TIPP&SEE: A Previewing & Navigating Strategy for Use/Modify Scratch Activities

Anonymous Authors

Abstract

With many school districts nationwide integrating Computer Science (CS) and Computational Thinking (CT) instruction at the K-8 level, it is crucial that CS instruction be effective for diverse learners. A popular pedagogical approach 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, little research has gone into strategies that increase learning during the Use→Modify step. This paper introduces TIPP&SEE, a learning that further scaffolds student learning during this step. Results from a quasi-experimental study show statistically-significant outperformance from students using the TIPP&SEE strategy on all assessment questions of medium and hard difficulty, suggesting its potential as an effective CS learning strategy.

Keywords: learning strategy, computational thinking, Scratch, elementary education

1 Objectives

Momentum has been building for integrating computer science into elementary school classrooms. Providing access to computing curricula is just one part of the solution. It is critical that computer science instruction be effective for diverse students. Four broad categories of students; students with disabilities, English language learners (ELL), students of color, and students in poverty, have academic challenges that may interfere with their success in a computing curriculum. Diverse learners significantly underperform white peers with higher socio-economic status on important academic markers (Eric A Hanushek, n.d.; *NAEP Nations Report Card*, n.d.; *NAEP Nations Report Card - National Assessment of Educational Progress - NAEP*, n.d.). Worse, Michelmore and Dynarski showed that performance gaps grow depending on the number of years spent in poverty (Michelmore & Dynarski, 2016).

Unfortunately, there is a strong correlation between students of color and poverty, and performance in public schools is similar for students of color and those in poverty. More recent work by [citation redacted] has shown strong correlations between overall school academic performance and learning in a computer science curriculum built on open-ended projects designed using a Constructionist pedagogical approach (Harel & Papert, 1991).

Some students will need more scaffolding in their learning of CT concepts than many current curricula provide. The Use→Modify→ Create pedagogical approach has been proposed to provide additional support, adding a Use→Modify task prior to an open-ended activity (I. Lee et al., 2011). This paper introduces TIPP&SEE, a learning strategy that scaffolds student learning during the Use→Modify step of Use→Modify→Create.

In a quasi-experimental study, a set of classrooms using the TIPP&SEE strategy performed statistically-significantly better on all questions of medium and hard difficulty on written assessments as compared to classrooms use an unmodified Use→Modify→Create approach.

2 Theoretical Framework

2.1 CS Education Pedagogy

As with other subjects, including literacy, computer science education researchers disagree on whether the best approach is to use open-ended, exploratory experiences or direct instruction (Cantrell, 1998; Topping & Ferguson, 2005). Papert, 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 (Harel & Papert, 1991). 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, especially compared to a more direct instruction approach (Biggs & Collis, 2014; M. J. Lee & Ko, 2015). On the other hand, an overly structured approach can dissuade students from continuing in programming courses, especially female students (Webb, Repenning, & Koh, 2012). A more moderate approach is informed by seeking the Zone of Proximal Flow, a combination of Vygotsky's Zone of Proximal Development theory with Csikszentmihalyi's ideas about Flow (Basawapatna, Repenning, Koh, & Nickerson, 2013; Vygotsky, 1978; Csikszentmihalyi, Abuhamdeh, & Nakamura, 2014). 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 which provides more scaffolded, guided instruction for each concept, followed by a more open-ended project to engage students' interest and creativity (I. Lee et al., 2011). However, little research has gone into strategies to increase learning that occurs during the Use→Modify steps.

2.2 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).

We draw from existing evidence-based reading comprehension strategies in designing TIPP&SEE – previewing and text structure.

Previewing helps students set goals for reading and activates prior knowledge (Klingner & Vaughn, 1998; Manz, 2002). 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 prepares students to recognize disciplinary-specific text structures and use this knowledge to plan for reading and guide comprehension (Gersten, Fuchs, Williams, & Baker, 2001; Williams, 2005). 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.

3 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 (Flannery et al., 2013). In Scratch, students program actions for sprites (i.e. characters on the screen) using visual code blocks; a group of blocks is called a script (see Figure 2).

Inspired by the 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 observation 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.

4 Methods

4.1 Experimental Design

15 teachers were recruited from a large, urban school district and underwent the same professional development to teach the Scratch Act 1 curriculum and the TIPP&SEE learning strategy. A total of 16 classrooms participated in the study, three of which were co-taught by two teachers and six of which were bilingual classrooms. Each classroom was assisted by an undergraduate CS researcher. Classrooms were randomly assigned to either the TIPP&SEE or the control condition, with five English-only and three bilingual classrooms in each condition. Classrooms in the control condition were taught Scratch Act 1 without the TIPP&SEE worksheets guiding them through the Use/Modify projects. After excluding students who switched schools, were chronically absent or took the Spanish version of the assessments, there were a total of 56 and 88 students in the control and TIPP&SEE condition respectively, for a total of 144 students in the study.

4.2 Scratch Act 1

Within a semester, students completed Scratch Act 1 (*Scratch Act 1*, n.d.), an introductory computational thinking (CT) curriculum modified from the Creative Computing curriculum (Computing, n.d.). Scratch Act 1 consists of three modules, one for each of the key CT concepts (sequence, events, and loops). Each module used Use/Modify projects to introduce the CT concept, and culminated in a Create project (see Table 1). All curriculum materials were available in both English and Spanish.

4.3 Assessments

Students took two pen-and-paper assessments, the first one after the completion of Module 2 (events) and the second one after the completion of Module 3 (loops). Each assessment consisted of a mix of multiple-choice, fill-in-the-blank and open response questions, and were designed to take 20-30 minutes to complete.

Following the Evidence-Centered Design framework (Mislevy & Haertel, 2006), assessments were designed based on K-8 learning trajectories for elementary computing (Rich, Strickland, Binkowski, Moran, & Franklin, 2017). Questions were evaluated by a team of CS and education researchers and practitioners, and tested with students from the previous school year for face validity.

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. To handle the imbalance between groups, Type I Sum of Squares was used because there is only one factor tested.

5 Evidence & Results

For comparison across questions with different point values, the summary graphs discussed in this section present normalized scores, where the scores from the control condition are normalized proportional to the scores from the TIPP&SEE condition. Asterisks denote statistical significance at $p = .05$.

5.1 Events & Sequence

In the Events & Sequence assessment, Q1-2 ask about events, while Q3-7 ask about sequence. The students in the TIPP&SEE condition outperformed the students in the control condition in all but the most basic questions (see Figure 3).

Q1 asks students to identify the event that triggered one action block, while Q2, the more advanced question, asks students to identify the script that is triggered by an event. Students in both conditions performed similarly well in Q1, with 94.3% of TIPP&SEE and 85.7% of the control students answering correctly ($F(1, 142) = 3.11; p = .08$). In contrast, TIPP&SEE students outperformed the control students

on Q2, with 70.5% of them answering correctly, compared with 50% of the control students ($F(1,142) = 6.29; p = 0.013$).

Q3 and Q4 test 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 80% of students in both conditions answered Q3 and Q4 correctly (Q3: $F(1,142) = 0.913; p = .34$, Q4: $F(1,142) = 2.0839; p = .15$).

By comparison, the TIPP&SEE students demonstrated a deeper understanding of sequence. They outperformed the control students on a question on parallel scripts (Q5a: $F(1,142) = 5.98; p = .016$, Q5b: $F(1,142) = 5.24; p = .024$) and both the free-response questions – one asking only about sequence (Q6: $F(1,142) = 11.08; p < .01$), and the other combining events and sequence (Q7a: $F(1,142) = 11.5; p < .01$, Q7b: $F(1,142) = 9.81; p < .01$).

5.2 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. TIPP&SEE students outperformed the control students in all but one question in loops, a more advanced topic than events and sequence.

The only question where TIPP&SEE and control students performed similarly was Q6b, a question asking about loops executing in parallel ($F(1,139) = 3.26; p = .07$). Students in both conditions struggled with Q6b, with only 44.2% of TIPP&SEE and 29.1% of control students answering correctly. In contrast, 93.1% and 80% of TIPP&SEE and control students correctly answered Q6a, which asked about loops executing sequentially ($F(1,139) = 5.49; p = .02$). This performance difference between the two question parts indicates room for improvement in the instruction of parallelism.

On the rest of the questions, TIPP&SEE students displayed a better understanding of loops compared with the control students. They outperformed the control students on the basic questions, which asked to count the number of loop iterations (Q1: $F(1,139) = 15.63; p < .01$) and to unroll a loop (Q2: $F(1,139) = 54.34; p < .01$, Q3: $F(1,139) = 22.25; p < .01$), as well as the advanced questions, which asked about repeat blocks vs iterations (Q4: $F(1,139) = 18.27; p < .01$), loops within a sequence (Q5a: $F(1,139) = 14.08; p < .01$, Q5b: $F(1,139) = 12.12; p < .01$, Q5c: $F(1,139) = 15.65; p < .01$), and to explain a loop in their own words (Q7: $F(1,139) = 12.29; p < .01$).

6 Scholarly Significance

In this paper, we present TIPP&SEE, a learning strategy to provide additional scaffolding in the Use → Modify step of the Use → Modify → Create pedagogical approach in CS. Our findings show students using TIPP&SEE statistically-significantly outperforming students who used an unmodified Use → Modify → Create approach on nearly all questions of medium and hard difficulty. TIPP&SEE students outperformed the control students in all but the most basic questions on the sequence assessment, and all but a question on parallel loops on the loops assessment.

The results of the study suggest TIPP&SEE's potential as an effective CS learning strategy that can be used in elementary computing classes. Future work would include replicating TIPP&SEE in various school districts nationwide to test its effectiveness. As momentum continues to build for integrating CS into elementary school classrooms, it is imperative that CS instruction be effective for diverse learners. A learning strategy like TIPP&SEE provide some much-needed scaffolding for such diverse learners, advancing not just equitable access, but also equitable outcomes in elementary CS.

References

Basawapatna, A. R., Repenning, A., Koh, K. H., & Nickerson, H. (2013). 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* (pp. 67–74).

Biggs, J. B., & Collis, K. F. (2014). *Evaluating the quality of learning: The solo taxonomy (structure of the observed learning outcome)*. Academic Press.

Cantrell, S. C. (1998). Effective teaching and literacy learning: A look inside primary classrooms. *The Reading Teacher*, *52*(4), 370–378.

Computing, C. (n.d.). *An introductory computing curriculum using scratch.*

Csikszentmihalyi, M., Abuhamdeh, S., & Nakamura, J. (2014). Flow. In *Flow and the foundations of positive psychology* (pp. 227–238). Springer.

Eric A Hanushek, L. M. T. L. W., Paul E Peterson. (n.d.). *The unwavering ses achievement gap: Trends in us student performance.* Retrieved from https://www.hks.harvard.edu/publications/unwavering-ses-achievement-gap-trends-us-student-performar

Flannery, L. P., Silverman, B., Kazakoff, E. R., Bers, M. U., Bontá, P., & Resnick, M. (2013). Designing scratchjr: support for early childhood learning through computer programming. In *Proceedings of the 12th international conference on interaction design and children* (pp. 1–10).

Gersten, R., Fuchs, L. S., Williams, J. P., & Baker, S. (2001). Teaching reading comprehension strategies to students with learning disabilities: A review of research. *Review of educational research*, *71*(2), 279–320.

Harel, I. E., & Papert, S. E. (1991). *Constructionism.* Ablex Publishing.

Klingner, J. K., & Vaughn, S. (1998). Using collaborative strategic reading. *Teaching exceptional children*, *30*(6), 32–37.

Lee, I., Martin, F., Denner, J., Coulter, B., Allan, W., Erickson, J., ... Werner, L. (2011). Computational thinking for youth in practice. *Acm Inroads*, *2*(1), 32–37.

Lee, M. J., & Ko, A. J. (2015). Comparing the effectiveness of online learning approaches on cs1 learning outcomes. In *Proceedings of the eleventh annual international conference on international computing education research* (pp. 237–246).

Manz, S. L. (2002). A strategy for previewing textbooks: teaching readers to become thieves.(teaching ideas). *The Reading Teacher*, *55*(5), 434–436.

Michelmore, K., & Dynarski, S. (2016). *The gap within the gap: Using longitudinal data to understand income differences in student achievement* (Tech. Rep.). National Bureau of Economic Research.

Mislevy, R. J., & Haertel, G. D. (2006). Implications of evidence-centered design for educational testing. *Educational Measurement: Issues and Practice*, *25*(4), 6–20.

Naep nations report card. (n.d.). Retrieved from https://nces.ed.gov/nationsreportcard/glossary.aspxbasic

Naep nations report card - national assessment of educational progress - naep. (n.d.). Retrieved from https://nces.ed.gov/nationsreportcard/?src=ft

Rich, K. M., Strickland, C., Binkowski, T. A., Moran, C., & Franklin, D. (2017). K-8 learning trajectories derived from research literature: Sequence, repetition, conditionals. In *Proceedings of the 2017 acm conference on international computing education research* (pp. 182–190).

Scratch act 1. (n.d.). Retrieved from https://www.canonlab.org/scratchact1modules

Topping, K., & Ferguson, N. (2005). Effective literacy teaching behaviours. *Journal of Research in Reading*, *28*(2), 125–143.

Vygotsky, L. (1978). Interaction between learning and development. *Readings on the development of children*, *23*(3), 34–41.

Webb, D. C., Repenning, A., & Koh, K. H. (2012). Toward an emergent theory of broadening participation in computer science education. In *Proceedings of the 43rd acm technical symposium on computer science education* (pp. 173–178).

Williams, J. P. (2005). Instruction in reading comprehension for primary-grade students: A focus on text structure. *The Journal of Special Education*, *39*(1), 6–18.

Tables & Figures

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.

SEE Inside:

Sprites: Click on the sprite that you want to learn from or change.

Events: Look at the event blocks starting the scripts. Which scripts are most useful?

Explore: Try different changes to the scripts and observe what happens!

Figure 1: TIPP&SEE Learning Strategy

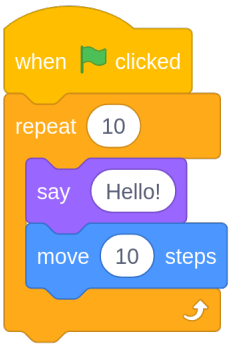

Figure 2: Script in Scratch made of code blocks

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

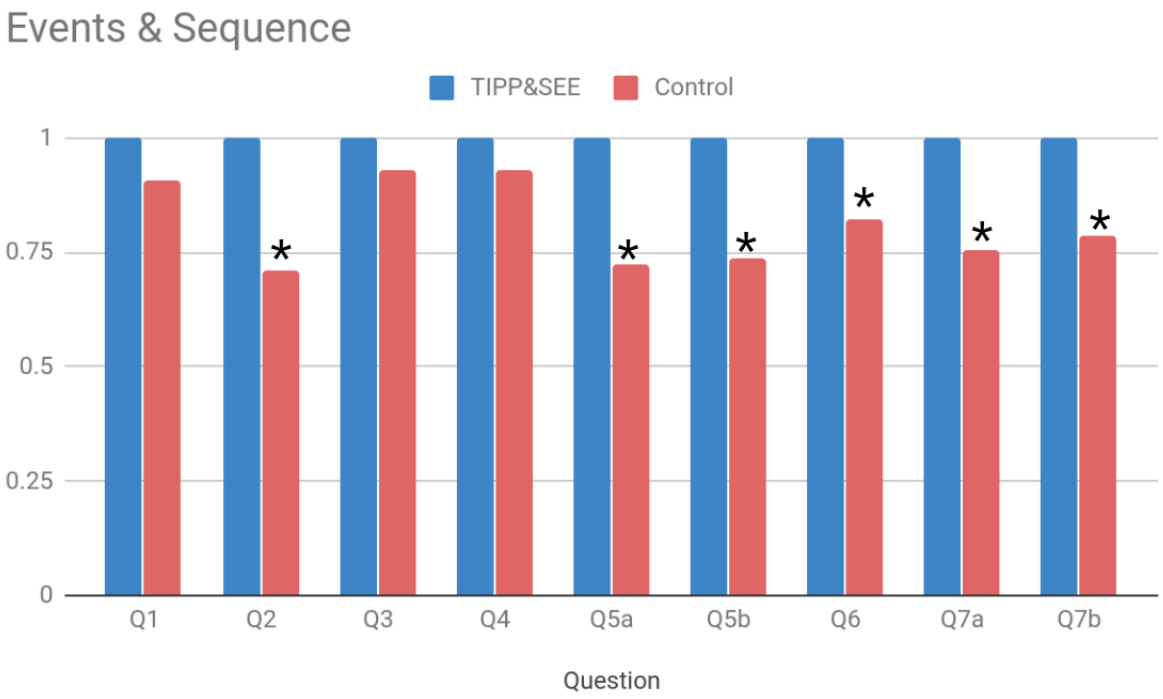

Figure 3: Events & Sequence Assessment Results

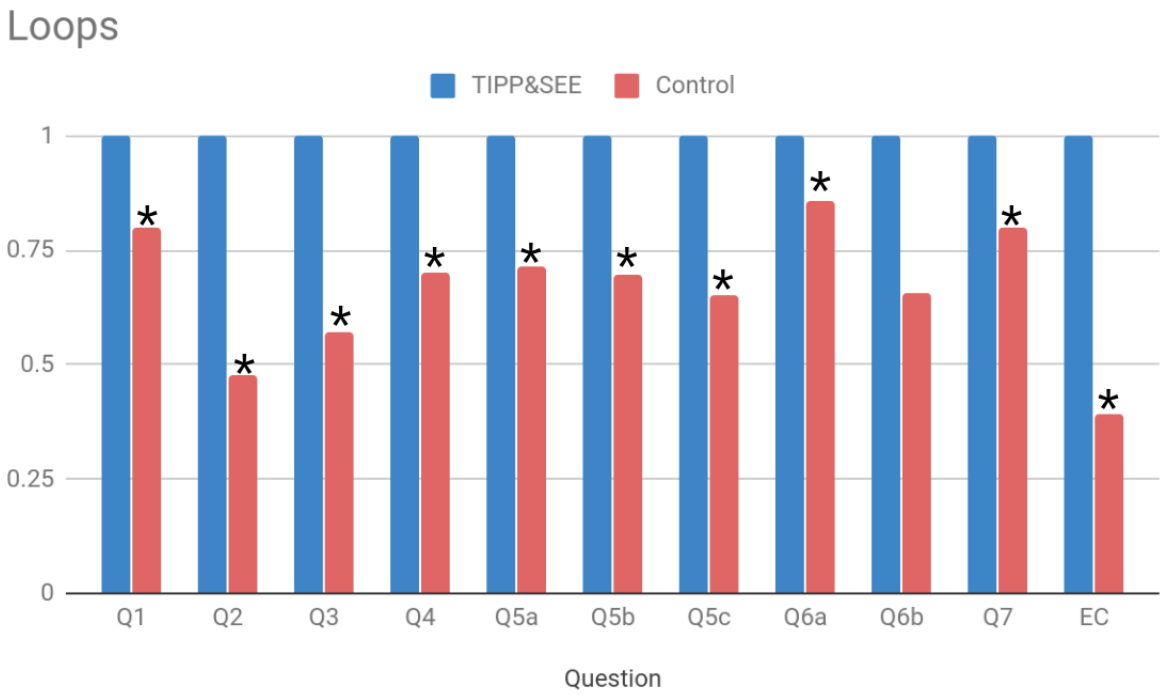

Figure 4: Loops Assessment Results