Exploring Student Behavior Using the TIPP&SEE Learning Strategy

Diana Franklin, Jean Salac, Zachary Crenshaw, Saranya Turimella, Zipporah Klain, Marco Anaya, & Cathy Thomas*
University of Chicago, Chicago, IL, USA
*Texas State University, San Marcos, TX, USA
{dmfranklin,salac,zcrenshaw,sturimella,zklain,manaya}@uchicago.edu;thomascat@txstate.edu

ABSTRACT

With the rise of Computational Thinking (CT) instruction at the elementary level, it is imperative for elementary computing instruction to support a variety of learners. TIPP&SEE is a meta-cognitive learning strategy that scaffolds student learning when learning from example code. Results from a previous study show statistically-significant performance differences favoring students using the TIPP&SEE strategy on a written assessment [43].

In this work, our goal is gain insight as to *why* such dramatic learning differences may have occurred. We analyze the students' computational artifacts and TIPP&SEE worksheets. Artifact analysis reveals that students in the TIPP&SEE group are more thorough in their work, completing more elements of the required tasks. In addition, they build open-ended projects with longer scripts that utilize more learned blocks. Worksheet analysis shows that students were highly accurate on some types of questions but largely skipped others. Despite these positive behaviors, there was little statistical correlation between student worksheet correctness, project completion, and written assessment performance. Therefore, while students in the TIPP&SEE group performed actions we believe lead to more success, no individual actions directly explain the results. Like other meta-cognitive strategies, the value of TIPP&SEE may lie in cognitive processes not directly observable, and may vary based upon individual student differences.

KEYWORDS

learning strategy; computational thinking; Scratch; elementary education

ACM Reference Format:
Diana Franklin, Jean Salac, Zachary Crenshaw, Saranya Turimella,, Zipporah Klain, Marco Anaya, & Cathy Thomas*. 2020. Exploring Student Behavior Using the, TIPP&SEE Learning Strategy. In *Proceedings of the 2020 International Computing Education Research Conference (ICER '20), August 10–12, 2020, Virtual Event, New Zealand*. ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3372782.3406257

1 INTRODUCTION

With the introduction of computer science (CS) instruction at the upper elementary school level (ages 8-12) in an increasing number of countries, such as Israel, India, and the United States [22], a broad population, that may not have had such opportunities previously, are getting access to CS. However, access is necessary but not sufficient for equity. As gains are made in access, it is crucial that we better understand how broad populations respond to different pedagogical approaches.

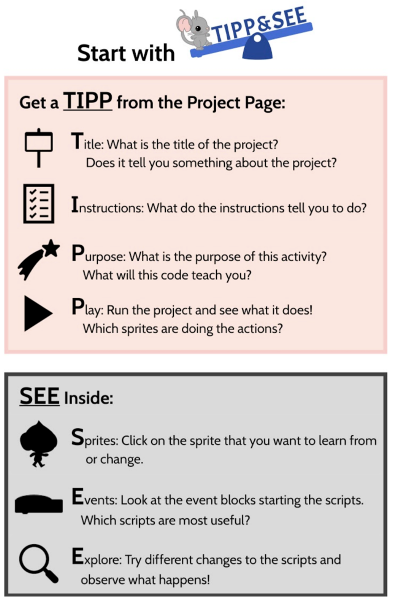
There are popular curricula that follow several different pedagogical approaches for young learners (ages 8-12). Code.org's curriculum features structured, puzzle-based instruction, with a series of carefully-chosen puzzles in increasing difficulty that focus on a single programming construct at once. On the other side of the spectrum, the Creative Computing Curriculum takes a Constructionist approach, with every activity having broad goals (e.g. introducing oneself) starting from a blank page.

The Use−>Modify−>Create [29] approach represents a middle ground, trading off between more structured activities (Use−>Modify tasks starting with example code) and open-ended activities (Create). In the Use−>Modify task, students are first provided with a functional example to explore how the construct works. They then perform some small modification to the example before tackling a more open-ended problem in which they apply this new knowledge to their own project.

Previous research introduced TIPP&SEE, a learning strategy for use during Use−>Modify activities, and showed that students learn computational thinking concepts better when using the TIPP&SEE learning strategy. In this paper, our goal is to explore *why* this might be. In particular, we ask three research questions:

- To what degree do students follow the TIPP&SEE protocol, and how accurately can they answer the posed questions?
- How does using the TIPP&SEE learning strategy affect student behavior during the completion of Use−>Modify and Create tasks?
- Are there any statistical correlations between behavior on the TIPP&SEE worksheets or project attributes and written assessment performance?

In the following section, we describe the TIPP&SEE learning strategy and the dramatic out-performance from students using this strategy that motivated this study. In section 3, we outline the different theories that ground the design of our curriculum, strategy, and study. In section 4, we delineate research that this study builds upon and in section 5, we detail our methodology. We follow with a presentation of our results in section 6 and discuss them in section


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! Which sprites are doing the actions?

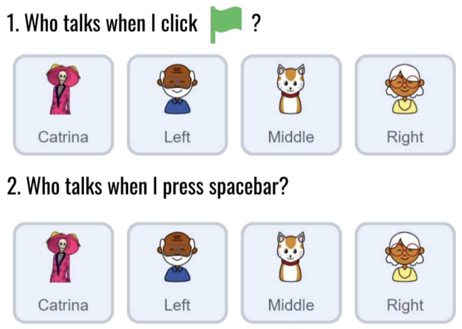
SEE Inside:

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

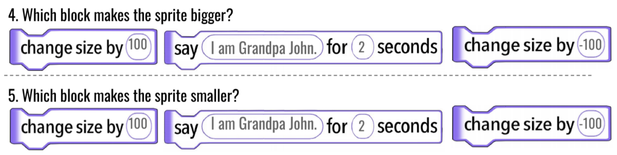

1. Who talks when I click ?

Catrina | Left | Middle | Right

2. Who talks when I press spacebar?

Catrina | Left | Middle | Right

Figure 2: Example "Observe" Questions

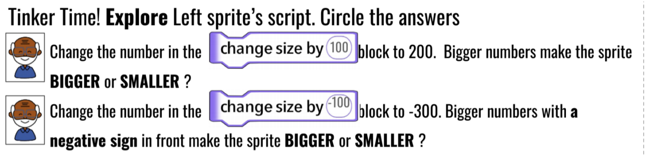

4. Which block makes the sprite bigger?

change size by 100 | say I am Grandpa John. for 2 seconds | change size by -100

5. Which block makes the sprite smaller?

change size by 100 | say I am Grandpa John. for 2 seconds | change size by -100

Figure 3: Example "Predict" Questions


Tinker Time! **Explore** Left sprite's script. Circle the answers

Change the number in the change size by 100 block to 200. Bigger numbers make the sprite **BIGGER** or **SMALLER** ?

Change the number in the change size by -100 block to -300. Bigger numbers with **a negative sign** in front make the sprite **BIGGER** or **SMALLER** ?

Figure 4: Example "Explore" Questions

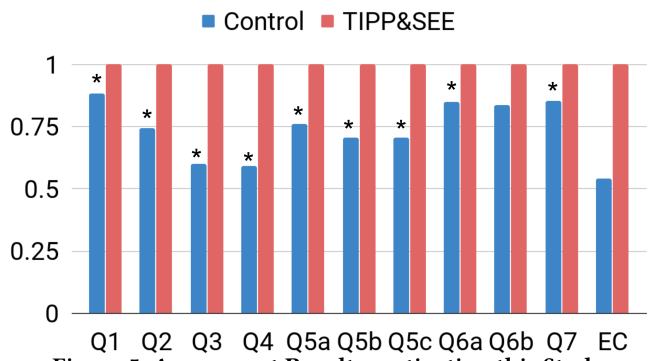


Figure 5: Assessment Results motivating this Study

7. We conclude in section 8 and delineate this study's limitations in section 9.

2 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 [15].

Inspired by previewing strategies that are effective scaffolds in reading comprehension [51], the first half, *TIPP*, guides students in previewing different aspects of a new Scratch project before looking at any code. They first read the Title, Instructions, and the Purpose of the project to get their attention focused on what they will be learning during this activity. The extra P is Play, in which they run the code with very deliberate observations of the events and actions that occur (Figure 2).

The second half, *SEE*, draws from text structure strategies from reading comprehension research [13]. This half involves looking at the code, and begins with students pressing the SEE INSIDE button in the Scratch programming interface. *SEE* provides a roadmap for finding code in the Scratch interface (clicking on the sprite and finding the event). They are asked questions about what they see in the code (Fig. 3), predicting what role the blocks in the code played in the actions they observed in the Play step. The last E is for Explore, in which they are taught how to make changes to the code (e.g. change parameters of blocks) to learn how a particular block works. This proceduralizes the process of learning how to code through deliberate tinkering (Figure 4).

This study is motivated by positive results in previous work [43] showing that students using TIPP&SEE had significantly stronger outcomes than control students. In that curriculum, the goal was for students to form an accurate mental model of a program, as expressed through Scratch. Like Sorva et al. [47], our definition of "understanding" was if students could predict the outcome of a certain script run by the computer or if students could postulate which script produced a certain outcome. Analyzed with the Block model [44], this is a structural understanding of Scratch. In assessments designed for students to demonstrate their understanding, TIPP&SEE students outperformed control students in questions of moderate to advanced difficulty. Figure 5 shows performance on questions related to loops, and student scores are normalized to the treatment group. Asterisks indicate a statistically-significant difference in performance between the two groups. Our goal in this study is to investigate behavioral differences in Use–>Modify–> Create artifacts and TIPP&SEE worksheets that may have led to such dramatic performance differences.

3 THEORETICAL FRAMING

The theoretical framing for this work draws on three sources. The Zone of Proximal Flow grounds our curriculum design, meta-cognitive strategies ground the design of our learning strategy, and Bloom's taxonomy and its variants for computing ground our study design and data analysis.

3.1 Zone of Proximal Flow

Papert [21], in his work on constructionism, posited that individuals learn best when they are constructing an artifact for public consumption, emphasizing self-directed learning. However, such a self-directed approach may not lead to immediate understanding of the concepts underlying their artifacts [2]. A more moderate approach is informed by seeking the Zone of Proximal Flow [1], a combination of Vygotsky's Zone of Proximal Development theory [52] with Csikszentmihalyi's ideas about Flow [9]. The Zone of Proximal Development describes what a student can learn with external support. In contrast, Flow is internally based; a student is in Flow when a task is not so challenging that they are overwhelmed, but not too easy for their skills that they are bored. The Zone of Proximal Flow refers to inquiry-based scaffolding that guides students through the Zone of Proximal Development so that they reach a state of Flow. Zone of Proximal Flow forms the basis of our instructional and curriculum design.

3.2 Meta-cognitive Strategies

TIPP&SEE is a meta-cognitive learning strategy. Meta-cognition involves both self-regulation in learning and motivational aspects of learning. People who are meta-cognitive are reflective and constructive in the learning process, thinking about their own thinking and using this knowledge to guide both thinking and behavior [12]. These expert learners are strategic and purposeful: establishing goals, planning, self-monitoring, self-evaluating, giving self-feedback and correction, and motivating themselves toward the desired end [39]. In short, expert learners are meta-cognitive and strategic about their own learning.

However, strategic learning is an internal monitoring system, and is covert. To a less strategic learner, the "how" of learning is not obvious, and denies access to both process and content. To provide equitable learning opportunities, researchers developed and explored explicit teaching of meta-cognitive strategies, a process for teaching students how to learn within a content area, historically reading [26], writing, [18], math [36], and content such as social studies [10] and science [10]. Most recently, researchers are investigating the use of learning strategies for elementary computer science [43].

Learning strategies prompt and scaffold meta-cognitive thinking. Learning strategies are techniques, principles, or rules that enable a student to learn, solve problems, and to complete tasks independently [11]. The foundational idea of learning strategies is to support all learners in becoming independent by directly teaching them the processes that expert learners use. Meta-cognitive learning strategies make the covert activities of expert learners overt, enabling struggling learners to engage in, practice, and eventually internalize ways to guide their own thinking, motivation, and behaviors to meet learning goals.

Mnemonic devices are one such scaffold. One type of mnemonic uses an acronym to cue memory and coordinate strategic thinking [45]. The mnemonic, TIPP&SEE, cues students to engage purposefully in a series of strategic steps and procedures that are foundational to higher order thinking skills [39], in this case, for computer science learning and problem solving.

3.3 Bloom's Taxonomy for Computing

The original Bloom's taxonomy defined six major cognitive categories: *Knowledge, Comprehension, Application, Analysis, Synthesis, and Evaluation* [3]. These categories were ordered from simple to complex and from concrete to abstract. Further, the original taxonomy represented a strict, cumulative hierarchy. Bloom's taxonomy was later revised to have a second dimension: the knowledge dimension [28]. The knowledge dimension consisted of the following categories: factual, conceptual, procedural, and meta-cognitive. With two dimensions, the revised Bloom's taxonomy was no longer a strict hierarchy, and instead had multiple mastery paths.

With its prominent use in computing, several scholars have proposed modifications to the Bloom's taxonomy to adapt it to have two aspects specific to computing: the ability to develop artifacts being a principal learning objective, and the centrality of studying process and problem solutions [17]. Johnson et al. [24] proposed that Bloom's taxonomy may need a "higher application" level, application informed by a critical approach to the subject .

Fuller et al. expanded upon Johnson's work and proposed the Matrix Taxonomy: a two-dimensional adaptation of Bloom's taxonomy. The two dimensions of the matrix encompass two different competencies: the ability to understand and interpret an existing product, known as the 'Interpreting' dimension, and the ability to design and build a product, known as the 'Producing' dimension. The levels in the 'Interpreting' dimension are *Remember, Understand, Analyze,* and *Evaluate*, while the levels in the 'Producing' dimension are *Apply* and *Create*. In our study, the Use–>Modify and Create tasks enable students to demonstrate their ability to produce code artifacts, with the Use–>Modify task at the *Apply* level and the Create task at the highest 'Producing' level. TIPP&SEE worksheets and end-of-module assessments allow students to hone and demonstrate their interpretation abilities, both with whole projects and individual code snippets respectively. We draw from these three sources of data to evaluate student behavior while using the TIPP&SEE learning strategy.

4 RELATED WORKS

Our study builds upon two flourishing bodies of work: CS education pedagogy and teaching strategies.

4.1 CS Education Pedagogy

As with other subjects, including literacy [6, 49], computer science education researchers disagree on the appropriate level of structure to balance different learning goals. Constructionist curricula are more focused on, and have been shown to be successful at, increasing awareness and engagement, changing perceptions of computing, and building self-efficacy, especially for students from underrepresented communities in computing in informal contexts [25, 38, 34, 37, 50, 14, 5]. This inspired Scratch to create a repository of projects which students can "remix" (copy and modify) and assessments geared towards practices [4] and artifact attributes [16, 19, 40], rather than the conceptual understanding we are seeking.

More structured approaches, on the other hand, aim to develop mental models — an understanding of specific CS concepts and how the code works [47]. While open-ended exploration may lead to the ability to create programs, a worthy goal, there is evidence that its success in teaching mental models is hard to evaluate because conceptual understanding of their own code is not always achieved [2], especially compared to a more direct instruction approach [30]. On the other hand, an overly-structured approach can dissuade some students (especially females) from continuing in programming courses [53].

This paper explores a middle ground, Use−>Modify−>Create [29]. In this approach, students first engage with a concept in a structured project with provided code, and then make changes as they explore how the concept is applied.

4.2 Teaching Strategies

Teaching or learning strategies can be independent from the curriculum. There has been recent work on strategies at the K-12 level. With students ages 15-17, simple debugging strategies (e.g. look at each instruction closely) were tested with limited success [31]. At ages 11-14, an instructional approach called PRIMM was found to be successful. PRIMM involves prediction of the output of example code as the first step of learning a new concept [46]. In addition, the Use−>Modify−>Create (UMC) curricular approach, in which students learn from example code and modify it before creating their own projects, was compared to a less scaffolded curriculum. Students in the UMC treatment found Use−>Modify (UM) tasks easier than control students did, and they felt more ownership over larger projects [33]. Finally, several researchers have provided guidance and research results on using Universal Design for Learning (UDL) instructional techniques in elementary computer science instruction, which posits that learning strategies specifically designed for some students often help many more and provide guidance and different ways to accommodate students with learning differences [20, 23].

Our work focuses on TIPP&SEE, a learning strategy used with students aged 9-10, which led to significant performance advantages [43]. This strategy uses careful observation followed by prediction and exploration to familiarize students with example code.

5 METHODS

5.1 School Context

Fifteen teachers were recruited for this IRB-approved study from a large, urban school district in the United States and underwent the same professional development to teach the Scratch Act 1 to students aged 9-10. A total of 16 classrooms participated in the study, including six of bilingual (English and Spanish) 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 [43], an introductory computational thinking (CT) curriculum modified from the Creative Computing curriculum [8] 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). Each project had a list of requirements for students to complete. 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	Ofrenda	Use−>Modify
	Parallel Path	Use−>Modify
	About Me	Create
Loops	Build a Band	Use−>Modify
	Interactive Story	Create

Table 1: Scratch Act 1 Modules

5.3 Data Analysis

We analyzed three data sources: computational artifacts, TIPP&SEE worksheets, and topical assessments.

5.3.1 Computational Artifacts. Student Scratch projects were statically analyzed to extract the completion of requirements in all the projects. These requirements were listed on their project planning worksheets. Some requirements were designed to help students demonstrate the CT concept, while others were designed to encourage creativity (Table 2). To see if there were any statistically-significant differences between the TIPP&SEE and control students in their requirement completion rates, we used two statistical tests suitable for our large sample size. When comparing the proportion of students who completed a specific requirement, the two-proportion z-test was used; we report z and p values. When comparing countable code attributes, such as script length, the ANOVA F-test was used; we report F, p, and effect size η^2. 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) [7]. 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. $p < .05$ was used for statistical significance.

5.3.2 TIPP&SEE Worksheets. Students worked on TIPP&SEE worksheets prior to starting the UM projects. Questions were divided between the three types of questions: Observe (Figure 2), Predict (Figure 3), and Explore (Figure 4). Answers were transcribed electronically and analyzed for completion and accuracy. Completion rates of each question type varied due to classsroom-level factors, such as instructional time constraints.

Project	Requirements
Name Poem	Modify at least half the sprites Modify backdrop Avg Script Length at least 2
Ladybug Scramble	Ladybug eats at least 1 aphid Use `Eat Aphid` Block Use `Move Steps` Block Use `Turn` Block
5-Block Challenge	Only use the 5 required blocks Add new backdrop Add at least 2 sprites
Ofrenda	Modify Say block for at least 1 sprite Modify at least 1 sprite's costume Add interactivity for at least 1 sprite
Parallel Path	At least 1 sprite has parallel actions on click 2 sprites have actions on "9" key press
About Me	At least 1 sprite At least 1 interactive sprite
Build a Band	Add a script for guitar At least 1 new sprite at least 1 new sprite with a script Cat sprite is animated
Interactive Story	Interactive backdrop At least 1 sprite with a script At least 1 event block At least 1 loop block

Table 2: Scratch Act 1 Project Requirements

5.3.3 Assessments. Students also took two pen-and-paper assessments, one each after Module 2 (events & sequence) and Module 3 (loops). Assessment design was guided by the Evidence-Centered Design Framework [35]. Domain analysis was informed by the CS K-12 Framework and Rich et al's K-8 learning trajectories for elementary computing [41]. 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.

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 reliability of the loops questions (α=.82), suggesting that it was not testing the same concepts as the other questions. Analysis revealed an understanding of the concept of parallelism, instead of loops, was likely more crucial to answering this question correctly.

To see if either project requirement or TIPP&SEE worksheet completion rates were correlated with assessment scores on individual questions, the Spearman's rank correlation coefficient (ρ) was calculated. Spearman's correlation was used as some of our metrics were on an ordinal, not an interval scale. This test also yields a p value for statistical significance.

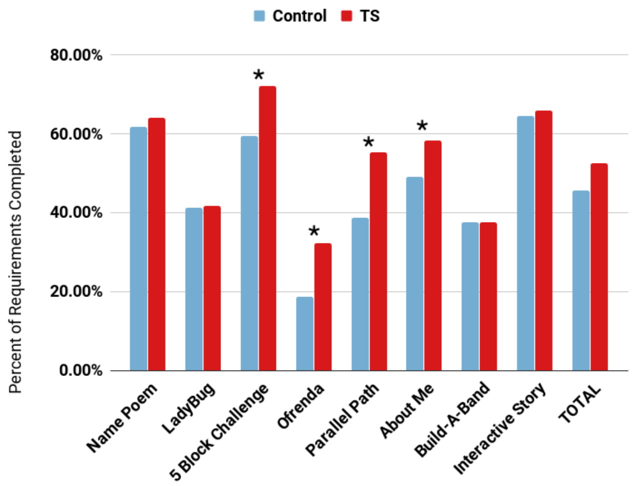


Figure 6: Requirement Completion Rate across Condition

6 RESULTS

We explored student behavior through their artifacts, worksheets, and assessments. Artifact analysis allows us to compare the behaviors of control classrooms to treatment classrooms. Their artifacts were examined for attributes that indicated the fulfilment of the project requirements. Treatment classrooms' TIPP&SEE worksheets were inspected for the completion of the Observe, Predict, and Explore phases. Finally, artifact attributes, worksheet correctness, and worksheet completion rates were analyzed for any correlations with assessment scores.

We present three sets of results: artifact attributes, TIPP&SEE worksheets, and correlations between data sources. Within each section, we highlight important individual findings alongside presentation of the evidence for those findings. We then discuss the findings with respect to each other and the original research questions in Section 7.

6.1 Artifact Attributes

We compared the attributes of student projects in two ways: (1) across condition (control vs TIPP&SEE) and (2) across individual classrooms. Each student project was created in the context of either a Use−>Modify or Create activity in the curriculum. Control and treatment students had different worksheets for Use−>Modify activities, but identical materials for Create activities. We first present the overall results, then we present select detailed results.

6.1.1 Overall Results. Figure 6 depicts overall requirement completion rate across the entire curriculum. For each project, the left (blue) bar shows control, and the right (red) bar shows the treatment results.

Finding 1: TIPP&SEE students satisfied either the same or higher percentage of requirements than the control students.

TIPP&SEE students were more likely to complete *all* the project requirements for *5-Block Challenge* ($z = 10.25, p < .01$), *Ofrenda* ($z = 9.34, p < .01$), *Parallel Path* ($z = 9.34, p < .01$), and *About*

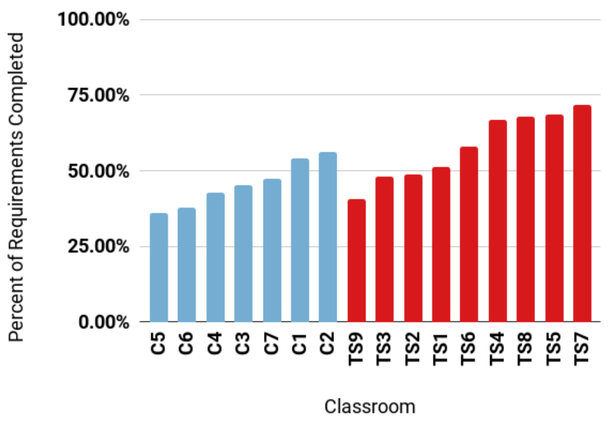


Figure 7: Per-Classroom Overall Requirement Completion

Me ($z = 6.12, p < .01$). Table 3 shows the individual requirements for each project where TIPP&SEE students had a statistically-significantly higher completion rate.

Project	Attribute	z
Ladybug Scramble	Ladybug eats 1 aphid	2.47*
5 Block Challenge	Used 2 specified blocks	5.63**
	Used 4 specified blocks	4.81**
	Used all 5 specified blocks	3.46**
Ofrenda	1 interactive sprite	2.47*
	More than 1 interactive sprite	2.12*
	All sprites with a different costume	3.42**
	All sprites have a different Say block	3.63**
Parallel Path	1 sprite with parallel actions on click	7.46**
	2 or more sprites with parallel actions on click	6.57**
	1 sprite acts on "9" key press	6.77**
	2 or more sprites act on "9" key press	6.06**
About Me	Has a Say block	2.38*
	Has an interactive sprite	3.51**
Build a Band	Modified scripts for at least 1 sprite	3.24**
Interactive Story	Interactive Backdrop	2.23*
	Using at least 10 blocks	2.19*
* $p < .05$	** $p < .01$	

Table 3: Attributes with Significant TIPP&SEE Outperformance

Finding 2: Completion rates varied for different classrooms, and with substantial overlap between treatment and control classrooms.

Figure 7 breaks down the overall results by classroom, ordered by percentage of requirements completed. We can see that not all treatment classrooms complete more requirements than all control classrooms. However, it is clear that treatment classrooms did better in general.

Finding 3: Individual requirements with higher completion rates by control students do not utilize as much coding.

While TIPP&SEE students fulfilled more requirements than control students overall, there were some requirements that more control students completed. These exceptions were mostly in superficial, not programming-related, requirements. For *Name Poem*, a greater proportion of students in the control condition changed the backdrop. In *Build a Band*, a larger percentage of control students added a new sprite and added new blocks that would animate the Cat sprite that was already in the example project, with the difference in adding a new sprite being statistically significant ($z = 2.64; p < .01$). TIPP&SEE students outperformed control students in requirements that were programming-related, as showcased in the *5-Block Challenge*, *Ofrenda*, *Parallel Path*, and *Interactive Story* projects.

Discussion. Overall, we see that students in treatment classrooms not only completed more elements (Finding 1) but focused more on programming elements (Finding 3). This shows that one purpose of the learning strategy and worksheets - to help students complete the projects - was successful. Our initial hypothesis was that, overall, completing more of the projects will lead to better performance on written assessments, a correlation we explore in Section 6.3.

6.1.2 *Per-Project Results.*

We now examine more closely a subset of the projects. *Ofrenda* was chosen because it represents typical behavior for a project on which students generally did well on the written assessment for both control and treatment groups. *5-Block Challenge* and *Parallel Path* were chosen because they were redesigned based on analysis of the previous year's student work and written assessments. Finally, Interactive story was chosen because it is the culminating project for the curriculum. *Ofrenda* and *Parallel Path* are Use−>Modify projects, whereas *5-Block Challenge* and *Interactive Story* are Create projects. We want to find out whether students completed these projects differently according to their group.

Ofrenda. Inspired by the Mexican holiday Día de los Muertos (Day of the Dead), the *Ofrenda* Use−>Modify project presented students with three ancestors as sprites. Students were then prompted to modify the project by adding their family members as sprites and making them interactive. There are three requirements in the Modify task, two requiring coding (Interaction and Speaking), and one involving changing the sprites' costumes.

In order to illustrate student behavior, we distinguish between fulfilling the requirement on a single sprite (practicing it once) and on all of the sprites (practicing it and completing the assigned task). Figure 8 depicts the results, with the top (red) portion of the bar showing the percentage of students who completed the task for a single sprite, and the bottom (blue) portion of the bar showing the percentage who completed the task for all sprites.

Finding 4: More treatment students completed requirements for at least one sprite as well as for all sprites.

The total height of the bars are higher for treatment students, indicating more completion of any sprite, as well as the bottom (blue) portion of the bar, indicating all sprites. This implies that students both demonstrate some understanding and are potentially more thorough in their work when following the TIPP&SEE strategy and worksheets. A statistical analysis revealed that TIPP&SEE students outperformed control students in making both one ($z = 2.47, p <$

.05) and all sprites ($z = 2.12, p < .05$) interactive, changing all sprites costumes ($z = 3.42, p < .01$), and making all sprites speak ($z = 3.63, p < .01$). This indicates that TIPP&SEE students better demonstrated their ability to *apply* their knowledge of Events.

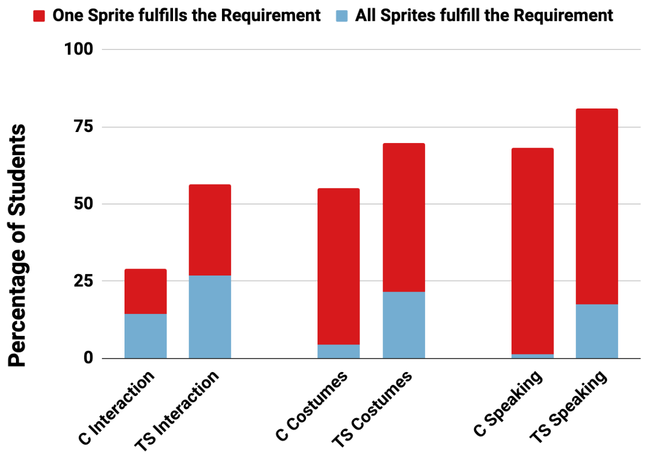


Figure 8: Ofrenda Completion Rate across Conditions

5-Block Challenge. The *5-Block Challenge* Create project, with identical materials for both groups, prompts students to create a project using only 5 blocks: `When this Sprite Clicked`, `When Green Flag Clicked`, `Wait`, `Say`, and `Glide`. The goal is twofold: encourage students to build with blocks they haven't been explicitly taught, and encourage students to create scripts with multiple action blocks rather than lots of scripts with a single action block. This project was modified from the 10-Block Challenge because analysis of previous years' student artifacts (including final projects) revealed few scripts contained more than a single action block.

Finding 5: Students in treatment classes created longer scripts, on average, than control classrooms during the 5-Block Challenge.

We analyzed two major statistics, shown in Figure 9. First, we calculated the average script length in each student's artifact. Second, we calculated the length of the longest script in each student's artifact. We can see that treatment classrooms, on average, create longer scripts, with treatment students creating scripts with 5.22 blocks and control students with 3.97 blocks ($F(1, 196) = 9.01, p < .01, \eta^2 = .044$). This means that treatment students, in general, went beyond the minimal two blocks per script to create sequential scripts. The mean maximum was slightly higher in the control groups, at 7.45 vs. 6.27, due to students with incredibly long scripts (over 45 blocks) in one classroom, whereas the median maximum script length was higher in the treatment groups. However, an analysis of variance of the maximum script lengths between the two conditions showed that these differences were not statistically significant ($F(1, 196) = 1.37, p = .24$). Therefore, in general, students in the treatment group created at least one script that was of non-trivial length, showing more practice at creating sequential scripts.

Finding 6: Students in treatment classes used more required blocks, on average, during the 5-Block Challenge.

The left bars on Figure 10 shows the percentage of students who used different numbers of the required blocks. The best case would be entirely blue bars, in which all students utilized all 5 blocks. A majority of treatment students used 4-5 blocks, whereas a majority of control students used 2-3 blocks. While students in the control condition were more likely to use only two specified blocks ($z = 5.63, p < .01$), students using TIPP&SEE used more of the specified blocks. A statistically-significantly greater proportion of treatment students used four ($z = 4.81, p < .01$) and five ($z = 3.46, p < .01$) blocks. Along the Matrix Taxonomy, higher block usage by the TIPP&SEE students suggests that they were better able to *create* artifacts in the context of a cumulative project on Sequence.

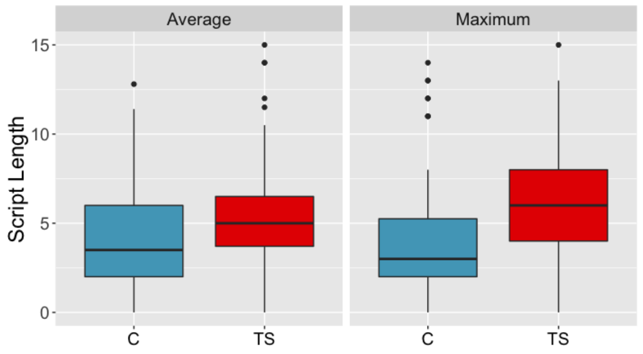


Figure 9: Block Lengths across Condition

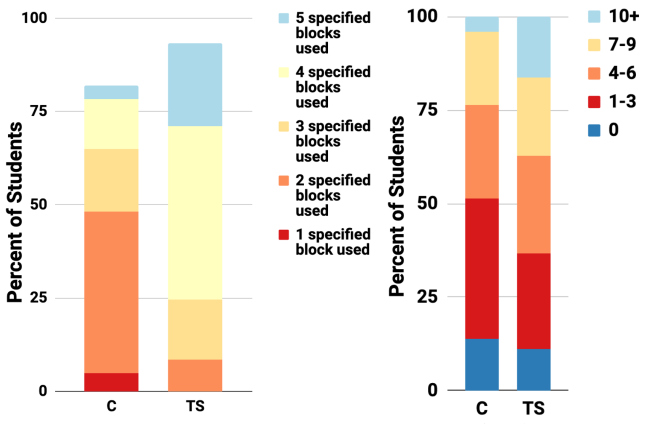


Figure 10: Block Use for 5-Block Challenge (left) & Interactive Story (right)

Parallel Path. The *Parallel Path* Use–>Modify project was created in response to poor student performance on written assessment questions involving parallel versus sequential code. The project presents students with two sprites that had actions either in sequence or in parallel, depending on which number they pressed. The TIPP&SEE worksheet had students identify what actions were sequential vs parallel and then inspect the corresponding code. Students were then asked to modify the project such that upon a mouse click, each sprite would do two actions in parallel, and when the number '9' key was pressed, both sprites would do an action in parallel. These results are shown because they represent the most staggering difference in behavior between the two groups.

Finding 7: A significantly larger percentage of TIPP&SEE students satisfied the requirements of Parallel Path than the control group.

Figure 11 depicts the percentage of students who completed each requirement. It shows that there was no single requirement that 20% of control students completed, yet for all requirements, at least 45% of TIPP&SEE students completed them. In fact, less than 25% of students in the control group completed any single requirement. In contrast, almost 75% of TIPP&SEE students completed a single requirement, and over 50% of these students completed the entire project. TIPP&SEE students significantly outperformed the control students in every requirement: programming 1 sprite ($z = 7.46, p < .01$) and at least 2 sprites ($z = 5.67, p < .01$) to do two parallel actions on click, programming 1 sprite ($z = 6.77, p < .01$) and at least 2 sprites ($z = 6.06, p < .01$) to act when the '9' key is pressed. TIPP&SEE students were also more likely to fulfil all requirements ($z = 6.67, p < .01$). The TIPP&SEE students' better performance is especially noteworthy because parallelism is a concept with which students commonly struggle [27, 32].

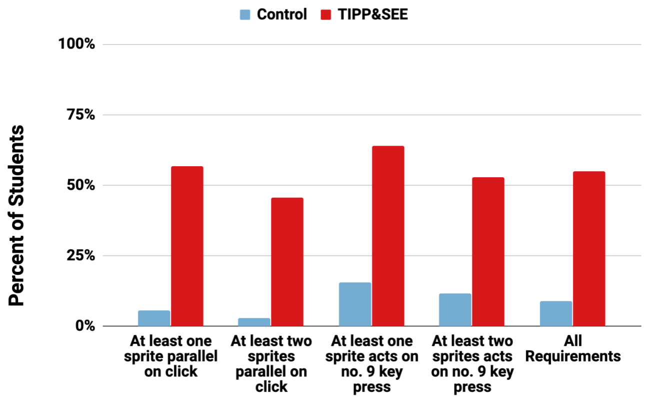


Figure 11: Parallel Path Completion Rate across Conditions

Interactive Story. *Interactive Story* is the culminating Create project in this curriculum, designed to encourage students to demonstrate their knowledge of the three CT concepts covered: events, sequence, and loops.

The right two bars of Figure 10 illustrate the number of unique blocks that students utilize in their final projects. A greater percentage of the control group used fewer distinct block types, while the TIPP&SEE group used more distinct block types. Most notably, TIPP&SEE student projects were more likely to have at least 10 unique blocks relative to the control student projects ($z = 2.19, p < .05$). Further, TIPP&SEE students outperformed the control students in using either the `switch backdrop` or `when backdrop changes` block to make their backdrop interactive ($z = 2.23, p < .05$).

6.2 TIPP&SEE Worksheets

We now turn our attention to analysis of the TIPP&SEE worksheets. These worksheets were completed only by students in the treatment group; the worksheets for control students presented the project and had the modify tasks listed, but they did not have a set of questions for students to answer.

All figures in this section break up student responses into four categories: Correct, Incorrect, Blank, and No Sheet. The distinction between Blank and No Sheet is that a Blank answer was collected but was not answered by the student, whereas No Sheet indicates that we are missing the entire worksheet for that student.

We begin by exploring student behavior on different types of TIPP&SEE questions. There are three categories of questions we analyzed. The Observe questions are first, asking students to record their observations from running the provided project. All worksheets have Observe questions. The other two question categories are only on a subset of worksheets. Predict questions ask students to look at the code and predict what blocks caused which actions they observed. Explore questions have two parts. First, make a change to the code and run it, such as changing the number in the `wait` block. Next, record what happened in response, such as whether the sprite moved faster or slower. There are other question categories, but these are the three we analyze.

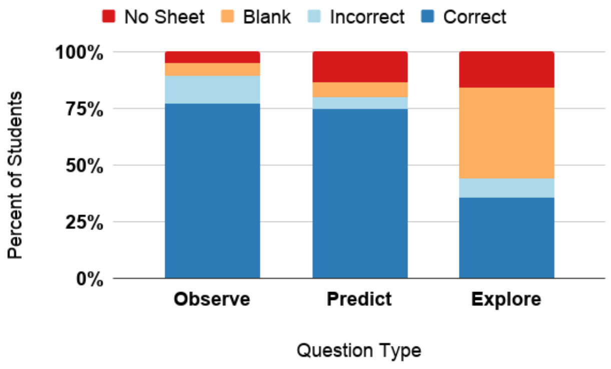


Figure 12: TIPP&SEE Worksheet Responses across Question Types

Finding 8: A majority of students completed and correctly answered Observe and Predict questions, while Explore questions were largely left blank.

Figure 12 shows the percentage of students that completed and correctly answered questions across all TIPP&SEE worksheets, sorted by the type of question. It shows that, overall, there were few incorrect answers. However, a majority of students did not record answers to Explore questions.

It is unclear if the reason for skipping Explore questions was because students did not follow the Explore prompt or because they did not record their observations. There are several reasons, however, that students could have skipped them. First, because explore questions were only included in a few projects, following and recording explore prompts may not have become a routine. On a related note, students may have needed more scaffolding with this type of questions, requiring the teacher to model and practice them. In addition, making code changes is a more difficult task than merely answering a question about what one observes or is thinking, so this may have been cognitively difficult for some students.

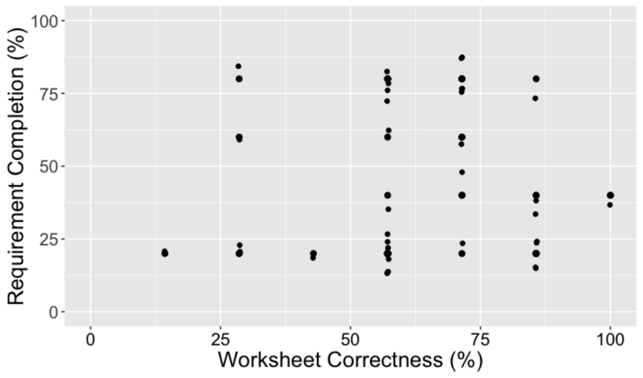


Figure 13: Ladybug Scramble Worksheet Correctness vs Project Completion

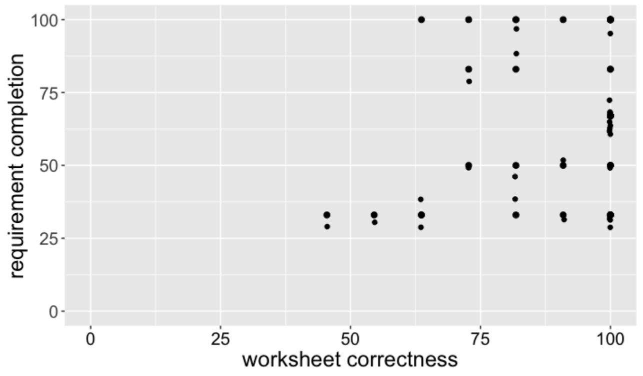


Figure 14: Parallel Path Worksheet Correctness vs Project Completion

6.3 Correlations between Projects, Worksheets, and Assessments

Having analyzed assessments [43], project completion, and worksheets independently, we now investigate relationships between them.

6.3.1 Worksheets vs Projects. We begin by analyzing correctness and completeness of TIPP&SEE worksheets compared with requirement completion on the projects. Only Use−>Modify activities have worksheets. Our question is, does higher completion or correctness of worksheet questions correlate with higher requirement completion on modify tasks?

Finding 9: There was very little correlation between TIPP&SEE worksheets and project completion.

The only UM project with any correlation between worksheet correctness and project completion was *Ofrenda* ($\rho = .33, p < .05$). For the rest of the projects, the distribution of these metrics per student fell into two broad categories. In the first category, worksheet correctness and requirement completion rates were scattered all over the place, such as *Name Poem* and *Ladybug Scramble* (Figure 13). In the second category, these metrics were concentrated in the right half of the plot (i.e. at least 50% worksheet correctness), but do not follow any pattern beyond that, such as *Parallel Path* (Figure 14) and *Build a Band.*

6.3.2 Worksheets and Projects vs Assessments. We now consider any correlations between worksheet completion, worksheet correctness, project completion, compared with written assessments.

Finding 10: There was very little correlation between project attributes, worksheets, and assessments.

The correlations that were statistically significant were relatively weak and came from the *Name Poem*, *Ladybug Scramble*, and *Ofrenda* projects.

In *Name Poem* , the first Sequence Use−>Modify Project, there was a weak correlation between requirement completion and scores on a question from the Events and Sequence assessment ($\rho = .33, p < .05$). This question showed two sprites with `Say` bubbles on the Scratch stage, triggered by clicking the green flag. Students were then asked to identify the script that belonged to one of the sprites.

In *Ladybug Scramble* , the second Sequence Use−>Modify Project, there were weak correlations between TIPP&SEE worksheet *completion* and a two-part question on Events and Sequence (a: $\rho = .33, p < .05$, b: $\rho = .34, p < .05$). This question presented students with a script; the first part asked students to identify the event that triggered the script and the second part asked students to describe the sequence of blocks in the script. There was also a weak correlation between the second part of this question and the overall project completion rate ($\rho = .32, p < .05$). We also found weak correlations between worksheet *correctness* and two questions: one asked students to identify the last `Say` block in a script ($\rho = .40, p < .01$) and another asked them to identify the scripts triggered by the `Green Flag`($\rho = .34, p < .05$).

7 DISCUSSION

We now revisit our overarching research questions and relate our individual findings to these questions.

To what degree do students follow the TIPP&SEE protocol, and how accurately can they answer the posed questions?

Students follow the Observe and Modify closely and, for the most part, answer the questions accurately. A majority of students, however, do not complete the Explore questions when they are present. We cannot tell if students were truly disengaged from the Explore questions or if students were exploring implicitly, which can occur with metacognitive strategies. Therefore, further research should address either improving the participation for Explore questions or determining that they are not useful for student learning.

How does using the TIPP&SEE learning strategy affect student behavior during the completion of Use–>Modify and Create tasks?

There was a significant difference in behavior, overall, between control and treatment students. We had findings on a variety of measures, including project requirement completion, length of scripts, and required block usage. The results from *Parallel Path* are particularly staggering, with treatment students completing requirements at about 8-10 times the rate of control students. Students in the treatment group stayed on task much better than control students, even on Create projects in which the materials were identical. This finding suggests that treatment students were more capable of applying their new knowledge, the first ‘production’ step in the Matrix Taxonomy, and that they benefited from the Zone of Proximal Flow encouraged by the curriculum design. In addition, when looking at the *5-Block Challenge* and *Interactive Story* results, TIPP&SEE students were better able to *create* more complex projects, the highest ‘production’ level in the Matrix Taxonomy.

This leads to an interesting question - is TIPP&SEE a learning strategy or an instructional strategy? We have strong evidence that it leads to positive outcomes from an instructional perspective. That is, when students follow this sequence of actions, mediated by a worksheet, it leads to positive outcomes. However, whether the students internalize this into a sequence they can complete without the worksheet, which would make TIPP&SEE a *meta-cognitive* learning strategy, is a question this study does not address. The cognitive aspects are harder to measure, and therefore harder to evaluate.

Are there any statistical correlations between behavior on the TIPP&SEE worksheets or project attributes and written assessment performance?

We find very few statistical correlations between any of the behavioral measures: individual requirement completed, percentage of requirements completed, worksheet questions completed, and individual written assessment question performance.

The lack of correlations between project attributes and assessments is not entirely surprising. On the Matrix Taxonomy, project attributes reflect the ’Producing’ dimension, while assessments reflect the ’Interpreting’ dimension; it is possible for both dimensions to develop independently [17]. Further, Brennan et al. [4] have shown that students frequently use code that they do not fully understand. Another prior study also revealed that student artifacts can have false positives, where students use code that they do not understand, and false negatives, where students understand a concept but do not use related code constructs [42]. Students may have run out of time to include these code constructs or simply did not see the need for those constructs in their projects.

In contrast, the fact that the worksheet behaviors (both completeness and correctness) were hardly correlated with the assessments was more unexpected, as both reflect the same ’Interpreting’ dimension of the Matrix Taxonomy. Previous studies have found relationships between formative activities or assignments and learning in Scratch [19, 48]. These activities and assignments varied widely in structure. Even within our curriculum, the TIPP&SEE worksheets differed in structure as well. The influence of TIPP&SEE worksheet design on learning merits further exploration.

8 CONCLUSIONS

This study sheds light on the *behaviors* of students using TIPP&SEE. There was a significant difference in behavior, overall, between control and treatment students. We had findings on a variety of measures, including project requirement completion, length of scripts, and required block usage. Students in the treatment group are better able to stay on task, either because their learning has been scaffolded well or because TIPP&SEE provides structure for their work. However, not all of the aspects of the TIPP&SEE strategy were successful. In particular, a subset of Use–>Modify projects had explore questions to scaffold learning by making changes in code. However, a majority of students did not answer those questions. Finally, there were little to no correlations between individual student acts, such as performance on worksheets, project requirements, and written assessments. In the end, worksheets and project work are imperfect measurements of what knowledge already exists or is being created within a student’s mind.

9 LIMITATIONS

While this study had a fairly large sample size, we only analyzed submitted work - worksheets computational artifacts, and assessments. However, data beyond that, such as time-on-task, teacher interviews and classroom observations, were not rigorously analyzed for this study. Further, although we balanced the language of instruction and teacher experience in each condition, our teachers came from different schools and may differ slightly in the way they teach the curriculum. Therefore, while we can observe similarities or differences, the reasons for these differences might be further illuminated through additional data analysis.

In addition, learning strategies are meant to teach cognitive processes that are invisible to the observer. We cannot tell whether something like the Explore prompts were useful because they made students think in a certain way (despite them not recording results). We also cannot tell if the prompts were not helpful because either students would already have thought that way without the prompt or students ignored them.

10 ACKNOWLEDGEMENTS

We would like to thank the teachers and students who participated in this study. This material is based upon work supported by the National Science Foundation under Grant Nos. 1660871, 1760055, and DGE-1746045.

REFERENCES

[1] 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.
[2] John B Biggs and Kevin F Collis. *Evaluating the quality of learning: The SOLO taxonomy (Structure of the Observed Learning Outcome)*. Academic Press, 2014.
[3] Benjamin S Bloom et al. “Taxonomy of educational objectives. Vol. 1: Cognitive domain”. In: *New York: McKay* (1956), pp. 20–24.
[4] Karen Brennan and Mitchel Resnick. “New frameworks for studying and assessing the development of computational thinking”. In: *Proceedings of the 2012 annual meeting of the American Educational Research Association, Vancouver, Canada*. Vol. 1. 2012, p. 25.
[5] Quinn Burke and Yasmin Kafai. “The Writers’ Workshop for Youth Programmers Digital Storytelling with Scratch in Middle School Classrooms”. In: *ACM Proceedings of the Annual ACM Symposium on Computer Science Education* (Feb. 2012), pp. 433–438.
[6] Susan Chambers Cantrell. “Effective teaching and literacy learning: A look inside primary classrooms”. In: *The Reading Teacher* 52.4 (1998), pp. 370–378.
[7] Jacob Cohen. *Statistical power analysis for the behavioural sciences.* 1988.
[8] Creative Computing. *An introductory computing curriculum using Scratch.*
[9] Mihaly Csikszentmihalyi, Sami Abuhamdeh, and Jeanne Nakamura. “Flow”. In: *Flow and the foundations of positive psychology*. Springer, 2014, pp. 227–238.
[10] Susan De La Paz. “Effects of historical reasoning instruction and writing strategy mastery in culturally and academically diverse middle school classrooms.” In: *Journal of educational psychology* 97.2 (2005), p. 139.
[11] DD Deshler and JB Schumaker. “Strategies instruction: A new way to teach”. In: *Salt Lake City: Worldwide Media* (1984).
[12] Anouk S Donker et al. “Effectiveness of learning strategy instruction on academic performance: A meta-analysis”. In: *Educational Research Review* 11 (2014), pp. 1–26.
[13] S Dymock. “Teaching Expository Text Structure Awareness”. In: *The Reading Teacher* 59.2 (2005), pp. 177–181.
[14] Barbara Ericson and Tom McKlin. “Effective and Sustainable Computing Summer Camps”. In: 2012.

[15] 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.

[16] Diana Franklin et al. "Assessment of computer science learning in a scratch-based outreach program". In: *Proceeding of the 44th ACM technical symposium on Computer science education*. ACM. 2013, pp. 371–376.

[17] Ursula Fuller et al. "Developing a computer science-specific learning taxonomy". In: *ACM SIGCSE Bulletin* 39.4 (2007), pp. 152–170.

[18] Steve Graham et al. "A meta-analysis of writing instruction for students in the elementary grades." In: *Journal of educational psychology* 104.4 (2012), p. 879.

[19] Shuchi Grover, Roy Pea, and Stephen Cooper. "Designing for deeper learning in a blended computer science course for middle school students". In: *Computer Science Education* 25.2 (2015), pp. 199–237.

[20] Alexandria K Hansen et al. "Differentiating for diversity: Using universal design for learning in elementary computer science education". In: *Proceedings of the 47th ACM Technical Symposium on Computing Science Education*. 2016, pp. 376–381.

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

[22] Peter Hubwieser et al. "A global snapshot of computer science education in K-12 schools". In: *Proceedings of the 2015 ITiCSE on working group reports*. ACM. 2015, pp. 65–83.

[23] Maya Israel, Cecelia Ribuffo, and Sean Smith. "Universal Design for Learning innovation configuration: Recommendations for teacher preparation and professional development (Document No. IC-7)". In: *Retrieved from University of Florida, Collaboration for Effective Educator, Development, Accountability, and Reform Center website: http://ceedar. education. ufl. edu/tools/innovation-configurations* (2014).

[24] Colin G Johnson and Ursula Fuller. "Is Bloom's taxonomy appropriate for computer science?" In: *Proceedings of the 6th Baltic Sea conference on Computing education research: Koli Calling 2006*. 2006, pp. 120–123.

[25] 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.

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

[27] Yifat Ben-David Kolikant. "Gardeners and cinema tickets: High school students' preconceptions of concurrency". In: *Computer Science Education* 11.3 (2001), pp. 221–245.

[28] David R Krathwohl. "A revision of Bloom's taxonomy: An overview". In: *Theory into practice* 41.4 (2002), pp. 212–218.

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

[30] Michael J Lee and Amy 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.

[31] Michael J Lee et al. "Principles of a debugging-first puzzle game for computing education". In: *2014 IEEE symposium on visual languages and human-centric computing (VL/HCC)*. IEEE. 2014, pp. 57–64.

[32] Gary Lewandowski et al. "Commonsense computing (episode 3): concurrency and concert tickets". In: *Proceedings of the third international workshop on Computing education research*. ACM. 2007, pp. 133–144.

[33] Nicholas Lytle et al. "Use, Modify, Create: Comparing Computational Thinking Lesson Progressions for STEM Classes". In: *Proceedings of the 2019 ACM Conference on Innovation and Technology in Computer Science Education*. 2019, pp. 395–401.

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

[35] 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.

[36] Marjorie Montague. "The effects of cognitive and metacognitive strategy instruction on the mathematical problem solving of middle school students with learning disabilities". In: *Journal of learning disabilities* 25.4 (1992), pp. 230–248.

[37] 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.

[38] 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.

[39] Michael Pressley, John G Borkowski, and Wolfgang Schneider. "Cognitive strategies: Good strategy users coordinate metacognition and knowledge". In: (2010).

[40] Alexander Repenning and Andri Ioannidou. "Broadening participation through scalable game design". In: *ACM SIGCSE Bulletin*. Vol. 40. 1. ACM. 2008, pp. 305–309.

[41] 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.

[42] Jean Salac and Diana Franklin. "If They Build It, Will They Understand It? Exploring the Relationship between Student Code and Performance". In: 2020.

[43] Jean Salac et al. "TIPPSEE: A Learning Strategy to Guide Students through Use - Modify Scratch Activities". In: 2020.

[44] Carsten Schulte. "Block Model: an educational model of program comprehension as a tool for a scholarly approach to teaching". In: *Proceedings of the Fourth international Workshop on Computing Education Research*. 2008, pp. 149–160.

[45] Thomas E Scruggs et al. "Mnemonic strategies: Evidence-based practice and practice-based evidence". In: *Intervention in School and Clinic* 46.2 (2010), pp. 79–86.

[46] 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.

[47] Juha Sorva. "Notional Machines and Introductory Programming Education". In: *ACM Transactions on Computing Education* 13 (June 2013), 8:1–8:31. DOI: 10.1145/2483710.2483713.

[48] Addison YS Su et al. "Effects of annotations and homework on learning achievement: An empirical study of Scratch programming pedagogy". In: *Journal of Educational Technology & Society* 18.4 (2015), pp. 331–343.

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

[50] 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.

[51] Sharon Vaughn et al. "Efficacy of Collaborative Strategic Reading With Middle School Students". In: *American Education Research Journal* 48.4 (2011), pp. 938–964.

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

[53] 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.