.1145/3328778.3366912>

1 INTRODUCTION

Succeeding in the goal of bringing computer science (CS) to all students requires high-quality, sustained efforts throughout K-12 education. Increasingly, CS and computational thinking (CT) are recognized as essential topics for all students. Most parents (90%) want CS instruction for their children [45]. In response to these trends, school districts across the United States, including San Francisco, New York, and Chicago, have pledged to bring CS experiences to all students in their schools through large-scale CS initiatives.

Chicago Public School District (CPS) is at the forefront of the CS for All movement. In 2013, CPS announced CS4All, an ambitious plan to bring computer science to all students, pledging to a) teach a high-quality, relevant computer science course at every high school, b) offer coursework in 25% of elementary schools (defined as K-8), and c) make computer science a core graduation requirement. As of 2017, CPS has introduced CS in 90 of 477 elementary schools, 63 of 106 district-run high schools, and added a computer science graduation requirement for students starting high school in 2016. They have now updated their pledge of 25% of elementary schools to 75%. It is within this context that the gap in high-quality intermediate curriculum for upper-elementary students (ages 10-14) became a barrier to equity.

Computer science high school curricula and teacher professional development have been targets of NSF funding for several years, and prominent organizations such as Code.org and Google have created polished introductory teaching materials and training for inexperienced elementary school teachers. These efforts have helped make progress towards the goals of CS for all but introduce a gap for schools to fill between heavily scaffolded introductory courses and more advanced programming courses designed for high school students. The extremes of expensive commercial solutions or free, ad-hoc, individual activities available at this level feed inequity by requiring either money or high teacher expertise. This situation disproportionately impacts students in under-resourced schools that often serve students of underrepresented minorities in computing.

In order to fill this gap, we introduce Scratch Encore, an intermediate Scratch curriculum for formal learning environments that balances goals of culturally-relevant curriculum design, Constructionist learning principles, and working within the constraints of formal learning environments and teachers who may be relatively inexperienced with the content.

This paper presents the design of Scratch Encore, followed by an exploration of two, interrelated research questions related to our pilot study: (1) How do teachers perceive it? And (2) Is Scratch Encore succeeding in providing opportunities for creativity and cultural expression while also introducing computing concepts in a structured way? And if so, how is the curriculum enabling it?

2 THEORY AND PRIOR WORK

In this section, we present the major theoretical and practical influences that have shaped the Scratch Encore curriculum and instructional materials. This includes prior work on elementary CS education, curricula and computing education outreach initiatives designed to broaden participation in the field, and work on elementary CS learning trajectories.

2.1 Programming Environments

While traditionally a subject for high school and beyond, there is growing demand for bringing CS into K-8 classrooms. Early work by Papert and colleagues found that programming was accessible to younger students and could serve as a powerful pedagogical strategy [20, 33, 34]. In the last decade, bringing CS to K-8 has grown in popularity and has been facilitated by programming tools designed for young students [11, 23]. An increasingly popular approach for creating engaging and accessible programming environments is the graphical, block-based programming interface [1, 47]. Visual block-based languages use a programming-command-as-puzzle-piece metaphor to visually render syntax rules and allow users to use a drag-and-drop approach to construct programs. Block-based programming tools provide numerous scaffolds that make programming easier, including limiting syntax errors, providing visual cues on how commands can be used, and providing an easy way to browse available commands [48]. Block-based programming interfaces have been used to create an array of programming environments and tools, including museum exhibits [22], libraries for controlling robots [29], and tools for creating mobile applications [9].

2.2 Pedagogical Approach

Much of the curricular and programming environment design effort for younger students has drawn inspiration from the Constructionist design and learning approach [33]. Constructionism emphasizes focusing on the powerful ideas of a discipline, and foregrounds learning-by-doing, giving students opportunities to construct personally meaningful, public artifacts. Constructionist learning experiences are often designed so as to give the learner agency in the activity, enabling self-directed learning and supporting exploration in the process of constructing public and shareable artifacts. Additionally, Constructionist designs provide students opportunities to present their work to peers and share their ideas and experiences through teacher-led classroom discussions. Providing a public forum for students to share their ideas, experiences, and programs encourages students to become more invested in their own learning and promotes an inclusive learning environment in which all students can see themselves as members of a computing community. This focus on student-driven learning, not specific technical content, has coincided with work in informal spaces [28] and placed

an emphasis on self-directed learning and online collaboration [15, 39] and the practices of computing [3].

There is also a growing library of curricula designed for early elementary learning created using specially-designed programming environments that lead students through a more structured experience. This includes curricula design by Code.org [6], the Foundations for Advancing Computational Thinking (FACT) curriculum [19], and the Creative Computing curriculum [4]. A middle-ground between student-driven learning and more structured activities can be seen in San Francisco Unified School District's Introduction to Computer Science, which provides instruction on specific CS concepts, followed by an open-ended project that follows Constructionist principles. A common strategy used in these and other introductory computing curricula used in K-8 is the Use->Modify->Create approach, which begins by having students use functional programs that demonstrate the concept being taught before making modifications to an existing program and then finally being given an opportunity to create a new program on their own [27]. This strategy provides a scaffolded approach to introducing new concepts while also retaining aspects of Constructionist learning.

2.3 Culturally-Relevant CS Curricula

A major contribution of the field of Learning Sciences is the recognition that aligning content with the culture, prior experience, and social values and norms of the learner can improve retention, engagement, and learning outcomes [26, 30, 32, 38]. Examples of culturally-responsive curricula [25, 18] can be found across math [31], science [14], and English language arts [8]. In computer science, culturally responsive computing curricula follow many of the same principles [42, 12, 43].

We view culturally-relevant curriculum development as consisting of four dimensions, not all of which are required to be in every learning experience. First, when explaining concepts, examples should be used that are drawn from students' current understandings and experiences, what Papert called "cultural syntonicity" [33]. Second, when possible, culturally-relevant curricula should draw upon the cultural heritage of students so as to help increase their sense of belonging. This can take the form of including visual elements drawn from their culture or situating programming activities within contexts that carry cultural significance with groups of students [18, 25]. Third, culturally-relevant curricula can draw upon current youth culture, such as social practices (e.g. texting), media and video games (e.g. Fortnite), or other elements associated with youth culture. Examples of following these design approaches can be seen in the Animal Tlatoque summer camp [17], which was built around themes of Mesoamerican culture and animal conservation and Code.org's approach of partnering with major entertainment properties such as Star Wars, Flappy Bird, and Minecraft. Another approach for introducing computing ideas in a culturally relevant way is to engage students in artistic endeavors such as making [2, 5, 46, 21] and image and sound manipulation [16, 13]. In such contexts, computing is used as a medium for students to participate in activities that align with their culture, be it youth culture (e.g. music-making) or historical culture (e.g. beading or sewing). Finally, beyond the cultural resources integrated into the learning activity, all students should have creative opportunities to tailor

the projects to their own selves. For example, the Exploring Computer Science curriculum accomplishes this goal by framing “CS learning around [students’] own questions and interests, culturally-relevant pedagogy drawing on students’ funds of knowledge, and core CS concepts relating to our everyday uses of new technology and participatory media” [40].

2.4 Computer Science Learning Progressions

While several approaches to computing education have been studied, there have been few systematic studies that investigate learning trajectories for students across several grades. We will draw upon two bodies of knowledge: The K-12 Computer Science Framework, which is being used to inform standards across the country [24] and learning trajectories developed using learning goals drawn from a literature of over 100 computer science education publications and organized by concept or practice, including sequence, repetition, conditionals, decomposition, and debugging [37, 36, 35].

2.5 Barriers to Equity

The lack of freely-available intermediate CS courses for elementary students is a barrier to equity. Instead of having complete, well-organized and structured curricula, teachers often search online for individual lessons and then adapt them to their needs creating their own curriculum one piece at a time. Alternatively, schools purchase closed packages that provide all-in-one solutions but are often costly to the school or district. These strategies are ill-suited for under-resourced schools without an experienced CS teacher.

An alternative venue for learning about CS is through informal venues, such as after-school programs, cultural institutions (e.g. museums), libraries, or summer camps. While such approaches provide a productive introduction to computing, they often lack the level of in-depth and extended exposure that formal classrooms afford. Also, many of these opportunities require investments from the attendees, either in the form of fees (e.g. registration, materials) or transportation requirements (e.g. pick-up/drop-off at certain times/places), producing inequitable access and reaching only a small, unrepresentative portion of the population. Finally, informal CS opportunities often do not connect with in-school opportunities, meaning interest developed does not necessarily transfer to enrollment in further computer science learning opportunities. Additionally, while free resources do exist, they are often difficult to find or require parents to have prior knowledge of their existence in order to take advantage of them, which further reinforces the disparity in equity and access [10].

3 SCRATCH ENCORE DESIGN

Scratch Encore[44] is a 2-3 year intermediate Scratch curriculum designed for students who have had a least one year of introductory coding (e.g. Code.org’s CS Discoveries, Creative Curriculum, or any other curriculum that covers sequence, loops, and conditionals). Scratch Encore is comprised of 15 learning modules, each of which is 4-5 class sessions in length and follows the same structure. Below we present the two defining features of Scratch Encore - multiple strands that provide different themes for the same learning modules and the within-module flow of Use->Modify->Create - followed by

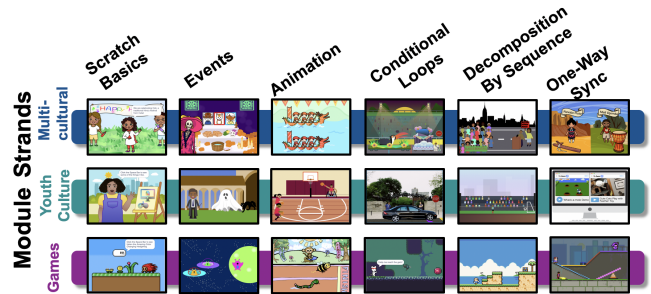


Figure 1: Multiple Strands providing three choices of Use/Modify activities for the first 6 modules.

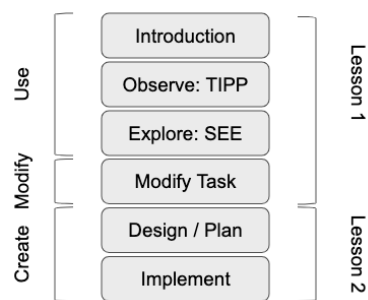


Figure 2: Common module design with two activities.

how those features satisfy our overall goals of balancing equity and learning in a formal learning environment.

The first defining feature of Scratch Encore is its modular, sequential structure (Fig. 1). Scratch Encore is comprised of 15 modules, each covering a different topic (the columns in Fig. 1). At the same time, each module can be taught using one of three strands (the rows in Fig. 1). Each strand situates the content of the activities within a specific context designed to resonate with youth. Looking across the strands, each module covers the exact same computer science material, just with a different graphical presentation. Scratch Encore currently includes three strands: Multicultural, Youth Culture, and Gaming. The Multicultural strand draws on celebrations, events, or traditions, including many that are used or celebrated by cultures often not highlighted in dominant American culture. These include the Million Women March, Dia de los Muertos, and Martin Luther King Jr.’s famous “I Have a Dream” speech. The second strand, Youth Culture, includes themes that resonate with many youth today, including sports and social networks. The third strand is the Gaming strand, which was teachers’ top subject that they identified their students as wanting to learn with computing. The strands and the specific contexts within them were developed through participatory design sessions with various education stakeholders, including teachers, students, parents, and administrators[7]. As teachers progress through the 15 modules, they can choose the strand that they think will best resonate with their students (Figure 1).

The second defining feature of Scratch Encore is its lesson flow, based on the Use->Modify->Create pedagogical approach [27] (Figure 2). Each module begins with an introduction to the concept tying the content to examples from students’ everyday lives. Students then receive example code and go through activities focused on observing and exploring that code before making small modifications. Finally, the Create activity involves a largely open-ended challenge where students are given design prompts that lead them towards a project that can utilize the new concept. Throughout the curriculum, care was taken to ensure the activities present authentic uses of the content being taught. This means the code students see as they go through the curriculum reflects how we would actually solve the problem, even if we had more advanced knowledge. In this way, the curriculum not only teaches how a given concept works but when to use it.





Designing for Culturally-Relevant Computing. Scratch Encore consists of all four elements of culturally-relevant curriculum design discussed in section 2.3. Prior to the Use->Modify activity, an Engage discussion relates the concept to students’ daily lives. For example, synchronization is introduced with a knock-knock joke in which the teacher uses the wrong timing with the students. The Multicultural strand allows teachers to choose projects that resonate with the cultural heritage of students in their own classroom, aligning with the second dimension of culturally responsive curricula. To align with contemporary youth trends and interests, the Gaming and Youth Culture strands, as well as the Create prompts, are designed around contemporary youth culture. Finally, students are able to personalize projects and put something of themselves into them. A first-level includes providing sprites with multiple skin tones and varying gender representations. Deeper levels include asking students to customize MLK Jr.’s speech to their own wishes for equality and putting memories of their own family members in the Ofrenda in Dia de los Muertos.

Designing for Student Variation in Skills / Exceptionalities. In order to support variation in student skills, we utilize a TIPP&SEE learning strategy to step students through the provided example code, mediated through a worksheet[41]. It leads students in purposeful play with recorded observation, finding and making predictions about code related to particular actions they observed, and deliberate tinkering by making code changes to understand how individual blocks work. Extensions to the Modify and Create activities allow students to go above and beyond what is required if they are able and interested. Finally, classrooms that need to review a topic can repeat a module by choosing activities from a different strand.

Designing for Teacher Variation. In order to support teachers who are new to CS, programming in general, or Scratch in particular, Scratch Encore provides several resources. These include teacher guides that provide key ideas to contextualize the learning goal(s) of the module, “Engage” sections that guide teachers on how to relate the concepts to students’ daily lives, automated analysis that provides instant feedback on the state of completion for the whole class, and quizzes to gauge student understanding.

Start with TIPP&SEE!
Get a TIPP from the Project Page:
 Title: What is the title of the project? Does it tell you something about the project?
 Instructions: What do the instructions tell you to do?
 Purpose: What is the purpose of this activity? What will this code teach you?
 Play: Run the project and see what it does! Look at which sprites are doing the actions.

What happened when you played the project? Circle or highlight the action(s) that happened for each event.

When I clicked the green flag:			
 talked	 talked	 talked	 talked
waved	changed size	changed size	changed size
did nothing	did nothing	did nothing	did nothing





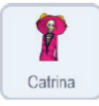



When I pressed the space bar:			
 talked	 talked	 talked	 talked
waved	changed size	changed size	changed size
did nothing	did nothing	did nothing	did nothing

Figure 3: TIPP worksheet guides students through mindfully playing and observing the provided Scratch project.

SEE Inside:
 For each Sprite, record the # of costumes and circle the Events that have scripts.

Sprite	# costumes	Circle the Event blocks that have scripts
 Catrina		  

Explore the project! Click on the Left sprite and...

- Change the number 100 to 300 in the change size by 100 block, then click on the Left picture again. What happens? Why?
- Change the number 100 to -50 in the change size by 100 block, then click on the Left picture again. What happens? Why?
- Remove the change size by -100 block, then click on the Left picture multiple times. What happens? Why?

Figure 4: SEE worksheet guides students through finding elements in the Scratch interface and deliberate exploration (tinkering) to discover how blocks work.

3.1 Example Module: Events

To more clearly illustrate the principles enacted in Scratch Encore, we provide details on Module 2: Events.

The module begins with an introductory discussion relating the term events to daily life (holidays, life milestones) and redefining it within computing (tapping on a shoulder, ringing a doorbell) and then to the computer itself (clicking on a mouse, typing on the keyboard). We then provide example code drawn from Mexican culture - an Ofrenda set up for Dia de los Muertos. The example program includes scripts for two sprites: the skeleton and the left-most picture on the Ofrenda. The skeleton talks about Ofrendas when the green flag is clicked, and the picture enlarges when clicked, says something, and shrinks back down. Students are scaffolded in their exploration of the project with a TIPP&SEE worksheet, shown in Figures 3 and 4. TIPP orients students to the project and purpose





 Planning Your Project:			
Use the Five W's to plan your project. Write your answers in the space provided. You may not need to use all five for your project.			
Who will be in the project (sprites)?	#1: _____	#2: _____	#3: _____
What are they doing? Say, Move, Change Size by ___ blocks			
When? The events this sprite will respond to are: <i>Choose at least two for each sprite. All three events need to be chosen at least once</i>			
Where (Choose your Stage/Backdrop)? _____			

Figure 5: Planning document for a Create activity to encourage the use of multiple sprites and events.

and steps them through playing and recording observation of the project execution. SEE asks them to navigate the Scratch interface and find the code that performed the actions they observed. They then step through small, deliberate modifications to learn how the change size block works.

In the Modify step, students are invited to make the project their own by choosing three special people or pets about whom they would like to share memories or feelings. Many costumes are provided to accommodate a variety of skin tones or animals. Students choose costumes to fill each picture frame and then write scripts to enlarge the sprite, say something about the subject in the picture frame, and shrink the sprite back down.

In lesson 2, students create a project about a holiday. They are asked to share different facts about the holiday. A planning document is provided to help with the selection of sprites and events to reinforce what they learned in Lesson 1 (Figure 5).

4 METHODS

In this section, we detail the recruitment of participants in our study, followed by data collected and data analysis methods.

4.1 Recruitment and Participants

Teachers were recruited from a major metropolitan school district. Twenty-seven teachers attended professional development, and eight were included in our study. Interested teachers were chosen for inclusion based on the number of courses they taught (preferring more classes) and the grade levels (5th-7th grade preferred). In total, there were 13 classes, 5th-8th grades, including 271 students.

4.2 Data Collection

Several types of data were collected and analyzed, including TIPP&SEE worksheets, observations, teacher interviews, teacher focus groups, and student computational artifacts. Researchers observed instruction for each teacher three times during the school year (same grade level, typically same classroom). Teachers were interviewed following each observation and participated in focus groups with

other teachers at the end of the school year. Computational artifacts were publicly available online, organized by pilot teachers into classrooms and studios for analysis.

4.3 Data Analysis

Data was analyzed to answer two overall questions, one about the use of Scratch Encore materials, and a second about student use of the second module - Events.

Scratch remix histories were used to determine the number of classrooms and students completing Scratch Encore projects as well as how far each classroom progressed through the curriculum. Timestamps for remixed projects were used to calculate the pace and frequency that classrooms completed activities.

First, worksheets were coded for accuracy and completeness. Second, artifacts from a subset of students in pilot classrooms were analyzed to understand in what ways students completed tasks beyond the requirements of the project. Finally, static analysis modules were coded in JavaScript that analyzed the code in all projects in a studio and produced a spreadsheet of results. These looked for not only the required elements, but also examples of the types of additional actions that were found through hand inspection.

5 RESULTS

Here we present teacher feedback and a case study of Module 2 - Events. Specifically, we want to know to what degree students engage with the activities in the ways intended, taking advantage of opportunities for personalization and creativity within the learning structure.

5.1 Teacher Feedback

We next present teacher feedback, separated into three categories: overall impressions of the curriculum; comments that specifically address the engagement aspects of the curricular design; and comments about the materials and design for learning.

Overall, feedback was incredibly positive for using Scratch Encore. On Likert scale questions from 1-5, teacher responses were an average of 4.29 / 5 for Scratch Encore being successful in their classroom and 4.38 / 5 for it being engaging to their students. The difficulty was appropriate, answering 2.97 / 5 to a question of whether the material was too difficult (1) or too easy (5). One teacher identified *“Starting with the basics of Scratch and explaining everything...I like having the detailed lessons of what to do and the extension idea and special note boxes with Scratch skills”* as a strength of the curriculum.

Three modules were specifically mentioned as being relevant to the students - the Ofrenda as a *“cultural learning tool”*, the public transportation as *“relevant”* to their own lives, and the soccer module for *“familiarity, ease, and relevance.”*

When asked what their favorite elements were or the biggest strength of the curriculum, several teachers mentioned the TIPP&SEE *“worksheet”* or *“process”*. One teacher said, *“I also like having my students do an old school worksheet, because it helps them to think about what they just learned and I can use them as exit tickets/mini assessments.”* Different teachers identified all aspects of the Use->Modify->Create approach as a favorite or strength. One liked *“the given starting code for one of the animals,”* another *“Experimenting*

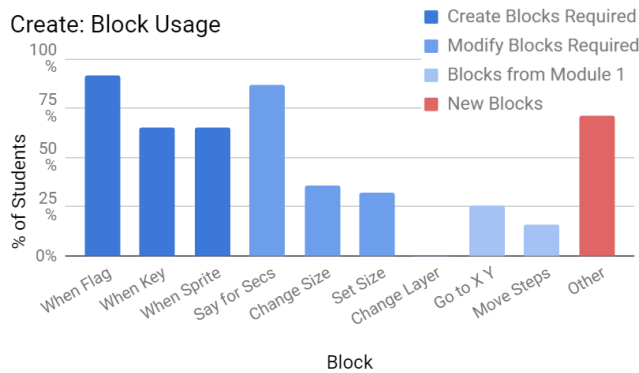


Figure 6: Percentage of students who used each of the introduced Scratch blocks in their Create lesson projects.

with code to show how changes made things go faster or slower,” and a third noted that “*They always enjoy the creative aspect of modifying the project.*” Several pointed out the open-ended Create lessons as favorites. Finally, several teachers identified our graphical organizers for broadcast/receive as a strength, writing “*The broadcast concept was best understood visually on the worksheet.*”

Finally, teachers noted that the lessons took more time than anticipated. Specifically, some Use->Modify activities required two class periods - one to explore the project and one to modify the project. In addition, students would be able to complete larger Create projects if given two days to plan and implement the projects.

5.2 Events Case Study

We now take a deep dive into one module to see to what degree the different design elements resulted in desired behaviors in students. We focus on three aspects of Module 2 - Events. We are specifically concerned with whether students were still able to be creative and explore within the structure of Use->Modify->Create and whether the culturally-relevant design strategies resulted in students personalizing their projects.

In Lesson 1, Events Ofrenda, students had a highly structured project in which they were asked to choose important people or pets, make scripts that expanded their size when clicked, and put in a Say bubble about the person/animal. We can see that within this structure, students still explored beyond the required elements. Our analysis shows that 4.92% of students changed the background of the project, 31.32% of students added or made use of sounds, and 79.64% of students added or changed at least one new costume.

In Lesson 2, My Favorite Holiday, students create a project to share information about their favorite holiday. The hope is that because this is an open-ended project, students will incorporate blocks they learned about in previous lessons. We can see the block usage in Figure 6. The dark blue bars represent blocks that are required for this activity. We can see that 60-90% of students used the required blocks in their projects. In addition, over 80% used the Say block which was used in the previous activity and introduced in Module 1. At least 25% of students used size manipulation blocks introduced in Lesson 1, and others used the Go To and Move blocks from Module 1. A second important goal of a Create project is to

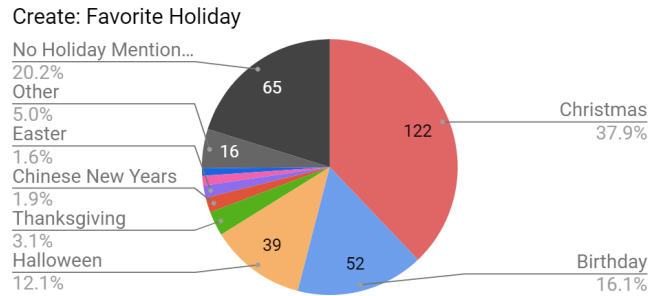


Figure 7: Students' choice of holiday for their Create lesson projects.

encourage students to explore blocks not yet introduced. We can see that almost 75% of students used blocks not formally introduced in the curriculum.

Finally, the purpose of the theme is to have students personalize their project following culturally-relevant pedagogical principles. We can see from Figure 7 that almost 80% of students did, indeed, choose a holiday of some sort. Christmas was by far the most commonly chosen holiday (37.9%), with other mainstream American holidays like Halloween (12.1%), Thanksgiving (3.1%), and Easter (1.6%) having a notable presence. Several students paid homage to their particular cultural heritage: Chinese New Year was the 5th most popular holiday choice (1.9%), one student elaborated on the customs of Day of the Dead, and another student chose Juneteenth, a holiday commemorating African American freedom. A majority of students chose holidays which they themselves celebrate and that are culturally significant to them, with over 1/3 of students incorporating their family into their projects. Other students took a more flexible definition of holiday, using their own birthday as a holiday, which 16.1% did, or celebrating less conventional holidays, like national food days, or making up their own holiday.

6 CONCLUSIONS

This paper introduces Scratch Encore, an intermediate Scratch programming curriculum 10-14 year olds. Scratch Encore integrates the Use->Modify->Create pedagogical approach, culturally-relevant pedagogy, and the TIPP&SEE learning strategy to create a set of materials that attend to issues of equity while also introducing foundational ideas of computer science. Finally, pilot results show high teacher interest and desirable and productive student engagement.

7 ACKNOWLEDGEMENTS

We would like to thank the teachers and students who piloted Scratch Encore. This material is based upon work supported by the National Science Foundation under Grant No. 1738758.

REFERENCES

- [1] David Bau et al. "Learnable programming: blocks and beyond". In: *arXiv preprint arXiv:1705.09413* (2017).
- [2] P. Blikstein. "Digital fabrication and 'making' in education: The democratization of invention". In: *FabLabs: Of Machines, Makers and Inventors*. Ed. by J. Walter-Herrmann and C. Büching. Transcript Publishers, 2013, 1--21.
- [3] K. Brennan. "Learning computing through creating and connecting". In: *Computer* 46.9 (2013), pp. 52-59.

- [4] K. Brennan, C Balch, and M. Chung. *Creative computing*. 2014. URL: <http://creativecomputing.gse.harvard.edu/guide/>.
- [5] L. Buechley and H. Perner-Wilson. "Crafting Technology: Reimagining the Processes, Materials, and Cultures of Electronics". In: *ACM Trans Comput-Hum Interact* 19.3 (Oct. 2012), 21:1–21:21.
- [6] *Code.org CS Curricula*. URL: <https://curriculum.code.org/>.
- [7] Merijke Coenraad et al. "Enacting Identities: Participatory Design As a Context for Youth to Reflect, Project, and Apply Their Emerging Identities". In: *Proceedings of the 18th ACM International Conference on Interaction Design and Children*. IDC '19. Boise, ID, USA, 2019, pp. 185–196. ISBN: 978-1-4503-6690-8.
- [8] Carol D Lee. "Is October Brown Chinese? A Cultural Modeling Activity System for Underachieving Students". In: *American Educational Research Journal* 38 (2001), p. 97.
- [9] E. Spertus D. Wolber H. Abelson and L. Looney. *App Inventor: Create Your Own Android Apps*. Sebastopol, California: O'Reilly Media, 2011.
- [10] Betsy DiSalvo, Cecili Reid, and Parisa Khanipour Roshan. "They can't find us: the search for informal CS education". In: *Proceedings of the 45th ACM technical symposium on Computer science education*. ACM, 2014, pp. 487–492.
- [11] Caitlin Duncan, Tim Bell, and Steve Tanimoto. "Should Your 8-year-old Learn Coding?" In: *Proceedings of the 9th Workshop in Primary and Secondary Computing Education*. WiPSCe '14. Berlin, Germany, 2014, pp. 60–69. ISBN: 978-1-4503-3250-7.
- [12] Ron Eglash et al. "Culturally responsive computing in urban, after-school contexts: Two approaches". In: *Urban Education* 48.5 (2013), pp. 629–656.
- [13] Shelly Engelman et al. "Creativity in Authentic STEAM Education with EarSketch". In: *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education*. SIGCSE '17. Seattle, Washington, USA, 2017, pp. 183–188. ISBN: 978-1-4503-4698-6.
- [14] F. Erickson and K. Gutierrez. "Culture, Rigor, and Science in Educational Research". In: *Educ. Res.* 31.8 (2002), p. 21.
- [15] Deborah A Fields, Michael Giang, and Yasmin Kafai. "Programming in the wild: trends in youth computational participation in the online scratch community". In: *Proceedings of the 9th workshop in primary and secondary computing education*. ACM, 2014, pp. 2–11.
- [16] Andrea Forte and Mark Guzdial. "Computers for Communication, Not Calculation: Media As a Motivation and Context for Learning". In: *Proceedings of the Proceedings of the 37th Annual Hawaii International Conference on System Sciences*. HICSS '04. 2004, 10–pp. ISBN: 0-7695-2056-1.
- [17] Diana Franklin et al. "Animal Tlatoque: Attracting Middle School Students to Computing Through Culturally-relevant Themes". In: *Proceedings of the 42nd ACM Technical Symposium on Computer Science Education*. SIGCSE '11. Dallas, TX, USA, 2011, pp. 453–458. ISBN: 978-1-4503-0500-6.
- [18] G. Gay. "Culturally responsive teaching principles, practices, and effects". In: *Handbook of urban Education*. Ed. by H. R. Milner and K. Lomoty. Abingdon: Routledge, 2013, pp. 353–372.
- [19] S. Grover, R. Pea, and S. Cooper. "Designing for deeper learning in a blended computer science course for middle school students". In: *Comput. Sci. Educ* 25.2 (Apr. 2015), pp. 199–237.
- [20] I. Harel and S. Papert. "Software design as a learning environment". In: *Interactive Learning Environments* 1.1 (1990), pp. 1–32.
- [21] Nathan Holbert. "Leveraging Cultural Values and 'Ways of Knowing' to Increase Diversity in Maker Activities". In: *Int. J. Child-Comp. Interact.* 9.C (Dec. 2016), pp. 33–39. ISSN: 2212-8689.
- [22] Michael S. Horn et al. "Frog Pond: A Codefirst Learning Environment on Evolution and Natural Selection". In: *Proceedings of the 2014 Conference on Interaction Design and Children*. IDC '14. Aarhus, Denmark, 2014, pp. 357–360. ISBN: 978-1-4503-2272-0.
- [23] Caitlin Kelleher and Randy Pausch. "Lowering the Barriers to Programming: A Taxonomy of Programming Environments and Languages for Novice Programmers". In: *ACM Comput. Surv.* 37.2 (June 2005), pp. 83–137. ISSN: 0360-0300.
- [24] *K-12 Computer Science Framework*. 2016. URL: <https://k12cs.org>.
- [25] G. Ladson-Billings. *The Dreamkeepers: Successful teachers of African American children*. 2nd ed. United States of America: Josey-Bass, 2009.
- [26] C. D. Lee. "The Culture of Everyday Practices and their implications for learning in school". In: *Culture, literacy, & learning: Taking bloom in the midst of the whirlwind*. Teachers College Pr, 2007, pp. 227–238.
- [27] Irene Lee et al. "Computational thinking for youth in practice". In: *Acm Inroads* 2.1 (2011), pp. 32–37.
- [28] John H. Maloney et al. "Programming by Choice: Urban Youth Learning Programming with Scratch". In: *SIGCSE Bull.* 40.1 (Mar. 2008), pp. 367–371. ISSN: 0097-8418.
- [29] Amon Millner and Edward Baafi. "Modkit: Blending and Extending Approachable Platforms for Creating Computer Programs and Interactive Objects". In: *Proceedings of the 10th International Conference on Interaction Design and Children*. IDC '11. Ann Arbor, Michigan, 2011, pp. 250–253. ISBN: 978-1-4503-0751-2.
- [30] L. C. Moll et al. "Funds of Knowledge for Teaching: Using a Qualitative Approach to Connect Homes and Classrooms". In: *Theory Pract.* 31.2 (Apr. 1992), pp. 132–141.
- [31] N. S. Nasir, V. Hand, and E. V. Taylor. "Culture and Mathematics in School: Boundaries Between 'Cultural' and 'Domain' Knowledge in the Mathematics Classroom and Beyond". In: *Rev. Res. Educ.* 32.1 (Feb. 2008), pp. 187–240.
- [32] N. S. Nasir et al. "Learning as a cultural process: Achieving equity through diversity". In: *The Cambridge handbook of the learning sciences*. Ed. by R. K. Sawyer. Cambridge Univ Pr, 2006.
- [33] S. Papert. *Mindstorms: Children, Computers, and Powerful Ideas*. Basic Books, Inc., 1980. ISBN: 0-465-04627-4.
- [34] S. Papert et al. *Final report of the Brookline Logo Project: Project summary and data analysis (Logo Memo 53)*. Tech. rep. Cambridge, MA: MIT Logo Group, Sept. 1979.
- [35] Kathryn M. Rich et al. "A K-8 Debugging Learning Trajectory Derived from Research Literature". In: *Proceedings of the 50th ACM Technical Symposium on Computer Science Education*. SIGCSE '19. Minneapolis, MN, USA, 2019, pp. 745–751. ISBN: 978-1-4503-5890-3.
- [36] Kathryn M. Rich et al. "Decomposition: A K-8 Computational Thinking Learning Trajectory". In: *Proceedings of the 2018 ACM Conference on International Computing Education Research*. ICER '18. Espoo, Finland, 2018, pp. 124–132. ISBN: 978-1-4503-5628-2.
- [37] Kathryn M. Rich et al. "K-8 Learning Trajectories Derived from Research Literature: Sequence, Repetition, Conditionals". In: *Proceedings of the 2017 ACM Conference on International Computing Education Research*. ICER '17. Tacoma, Washington, USA, 2017, pp. 182–190. ISBN: 978-1-4503-4968-0.
- [38] B. Rogoff. *The cultural nature of human development*. Oxford Univ Pr, 2003.
- [39] Ricarose Roque, Yasmin Kafai, and Deborah Fields. "From Tools to Communities: Designs to Support Online Creative Collaboration in Scratch". In: *Proceedings of the 11th International Conference on Interaction Design and Children*. IDC '12. Bremen, Germany, 2012, pp. 220–223. ISBN: 978-1-4503-1007-9.
- [40] J. J. Ryoo et al. "Democratizing computer science knowledge: transforming the face of computer science through public high school education". In: *Learn. Media Technol.* 38.2 (June 2013), pp. 161–181.
- [41] Jean Salac et al. "TIPP&SEE: A Learning Strategy to Guide Students through Use->Modify Scratch Activities". In: *Proceedings of the 2019 ACM SIGCSE Technical Symposium on Computer Science Education*. SIGCSE '19. ACM. Portland, OR, USA, 2019.
- [42] K. A. Scott, K. M. Sheridan, and K. Clark. "Culturally responsive computing: a theory revisited". In: *Learning, Media and Technology* 40.4 (2015), pp. 412–436.
- [43] Kimberly A Scott and Mary Aleta White. "COMPUGIRLS standpoint: Culturally responsive computing and its effect on girls of color". In: *Urban Education* 48.5 (2013), pp. 657–681.
- [44] *Scratch Encore*. URL: <https://www.canonlab.org/scratchencoremodules>.
- [45] *Searching for Computer Science: Access and Barriers in U.S. K-12 Education*. Tech. rep. Google, 2015.
- [46] Kristin A. Searle and Yasmin B. Kafai. "Boys' Needlework: Understanding Gendered and Indigenous Perspectives on Computing and Crafting with Electronic Textiles". In: *Proceedings of the Eleventh Annual International Conference on International Computing Education Research*. ICER '15. Omaha, Nebraska, USA, 2015, pp. 31–39. ISBN: 978-1-4503-3630-7.
- [47] David Weintrop. "Block-based Programming in Computer Science Education". In: *Commun. ACM* 62.8 (July 2019), pp. 22–25. DOI: 10.1145/3341221. URL: <http://doi.acm.org/10.1145/3341221>.
- [48] David Weintrop and Uri Wilensky. "To Block or Not to Block, That is the Question: Students' Perceptions of Blocks-based Programming". In: *Proceedings of the 14th International Conference on Interaction Design and Children*. IDC '15. Boston, Massachusetts, 2015, pp. 199–208. ISBN: 978-1-4503-3590-4.