TIPP&SEE: A Learning Strategy to Guide Students through Use–>Modify Scratch Activities

Jean Salac*, Cathy Thomas†, Chloe Butler†, Ashley Sanchez†, & Diana Franklin*
*University of Chicago, Chicago, IL
† Texas State University, San Marcos, TX, USA
{salac,dmfranklin}@uchicago.edu;{thomascat,cbb64,a_s1122}@txstate.edu

ABSTRACT

With the rise of Computational Thinking (CT) instruction at the elementary level, it is imperative that elementary computing instruction support a variety of learners. A popular pedagogical approach for this age group is Use–>Modify–>Create, which introduces a concept through a more scaffolded, guided instruction before culminating in a more open-ended project for student engagement. Yet, there is little research on student learning during the Use–>Modify step, nor strategies to promote learning in this step. This paper introduces TIPP&SEE, a metacognitive learning strategy that further scaffolds student learning during this step. Results from an experimental study show statistically-significant performance gains from students using the TIPP&SEE strategy on nearly all assessment questions of moderate and hard difficulty, suggesting its potential as an effective CS/CT learning strategy.

CCS CONCEPTS

• **Social and professional topics → Computer science education**; *Computational thinking*; *K-12 education*;

KEYWORDS

learning strategy, computational thinking, Scratch, elementary education

ACM Reference Format:
Jean Salac*, Cathy Thomas†, Chloe Butler†, Ashley Sanchez†, & Diana Franklin*. . TIPP&SEE: A Learning Strategy to Guide Students through Use–>Modify Scratch Activities. 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/10.1145/3328778.3366821

1 INTRODUCTION

To provide equity in K-12 computing education, momentum has been building for integrating computer science into elementary school classrooms in school districts such as Chicago, San Francisco, and New York City. However, many curricula and programming platforms were designed for informal learning environments when there was very little opportunity for computational thinking education in formal education. These curricula and tools have been tremendously successful. Research has shown informal learning spaces to be effective at increasing awareness and engagement, changing perceptions of computing, and building self-efficacy, especially for students from underrepresented communities in computing [14, 24, 20, 23, 30].

While there has been tremendous success for some in the informal space, the goal of integrating into schools is to translate those successes to a much broader audience within the school day. Merely providing the same instruction within the school day may not result in the same outcomes. Providing access to computing curricula is just one part of the solution; CS/CT instruction must also be effective for diverse students. Four broad categories of students – students with disabilities, English language learners (ELL), students of color, and students in poverty – face academic challenges that may interfere with their success in a computing curriculum. It is time to revisit programming languages, programming environments, and curricula with the goal of equitable learning outcomes.

More recent work has shown strong correlations between overall school academic performance and learning [26] in a computer science curriculum built on open-ended projects designed using a Constructionist pedagogical approach [13]. This points to the need for scaffolding for some students.

The Use–>Modify–> Create [16] pedagogical approach has been proposed to provide additional support, adding a Use–>Modify task prior to an open-ended activity. In this Use–>Modify task, students learn by example code. They are provided with something that works to illustrate how to code a particular construct then are first asked to perform some small modification before tackling a more open-ended problem in which they apply that knowledge to their own project. This paper introduces TIPP&SEE, a metacognitive learning strategy that scaffolds student learning during a Use–>Modify activity. In this paper, we investigate the following overarching question: *Does our new strategy, TIPP&SEE, improve student learning of introductory CT concepts—events, sequence, & loops?* The contributions of this paper are to:

- introduce TIPP&SEE, a meta-cognitive learning strategy that scaffolds student learning of learn-by-example Scratch activities, and
- show that TIPP&SEE results in statistically-significant improvements in performance on most medium- and high-difficulty assessment questions.

In the next section, we present related work on learning strategies for CS/CT, followed by the theoretical framework grounding this research. In section 4, we provide a more in-depth description

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 2020, March 11-14, Portland, OR, USA
© Association for Computing Machinery.
ACM ISBN 978-1-4503-6793-6/20/03…$15.00
https://doi.org/10.1145/3328778.3366821

RIGHTSLINK

of TIPP&SEE, and in section 5, we describe our study. Question-level results are presented in section 6, followed by a discussion of their overall implications in section 7.

2 RELATED WORK

In this section, we present related work on learning strategies for CS/CT education more broadly, and in elementary education more specifically.

There is a wealth of research in CS/CT learning strategies at the university level, taking many forms and addressing various aspects of CS/CT. Lister et al. found that students who had better skills at reading, tracing, and explaining code tended to be better at writing code [19, 18]. Falkner et al. also identified self-regulated learning strategies such as diagramming and using design to understand a problem or code [10]. In addition to programming-specific skills, Bergin et al. identified cognitive (i.e. elaboration and organization), metacognitive (i.e. critical thinking), and resource management (i.e. effort regulation) strategies that were linked to success in a computing curriculum [3]. Game development was also found to be successful at promoting learning and engagement [1]. Nevertheless, the strategies for this age group are unlikely to directly translate to younger learners, who may not have the maturity necessary to regulate their own learning.

Due to the relatively recent push for elementary CS/CT instruction, prior work in learning strategies for this age group is sparse by comparison. Based on the idea that scaffolding increasingly deep interactions will promote the acquisition and development of CT, Lee et al developed the three-stage progression called Use–>Modify–>Create [16]. Their approach provides more scaffolded, guided instruction for each concept, followed by a more open-ended project to engage students' interest and creativity. Another strategy is called PRIMM, which stands for Predict-Run-Investigate-Modify-Make [28]. PRIMM guides teachers in creating scaffolded activities in text-based programming languages to encourage learning, especially for struggling students.

TIPP&SEE adds to this growing body of work in the following ways: (1) it provides additional scaffolding in the Use –> Modify step of Use –> Modify –> Create to better promote learning for struggling students, and (2) it is a learning strategy designed specifically for use with Scratch, a widely-used programming language at the elementary level [11].

3 THEORETICAL FRAMEWORK

3.1 CS Education Pedagogy

As with other subjects, including literacy [5, 29], computer science education researchers disagree on whether the best approach is to use open-ended, exploratory experiences or direct instruction. Papert [13], in his work on constructionism, posited that individuals learn best when they are constructing an artifact for public consumption, putting a premium on self-directed learning. This inspired Scratch to create a repository of projects which students can "remix" (copy and modify). Critics argue that open-ended exploration of such environments may not lead to immediate understanding of the concepts behind what they produce [4], especially compared to a more direct instruction approach [17]. On the other hand, an overly structured approach can dissuade students from continuing in programming courses, especially female students [32]. A more moderate approach is informed by seeking the Zone of Proximal Flow [2], a combination of Vygotsky's Zone of Proximal Development theory [31] with Csikszentmihalyi's ideas about Flow [9]. The Zone of Proximal Flow refers to learning experiences that are not too challenging as to overwhelm students, but not too easy as to lead to little learning. One such moderate approach is Use–>Modify–>Create [16], which TIPP&SEE extends to better support diverse learners.

3.2 Reading Comprehension Strategies

Learning to program relies heavily on reading comprehension at several stages in the learning process – reading (a) individual instructions, (b) a sequence of instructions provided as an example or starting code, (c) one's own partially-completed code, or (d) one's completed but incorrect code. Just as in reading, it is not enough to decode the letters into words; to succeed, the student needs to make meaning of the sequences of words into instructions (like sentences) and the sequences of instructions into functions or programs (like paragraphs). There are several existing evidence-based reading comprehension strategies that may have connections to programming, in particular previewing and text structure.

Previewing [15, 21] helps students set goals for reading and activates prior knowledge. When reading example code containing a new concept, students might scan the code to quickly identify familiar and unfamiliar concepts. They could think about their prior knowledge of the concepts, predict how the new concept might work, and inspect the syntax of the new concept.

Text structure [12, 33] prepares students to recognize disciplinary-specific text structures and use this knowledge to plan for reading and guide comprehension. In CS, programming languages and environments have specific structures that students must be able to discover to comprehend code and must be able to differentiate as they learn new languages and environments.

We drew from these reading comprehension strategies in designing TIPP&SEE, which will be discussed in more depth in the next section.

4 TIPP&SEE LEARNING STRATEGY

TIPP&SEE (Figure 1) is a learning strategy that scaffolds student exploration of provided programs for Use–>Modify activities. The strategy is specifically designed for use with Scratch, a popular programming language and development environment used in elementary schools [11].

Inspired by previewing strategies, the first half, *TIPP*, guides students in previewing different aspects of a new Scratch project before looking at any code. As a last step, they run the code with very deliberate observations of the events and actions that occur. The second half, *SEE*, draws from text structure strategies. *SEE* provides a roadmap for finding code in the Scratch interface (clicking on the sprite and finding the event) and proceduralizes the process by which they can learn how code works by methodical exploration (deliberate tinkering).

RIGHTSLINK

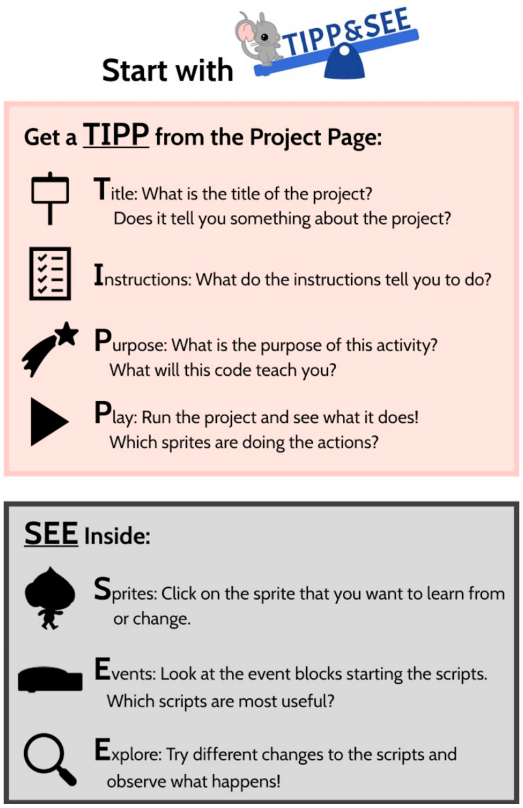

Figure 1: TIPP&SEE Learning Strategy

5 METHODS

5.1 Experimental Design

Fifteen teachers were recruited from a large, urban school district and underwent the same professional development to teach the Scratch Act 1 curriculum to 4th grade students (ages 9-10). A total of 16 classrooms participated in the study, including six of bilingual classrooms. Teachers were randomly assigned to either the TIPP&SEE or the control condition, resulting in five English-only and three bilingual classrooms in each condition. Treatment classrooms used TIPP&SEE worksheets, whereas control classrooms used worksheets that introduced the overall project and modify task without stepping students through the protocol. Lessons were taught by the teacher and assisted by an undergraduate CS student. After excluding students who switched schools or were chronically absent, there were a total of 96 and 88 students in the control and TIPP&SEE condition, respectively, for a total of 184 students.

5.2 Scratch Act 1

Within a semester (approximately 5 months), students completed Scratch Act 1 [27], an introductory computational thinking (CT) curriculum modified from the Creative Computing curriculum [7] consisting of three modules: Sequence, Events, and Loops. Each module begins with Use/Modify project(s) and culminates in a Create project (see Table 1). All curriculum materials and assessments were available in English and Spanish.

Module	Project	Use-Modify-Create
Sequence	Name Poem	Use/Modify
	Ladybug Scramble	Use/Modify
	5-Block Challenge	Create
Events	Events Ofrenda	Use/Modify
	About Me	Create
Loops	Build a Band	Use/Modify
	Interactive Story	Create

Table 1: Scratch Act 1 Modules

5.3 Assessments

Students took two pen-and-paper assessments, one each after Module 2 (events & sequence) and Module 3 (loops). Each 20-30 minute assessment consisted of multiple-choice, fill-in-the-blank and open response questions.

Assessment design was guided by the Evidence-Centered Design Framework [22]. Domain analysis was informed by the CS K-12 Framework and Rich et al's K-8 learning trajectories for elementary computing [25]. These overarching goals were narrowed in domain modeling to identify specific knowledge and skills desired.

The questions were designed by a team of CS and education researchers and practitioners. Questions were then evaluated by more practitioners and a reading comprehension expert, and tested with students from the previous school year for face validity.

In addition, item difficulty (P) and item discrimination (D) values were calculated for each question (see Tables **??** & **??**). Item difficulty is the proportion of students who answered the question correctly—the higher the item difficulty value, the easier the question is. Item discrimination is a measure of how well a question differentiates among students on the basis of how well they know the material being tested—the higher the item discrimination value is, the better the question is at differentiating students [8].

Cronbach's alpha (α) was also calculated for internal reliability between questions on the same topic. Between the questions and sub-questions on both assessments, there were 5 items on events (α=.72), 4 items on sequence (α=.7), and 9 items on loops (α=.85). A question with parallel loops was excluded in the reliability calculation because its inclusion lowered the the reliability of the loops questions (α=.82), suggesting that it was not testing the same concepts as the other questions. An understanding of the concept of parallelism, instead of loops, was likely more crucial to answering this question correctly.

5.4 Data Analysis

The assessments were graded by two researchers to ensure reliability. To see if TIPP&SEE had an influence on their assessment performance, the ANOVA F-test was used. The ANOVA F-test returns a p-value; for this study, $p < .05$ is statistically significant (see Table 2). To handle the imbalance between groups, Type 3 Sum of Squares was used. Free-response questions were qualitatively coded by two researchers with a Fleiss' Kappa inter-rater reliability of at least 80%.

The eta squared (η^2) effect size was also calculated. η^2 measures the proportion of the total variance in a dependent variable (DV) that is associated with the membership of different groups defined by an independent variable (IV) [6]. For example, if an IV has a η^2 of 0.25, that means that 25% of a DV's variance is associated with that IV.

RIGHTSLINK

Question	P	D	$F(1, 184)$	η^2
Sequence & Events Assessment				
Q1	.89	.27	4.76*	.025
Q2a	.65	.67	2.95	–
Q2b	.63	.65	2.86	–
Q3	.60	.29	7.27**	.038
Q4	.8	.4	3.95	–
Q5	.89	.37	2.87	–
Q6	.86	.69	6.86 **	.036
Q7a	.78	.41	10.93**	.056
Q7b	.81	.67	10.93**	.056
Loops Assessment				
Q1	.92	.48	7.92**	.042
Q2	.69	.55	17.26**	.087
Q3	.73	.61	36.9**	.17
Q4	.65	.48	25.9**	.13
Q5a	.92	.48	11.8**	.061
Q5b	.7	.65	13.8**	.071
Q5c	.7	.62	13.8**	.071
Q6a	.86	.19	7.49**	.039
Q6b	.41	.13	1.01	–
Q7	.84	.51	8.57**	.045
EC	.17	.37	3.59	–

* $p < .05$, ** $p < .01$

Table 2: Item Difficulty, Discrimination, & ANOVA F-Test

6 RESULTS

We begin with question-level results. There were 7 questions in the events & sequence assessment and 8 questions in the loops assessment. Overall results and implications are discussed in Section 7.

6.1 Events & Sequence

In the Events & Sequence assessment, Q1-3 asked about events, while Q4-6 asked about sequence. Q7 had two sub-questions; the first asked about events, and the second asked about sequence.

6.1.1 Q1-3: Events Triggering Scripts. Q1 asks students to identify the event that triggered one action block. Students in both conditions performed similarly well in Q1, with 94.3% of TIPP&SEE and 84.5% of the control students answering correctly. However, there was a statistically-significant difference in performance (Figure 2).

Q2 showed students a stage with two sprites saying different things after the green flag was clicked and asked which script ran for each sprite in two parts. Around 70% of TIPP&SEE students answered both parts correctly, compared with around 58% of the control students. However, this difference was not statistically significant.

Q3, a more advanced question, asks students to identify a multi-block script triggered by the `when sprite clicked` event. TIPP&SEE students also outperformed the control students on Q3, with 69.7% of them answering correctly, compared with 50.5% of the control students (ee Figure 2). Closer inspection of the responses revealed that the control students were more likely to select the options that started with `when green flag clicked`, the event that students were most familiar with (75% of the control students vs 40.66% of the TIPP&SEE students).

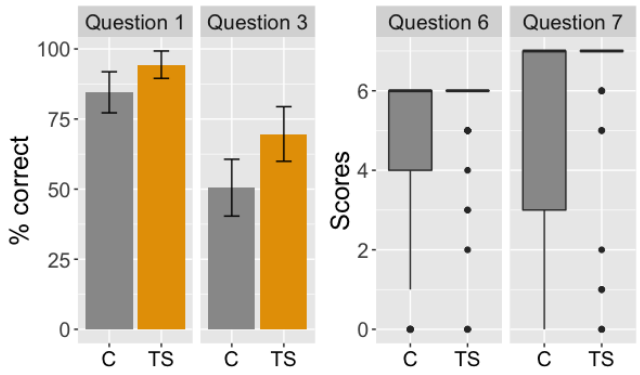

Figure 2: Events Q1 & Q3 & Sequence Q6 & Q7 Results (left to right)

6.1.2 Q4 & Q5 Sequence Basics. Q4 and Q5 cover a basic understanding of sequence, asking students to identify the last block in a sequence and the different orders of blocks in two scripts, respectively. Over 70% of students in both conditions answered Q4 and Q5 correctly with no statistically-significant difference in performance, suggesting that the curriculum sufficiently supports students in learning the easiest concepts without any additional scaffolding.

6.1.3 Q6 & Q7 Free-Response on Events & Sequence. Q6 and Q7 both asked students to explain a script of 3 blocks in their own words. Both questions had blank lines preceded by "First", "Next", and "Last" to scaffold their answers; each line was worth 2 points. Q6 was worth 6 points, while Q7 was worth 7 points because it had an additional question asking about the event worth 1 point. As shown in Figure 2, there were statistically-significant differences in performance on both Q6 & Q7 between the TIPP&SEE and control students, suggesting that the additional scaffolding provided by TIPP&SEE encouraged a deeper understanding of events & sequence.

Qualitative analysis further illuminated some patterns. TIPP&SEE students were less prone than the control students to respond with an incorrect sequence or missing blocks (Q6: 10.11% vs 18.75%; Q7: 11.24% vs 31.25%). They also provided more precise responses & were less likely to leave out the block name or, if applicable, an important parameter when describing blocks (Q6: 8.98% vs 27.08%; Q7: 10.12% vs 16.66%).

Overall, the data from the free-response questions show that students in the TIPP&SEE condition could demonstrate a more sophisticated understanding of the blocks themselves, as well as the CT concepts of events & sequence, than students in the control condition.

6.2 Loops

The loops assessment comprised of seven questions (Q1-7) and an extra credit question asking about nested loops, a concept not explicitly covered in Scratch Act 1.

6.2.1 Q1 & Q2 Loop Basics. Q1 showed students a loop and asked them how many times it would repeat. While students in both conditions performed well with 94.4% of TIPP&SEE & 84.5% of control students answering correctly, there was still a statistically-significant difference (Figure 3).

RIGHTSLINK

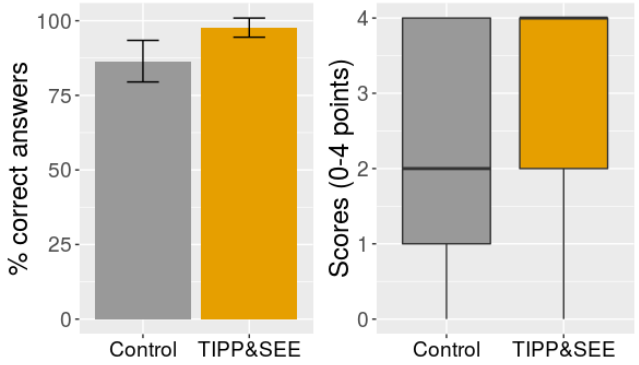

Figure 3: Loop Basics Q1 (left) & Q2 (right) Results

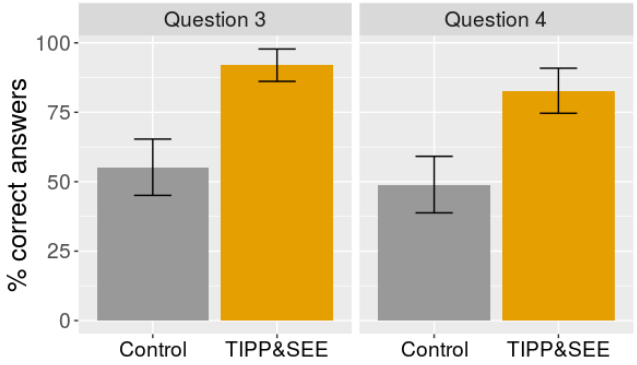

Figure 4: Loop Unrolling Q3 (left) & Q4 (right) Results

In Q2, students were shown 4 code snippets and asked which snippet would cause the sprite to change costumes 3 times. One code snippet was inspired by a common misconception observed by teachers where students would wrap repeated blocks with a repeat loop that had the same number of iterations as the number of blocks. There were 2 correct answers; students received 2 points for each correct answer and lost 1 point for each wrong answer for a maximum of 4 points. TIPP&SEE students outperformed control students with mean of 3.2 points compared with 2.4 points (Figure 3). Students in the control condition were more than twice as likely as the students in the TIPP&SEE condition to choose the common misconception option (43.75% vs 18.39%), supporting the observations made by the teachers.

6.2.2 Q3 & Q4 Loop Unrolling. Q3 & Q4 both asked students to unroll a `repeat 3` loop with 2 blocks, but with different answer choices. Q3 showed the blocks in the loop repeated 1,2,3 and 4 times, while Q4 had a "split loop" option – where the first block was repeated 3 times followed by the second block repeated 3 times.

For Q3, 91.9% of TIPP&SEE students answered correctly, compared with only 55.2% of the control students (Figure 4).

Similarly, 82.8% of TIPP&SEE students answered Q4 correctly, in comparison to only 48.9% of the control students (see Figure 4). An analysis of common mistakes revealed that the TIPP&SEE students who answered incorrectly tended to choose responses that suggested a closer, but flawed, understanding of loops – 14.94% of TIPP&SEE students chose the "split loop" option, compared with 12.5% of the control students. In contrast, 31.25% of the control students selected the option where the blocks were repeated only once, compared with only 2.29% of the TIPP&SEE students.

Taking Q1-4 in perspective, we find that the control students displayed a more superficial understanding of loop functionality compared with the TIPP&SEE students. While many control students were able to answer the simplest question on repeat iteration count, they struggled with the more advanced questions on loop unrolling.

6.2.3 Q5 Loops within a Sequence. Q5 showed a script with a loop and asked 3 sub-questions: (a) code *in*, (b) *before*, and (c) *after* the loop. Part (a) was worth 4 points—students earned 2 points for each correct block circled and lost 1 point for each incorrect block circled; parts (b) and (c) were worth 1 point.

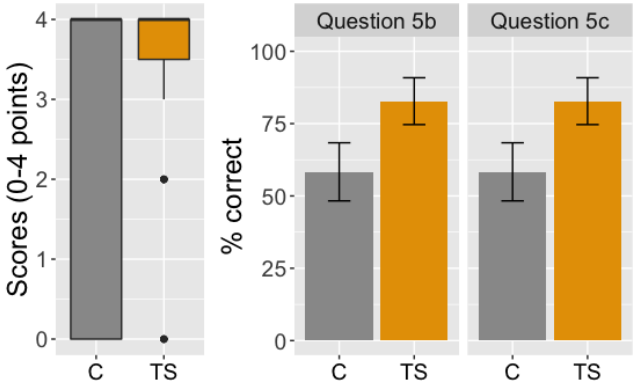

Figure 5: Loops in Sequence Q5 Results

TIPP&SEE students outperformed the control students on all three parts, as shown in Figure 5. On part (a), TIPP&SEE students scored a mean of 3.29 points, while the control students scored a mean of 2.51 points. On parts (b) and (c), 82.8% of the TIPP&SEE students answered correctly, compared with only 58.3% of the control students.

6.2.4 Q6 Parallel Loops. Q6 asked students about the execution of two sprites' code: part (a) asked about a sprite with sequential loops, while part (b) asked about a sprite with two loops in parallel.

For part (a), 93.1% of the TIPP&SEE students were able to correctly identify the sequential behavior, as opposed to 79.2% of the control students. In contrast, students in both conditions struggled with part (b) with only 44.8% of TIPP&SEE and 37.5% control students answering correctly, suggesting the difficulty of parallel execution for this age group. Results for both parts are shown on Figure 6.

6.2.5 Q7 Free-Response on Loops. Q7, the free-response question in the loops assessment, was similarly scaffolded to the ones in the events & sequence assessment. In addition to the blank lines preceded by the "First", "Next", and "Last", there was a blank where students would fill in how many times the loop would repeat. This blank was worth 1 point, while the other three blank lines were worth 2 points each, for a total of 7 points.

As indicated on Figure 7, TIPP&SEE students scored better on this question with a mean of 6.36 points, in contrast to the mean of 5.44 points earned by the control students.

RIGHTSLINK

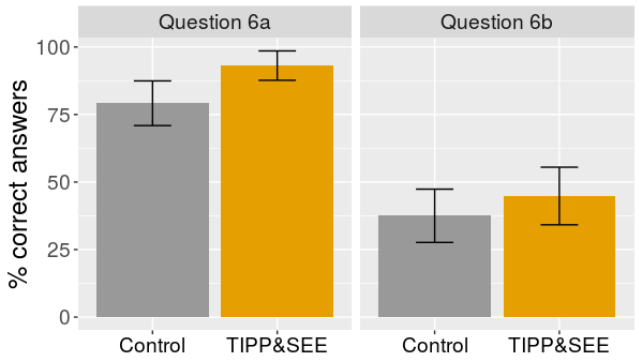

Figure 6: Parallel Loops Q6 Results

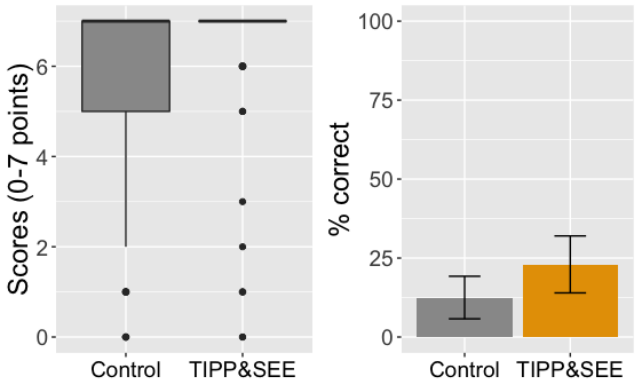

Figure 7: Loops Q7 & Extra Credit Results

From qualitatively analyzing responses, we found that control students demonstrated more fundamental misconceptions of loops—they were 4 times more likely to write the wrong number of iterations, list an incorrect sequence of the blocks within the loop, or leave out a block entirely (40.62% vs 9.19%). By comparison, the misconceptions of the TIPP&SEE students were more nuanced. 12.64% of TIPP&SEE students provided responses that described individual blocks in the loop repeated in sequence, instead of the entire loop repeated, compared with 10.42% of the control students.

6.2.6 Extra Credit Nested Loops. The final question on the loops assessment was an Extra Credit (EC) question and asked about nested loops, a topic not explicitly covered in the curriculum. Unsurprisingly, students in both conditions similarly struggled with this question (see Figure 7). The TIPP&SEE students performing slightly better with 22.9% of them answering correctly, compared with 12.5% of the control students. This difference, however, was not statistically significant.

7 DISCUSSION

We now revisit our key research question: *How does TIPP&SEE influence student learning of introductory CT concepts—events, sequence, & loops?*

Our findings show that students using TIPP&SEE outperformed students who used an unmodified Use –> Modify –> Create approach on nearly all questions of moderate and hard difficulty. TIPP&SEE students outperformed the control students in all but

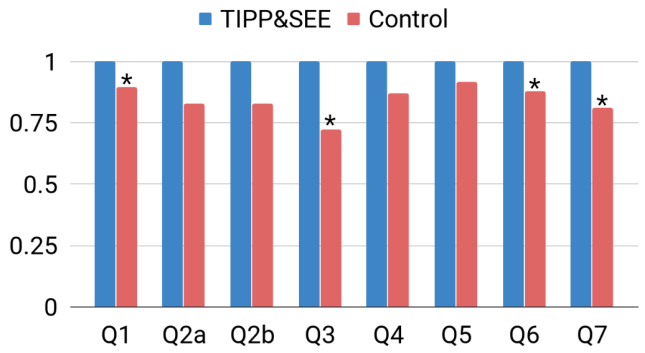

Figure 8: Events & Sequence Assessment Results

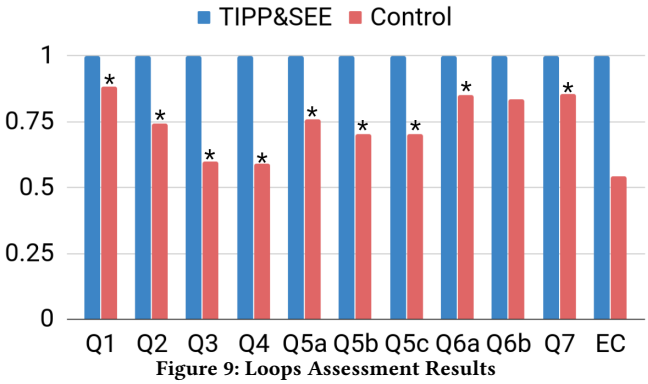

Figure 9: Loops Assessment Results

the most basic questions on the events & sequence assessment (Figure 8; asterisks denote significance). Most students were able to demonstrate a simple understanding of events & sequence with just the scaffolding provided by Use –> Modify –> Create, but with TIPP&SEE, they could demonstrate a more sophisticated understanding.

On the loops assessment, the TIPP&SEE students performed better than the control students in almost all questions; only parallelism and nested loops (which was not explicitly covered in the curriculum) were beyond their grasp (Figure 9). This suggests that while students are able to make significant learning gains with TIPP&SEE, there is still room for improvement in the instruction of parallelism.

As momentum continues to build for integrating CS/CT into elementary school classrooms, it is imperative that CS/CT instruction be effective for diverse learners. A learning strategy like TIPP&SEE provides some much-needed scaffolding for such diverse learners, advancing not just equitable access, but also equitable outcomes in elementary computing.

8 LIMITATIONS & FUTURE WORK

The students, teachers, and schools in this study were not randomly sampled throughout the school district, nor school districts nationwide or worldwide. In the future, a large-scale replication TIPP&SEE in more school districts would provide further data on its effectiveness.

9 ACKNOWLEDGEMENTS

This project was funded by National Science Foundation (NSF) Grant No. 1660871 and DGE-1746045.

RIGHTSLINK

REFERENCES

[1] Tiffany Barnes et al. "Game2Learn: building CS1 learning games for retention". In: *ACM SIGCSE Bulletin* 39.3 (2007), pp. 121–125.

[2] Ashok R Basawapatna et al. "The zones of proximal flow: guiding students through a space of computational thinking skills and challenges". In: *Proceedings of the ninth annual international ACM conference on International computing education research*. ACM. 2013, pp. 67–74.

[3] Susan Bergin, Ronan Reilly, and Desmond Traynor. "Examining the role of self-regulated learning on introductory programming performance". In: *Proceedings of the first international workshop on Computing education research*. ACM. 2005, pp. 81–86.

[4] John B Biggs and Kevin F Collis. *Evaluating the quality of learning: The SOLO taxonomy (Structure of the Observed Learning Outcome)*. Academic Press, 2014.

[5] Susan Chambers Cantrell. "Effective teaching and literacy learning: A look inside primary classrooms". In: *The Reading Teacher* 52.4 (1998), pp. 370–378.

[6] Jacob Cohen. *Statistical power analysis for the behavioural sciences*. 1988.

[7] Creative Computing. *An introductory computing curriculum using Scratch*.

[8] Linda Crocker and James Algina. *Introduction to classical and modern test theory*. ERIC, 1986.

[9] Mihaly Csikszentmihalyi, Sami Abuhamdeh, and Jeanne Nakamura. "Flow". In: *Flow and the foundations of positive psychology*. Springer, 2014, pp. 227–238.

[10] Katrina Falkner, Rebecca Vivian, and Nickolas JG Falkner. "Identifying computer science self-regulated learning strategies". In: *Proceedings of the 2014 conference on Innovation & technology in computer science education*. ACM. 2014, pp. 291–296.

[11] Louise P Flannery et al. "Designing ScratchJr: support for early childhood learning through computer programming". In: *Proceedings of the 12th International Conference on Interaction Design and Children*. ACM. 2013, pp. 1–10.

[12] Russell Gersten et al. "Teaching reading comprehension strategies to students with learning disabilities: A review of research". In: *Review of educational research* 71.2 (2001), pp. 279–320.

[13] Idit Ed Harel and Seymour Ed Papert. *Constructionism*. Ablex Publishing, 1991.

[14] Yasmin Kafai et al. "Ethnocomputing with electronic textiles: culturally responsive open design to broaden participation in computing in American indian youth and communities". In: *Proceedings of the 45th ACM technical symposium on Computer science education*. ACM. 2014, pp. 241–246.

[15] Janette K Klingner and Sharon Vaughn. "Using collaborative strategic reading". In: *Teaching exceptional children* 30.6 (1998), pp. 32–37.

[16] Irene Lee et al. "Computational thinking for youth in practice". In: *Acm Inroads* 2.1 (2011), pp. 32–37.

[17] Michael J Lee and Andrew J Ko. "Comparing the effectiveness of online learning approaches on CS1 learning outcomes". In: *Proceedings of the eleventh annual international conference on international computing education research*. ACM. 2015, pp. 237–246.

[18] Raymond Lister, Colin Fidge, and Donna Teague. "Further evidence of a relationship between explaining, tracing and writing skills in introductory programming". In: *Acm sigcse bulletin* 41.3 (2009), pp. 161–165.

[19] Raymond Lister et al. "A multi-national study of reading and tracing skills in novice programmers". In: *ACM SIGCSE Bulletin*. Vol. 36. 4. ACM. 2004, pp. 119–150.

[20] John H Maloney et al. *Programming by choice: urban youth learning programming with scratch*. Vol. 40. 1. ACM, 2008.

[21] Suzanne Liff Manz. "A strategy for previewing textbooks: teaching readers to become THIEVES.(Teaching Ideas)". In: *The Reading Teacher* 55.5 (2002), pp. 434–436.

[22] Robert J Mislevy and Geneva D Haertel. "Implications of evidence-centered design for educational testing". In: *Educational Measurement: Issues and Practice* 25.4 (2006), pp. 6–20.

[23] Lijun Ni et al. "Computing with a community focus: outcomes from an app inventor summer camp for middle school students". In: *Journal of Computing Sciences in Colleges* 31.6 (2016), pp. 82–89.

[24] Lori Pollock et al. "Increasing high school girls' self confidence and awareness of CS through a positive summer experience". In: *ACM SIGCSE Bulletin*. Vol. 36. 1. ACM. 2004, pp. 185–189.

[25] 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*. ACM. 2017, pp. 182–190.

[26] Jean Salac et al. "An Analysis through an Equity Lens of the Implementation of Computer Science in K-8 Classrooms in a Large, Urban School District". In: *Proceedings of the 50th ACM Technical Symposium on Computer Science Education*. ACM. 2019, pp. 1150–1156.

[27] *Scratch Act 1*. URL: https://www.canonlab.org/scratchact1modules.

[28] Sue Sentance, Jane Waite, and Maria Kallia. "Teachers' Experiences of using PRIMM to Teach Programming in School". In: *Proceedings of the 50th ACM Technical Symposium on Computer Science Education*. ACM. 2019, pp. 476–482.

[29] Keith Topping and Nancy Ferguson. "Effective literacy teaching behaviours". In: *Journal of Research in Reading* 28.2 (2005), pp. 125–143.

[30] Timothy Urness and Eric D Manley. "Generating interest in computer science through middle-school Android summer camps". In: *Journal of Computing Sciences in Colleges* 28.5 (2013), pp. 211–217.

[31] Lev Vygotsky. "Interaction between learning and development". In: *Readings on the development of children* 23.3 (1978), pp. 34–41.

[32] David C Webb, Alexander Repenning, and Kyu Han Koh. "Toward an emergent theory of broadening participation in computer science education". In: *Proceedings of the 43rd ACM technical symposium on Computer Science Education*. ACM. 2012, pp. 173–178.

[33] Joanna P Williams. "Instruction in reading comprehension for primary-grade students: A focus on text structure". In: *The Journal of Special Education* 39.1 (2005), pp. 6–18.

RIGHTSLINK