# Learning Computational Thinking Efficiently

## How Parsons Programming Puzzles within Scratch Might Help

Jeff Bender
Computer Science, Columbia University, New York City NY USA
jeffrey.bender@columbia.edu

Bingpu Zhao
Computer Science, Columbia University, New York City NY USA
bz2328@columbia.edu@columbia.edu

Alex Dziena
Computer Science, Columbia University, New York City NY USA
ad333@columbia.edu

Gail Kaiser
Computer Science, Columbia University, New York City NY USA
kaiser@cs.columbia.edu

## ABSTRACT

Using a design thinking approach, we surveyed and interviewed grade 6-9 teachers on their experience with Scratch and Parsons Programming Puzzles (PPP). The results lead us to extend Scratch with gameful PPP functionality focused on individual computational thinking (CT) concepts. In this paper, we vary elements of PPPs presented to 624 adult learners to identify those yielding manageable cognitive load (CL), and maximum CT motivation and learning efficiency, for a general populace. Findings indicate PPPs with feedback and without distractors limit CL, those with feedback produce highest CT motivation, and those with an isolated block palette and without distractors produce highest CT learning efficiency. We analyze study data across nine conditions to offer insight to those developing PPP systems with the aim to advance equitable CT education for all.

## CCS CONCEPTS

• **Social and professional topics** → Professional topics; Computing education; Computational thinking.

## KEYWORDS

Computational Thinking, Parsons Programming Puzzles, Scratch, Motivation, Cognitive Load, Learning Efficiency

**ACM Reference Format:**
Jeff Bender, Bingpu Zhao, Alex Dziena, and Gail Kaiser. 2022. Learning Computational Thinking Efficiently: How Parsons Programming Puzzles within Scratch Might Help. In *Australasian Computing Education Conference (ACE '22), February 14–18, 2022, Virtual Event, Australia.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3511861.3511869

## 1 INTRODUCTION

Computational thinking (CT) is becoming a necessary literacy alongside reading, writing, and arithmetic [1]. However, negative perceptions and poor understanding of computing remains prevalent among the general population across ages [2, 3]. The dependency on CT competencies has emerged in an era of standardized testing that limits curricula flexibility in formal education, leading administrators toward zero-sum requirements choices [4]. When schools do integrate CT, they often operate with teachers with limited subject or pedagogical content knowledge who are unlikely to have learned CT in K-12 [5, 6].

The CSforAll initiative in the U.S. used a commitment-making model to engage the community and distribute effort to local leaders in both formal and informal education [7]. The scale of the needed transformation, though, introduces risk to lasting equitable CT uptake for all. Since different demographic groups' communal values vary, and computing is not perceived to fulfill those values equally across groups, the opportunity to learn CT and develop a sense of belonging in computing is important to afford both to kids and the adults who shape those values to reduce inequity [8]. A CS career orientation for female students is strongly correlated with self-efficacy in computing [9], and computing exposure and encouragement most influence women to study the field [10], yet only 33% of the 2018 enrolments in the Australian National Computer Science School challenge were female [11], and 20% of 2018 Information and Communications Technologies OECD tertiary education students were women [12].

Since motivation and previous programming experience can be highly correlated with self-efficacy and CS career orientation across genders [9], and computing experiences impact computing self-image and habits [2], increasing the general populace's computing motivation and self-perception of CT skill are key objectives. For middle school students, previous experience with block-based programming has correlated with computer use and confidence, as well as interest in future CS courses [13]. Like early efforts to use block programming to introduce constructs and logic prior to syntax for adult university students [14], our study explores the motivation changes and CT learning efficiency associated with introducing to an adult population modified versions of a popular block-based environment, Scratch [15].

Previous work integrated with Scratch Parsons Programming Puzzles (PPPs) [16], which are program completion tasks that enable learners to practice CT by assembling into correct order sets of mixed-up blocks that comprise samples of well-written code, often focused on a single concept [17]. The findings aligned with others


*ACE '22, February 14–18, 2022, Virtual Event, Australia*

ACM ISBN 978-1-4503-9643-1/22/02…$15.00
https://doi.org/10.1145/3511861.3511869

indicating this structured approach can lead to more efficient CT learning than alternatives such as via tutorial, or writing/fixing code [18–20]. In this study, we seek further evidence of PPP learning efficiency and impact on CT motivation.

We first further augment Scratch, which is used by K-8 teachers more than any other coding language internationally and is the most studied in CS literature for K-12 [21–23]. Despite its success, Scratchers do not consistently demonstrate skill increases over time [24], often misconceive CT concepts, and can adopt programming habits misaligned with CS norms, such as bricolage [25], which can involve bottom-up tinkering without meaningful learning [26]. To address these weaknesses, researchers have designed external curricula [27, 28], advocated for the Use->Modify->Create pedagogy to scaffold instruction [29], and created introductory Scratch Microworlds with fewer features [30]. Our integration of PPPs within Scratch internalizes this trend by challenging learners with explicit goals, gameful scoring targets, and per-block feedback that discourages trial-and-error behavior and focuses CT concept learning. We reason that if learners can master CT concepts efficiently via PPPs in Scratch, they can more effectively deepen understanding in less restrictive open-ended projects that embrace Scratch's roots in learning via the creation of personally meaningful artifacts [31, 32].

To test this reasoning, we ran an online international study targeting adults in which each participant was randomly selected into 1 of 9 conditions to learn the CT concept *conditionals* via 4 puzzles each limited to 8 minutes. Between conditions, we varied the presentation of programming constructs, the inclusion of distractors, feedback activation, and for the control condition, the CT concept. Here, we investigate 3 research questions: after at most 32 minutes (m) of puzzle solving, what are the effects of PPP variation on adult learner **R1**) cognitive load (CL)?; **R2**) CT motivation?; **R3**) learning efficiency? Findings from 624 participants indicate: **F1**) PPPs with feedback and without distractors produce the lowest CL; **F2**) PPPs with feedback produce highest CT motivation; **F3**) PPPs with an isolated block palette and without distractors produce highest CT learning efficiency. We first review related work and the software developed, then the study purpose, formative and summative evaluations, and results, before previewing future work. We detail measurements for efficiency in section 2.3 and motivation in 6.1.

## 2 RELATED WORK

PPPs first emerged as a new form of program completion problem in 2006. [16] documents their strengths and weaknesses, [33] consolidates results from empirical studies, and [34] reviews the many variants in the research literature. Summarily, the scaffolded support of syntax and semantics learning can lead to shorter training time compared to code writing without reducing performance on transfer tasks, resulting in meaningful learning efficiency gains. Here we review work related to three axes of variation introduced in our study to identify optimal conditions for efficient CT learning: 1) presentation; 2) distractors; 3) feedback.

### 2.1 Presentation

The form of content presentation can significantly influence the learning experience [35]. Block-based languages offer visual, natural language, browsability, drag-and-drop composition, and dynamic rendering affordances beyond those typically available in text-based languages [36]. Open-ended environments like Scratch, however, traditionally offer no clear objectives nor direct instruction. Most PPPs provide learners with a precise challenge to solve via a prompt and focus the learner on arranging a small set of selected programming constructs. In this study, we replace short prompts with step-by-step instructions like those used in tutorials and vary the presentation of constructs across conditions.

[37] studied PPPs in comparison to tutorials by offering students choice in modality for each task and found that learners sought a challenge for 57% of PPPs compared to 32% when selecting tutorials, while aiming to avoid challenge in 39% of the tutorials compared to 12% of PPPs. PPP solvers also completed tasks in 23% less time and performed 26% better on transfer tasks than tutorial followers while reporting higher mental effort. The tutorial tasks, however, attempted to teach both the interface mechanics and the programming concepts, while the PPPs directed learner attention only to the latter. While providing all participants tutorial-like instructions in PPPs, we vary the focus on interface mechanics by providing conditions either with blocks scrambled in an isolated palette, or with blocks scattered across palettes in categories native to Scratch (e.g. Motion, Events).

To investigate the effects on CT learning, we measure learner time-on-task, number of block moves, performance, and self-reported CL. Per CL theory, the brain provides limited short-term memory and processing capability along with infinite long-term memory, and learning occurs via schema construction and elaboration that leads to automation [38]. For learning to occur effectively, the CL of complex tasks should be reduced. Three dimensions comprise CL, however, and the reduction need not occur in all three. The total number of interacting elements perceived by the learner determines intrinsic load (IL); the sometimes-impeding presentation of the content determines extraneous load (EL); and the instructional features necessary for schema construction determine germane load (GL). Our study conditions with an isolated palette aim to reduce EL introduced via interface navigation to free learners' capacity to contend with GL. We also attempt to induce GL by requiring self-explanations after PPP play, like [39], since findings indicate self-explanation positively influences near and far transfer [40].

### 2.2 Distractors

Earlier research also has aimed to induce additional GL in PPPs by providing access to constructs not needed in solutions, called distractors. Example distractor types include: random constructs; unrelated control flow constructs; tangentially related constructs (TRCs); and partial suboptimal path distractors that might tempt a learner toward faulty progress without enabling her to solve the problem fully, thereby triggering misconception recognition and productive backtracking toward the correct solution [41]. Results indicate PPPs without distractors are the easiest to solve [42], and that more time and higher CL result when distractors are included, leading to lower learning efficiency [41]. To reduce distractor difficulty,

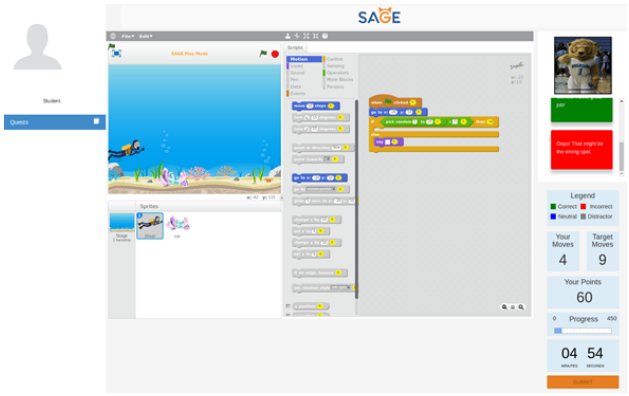


**Figure 1: PPP with selectively included palettes & blocks**

some have tried pairing them with correct constructs to increase GL by focusing learner attention on misconception-revealing differences between two solutions while reducing EL by eliminating the need to search for the relevant options amidst a jumble [43]. In our study we vary across conditions the inclusion of TRCs intended to help learners address misconceptions, and pair them close to correct constructs, with a goal of contributing insight into whether the efficiency lost by including distractors might be counterbalanced by the learning outcomes gained.

## 2.3 Feedback

Previous work has also explored increasing GL by including feedback mechanisms intended to guide learners to fix errors without providing the correct answer. These include line-based and execution-based feedback, as well as incremental progress indicators that introduce ambiguity to discourage trial-and-error behavior [41, 44]. These studies analyze usage patterns without conclusions about which feedback type is the better option, and without evaluating feedback's impact on learning efficiency independent of confounding variables. In our study, we provide feedback through a progress bar reflecting points scored, a count of block-moves made next to the minimum count needed to solve the puzzle, and correctness feedback on the position of each block placed, as shown in Figure 1. According to the feedback classification in [45], this feedback is constructivist since it is problem- and instance-oriented, which has been correlated with significantly lower student failure rates than alternative types such as those solution- and theory-oriented. We intend this feedback to be part of the landscape of interactions through which the learner constructs their understanding. For each variation of presentation and distractor disposition, we either activate or deactivate feedback, and then compare learning efficiency across conditions.

Previous research found that participants encountering distractors performed similarly on transfer tasks and exhibited decreased learning efficiency compared to those who did not [41]. When PPPs are evaluated in comparison to writing equivalent programs, they have been shown to take half as much time to complete without reducing performance on subsequent assignments [20]. To account for learners who compensate for an increase in mental load by committing more mental effort, thereby maintaining constant performance while load varies, researchers calculate instructional and performance efficiency [46]. The data recorded often include empirical estimates of mental effort during instruction (EI) and transfer (ET) tasks and the performance (P) on transfer tasks. The EI and P calculation measures the instructional efficiency of the learning process, while the ET and P calculation measures the performance efficiency of the learning outcome. For example, one study found that PPPs result in increased instructional efficiency compared to writing block-based code [16]; another indicated PPPs with randomly distributed distractors decrease performance efficiency [41]. Since we vary presentation, distractors, and feedback in ways that might modulate CL, and potentially performance on transfer tasks, these efficiency measurements operate as equalizers in our analysis.

# 3 SOFTWARE DEVELOPMENT

To investigate **R1-R3**, we modified Scratch to enable the design, play, and assessment of PPPs. Since user behavior can be affected by the programming environment [47], we aimed to preserve Scratch's strengths while integrating gameful elements like those in SQL-Tutor [48] and feedback systems similar to iSnap integrations which provide progress panels and adaptive messages [20, 49]. PPP gamification development and architecture are reported in [16, 50]. Here we document development that facilitates presentation, distractor, and feedback variation.

Having previously introduced an isolated palette to Scratch that enables the assembly of blocks from native Scratch palettes [16], we further modified the palette selector to allow the selective inclusion of palettes displayed during puzzle play as shown in Figure 2a. Similarly, within each palette, we afforded the selective inclusion of each block during puzzle design (Figure 2b), resulting in a play experience in which the learner might encounter a subset of Scratch palettes and blocks enabled for use. This functionality allows us to construct matching PPPs with and without an isolated palette (Figure 2c). For example, in a condition without an isolated palette and without distractors (Figure 1), we enable only the palettes and blocks that are included within the original PPP isolated palette. Consequently, the learner must explore the enabled palettes to identify block options, instead of choosing each block from one jumbled assortment. While this exploration might induce EL, it could bridge learning from PPPs to open-ended Scratch assignments since it encourages familiarization with the conceptually driven block organization. This approach relates historically to SP/k [51], which offered a sequence of language subsets (e.g. SP/1, SP/2) that introduced constructs incrementally; our learning system enables the design of palette and block subsets that learners encounter as they advance.

The selective inclusion of palettes and blocks also allows for distractors to be included or not. To provide an additional axis of variation, we configured a play mode with feedback systems disabled; the score-based progress bar, score value, number of moves, and correctness feedback, which includes alerts when distractors are used, are removed. We also remove the display of the minimum target moves needed to solve the puzzle, though this value remains discernable, since we also disable the puzzle submit button until

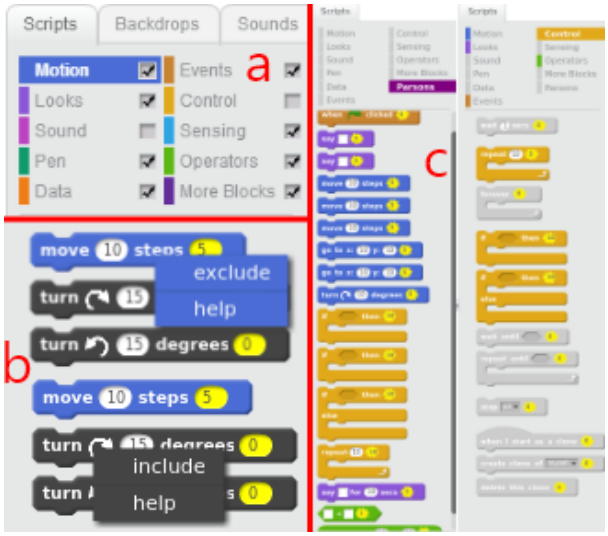


**Figure 2: Palette & block selective inclusion systems**

twice the number of minimum moves is made to encourage engagement but also to provide relief if a learner is flailing and ready to move to the next puzzle, similar to [52].

## 4 STUDY PURPOSE

This positioned us to study the effects of presentation, distractor, and feedback variation across conditions in a short intervention with a general adult population. One study purpose was to replicate findings of previous PPP studies that have typically targeted narrower populations, such as university or middle/high school students [18, 33]. If CT is a crucial competency for all as Wing evangelized in 2006 [1], it is important for the community to confirm pathways toward motivated and efficient learning for the general populace. In an era of remote schooling acceleration, children with parents who know CT themselves likely are advantaged over their peers [53]. Furthermore, given the crisis in quantity of CS teachers equipped to teach CT in schools [5, 54, 55], there remains a need to identify scalable professional development tactics for existing teachers so that they can help students engage in creative computing [56, 57]. Our study aims to provide insight into the applicability of the block-based PPP paradigm for adults with the implications for children deliberately considered.

A second study purpose was to identify PPP elements that maximize learning efficiency. Previous PPP studies have varied grading leniency and distractor positioning [58], variable naming conventions [59], distractor inclusion [41], execution and line-based feedback [44], and intra-problem and inter-problem difficulty adaptiveness [33]. Like the approach used in [60] in which the type of instructional support is varied in a programming game, we varied the instructional support provided to learners across PPP conditions. We include 8 conditions focused on the CT concept conditionals, plus one control on sequences, and analyze the effects of variation of presentation, distractors, and feedback, in the context of a series of 4 puzzles, each time-boxed for 8 minutes of solving, that are connected by a story line with explicit goals related to playful animations. Based on results discussed in section 2, our hypotheses associated with **R1-R3** were: **H1**) training with an isolated palette and without distractors yields lowest CL; **H2**) training with feedback yields highest CT motivation; **H3**) training with an isolated palette with feedback and without distractors yields the highest learning efficiency.

## 5 FORMATIVE EVALUATION

### 5.1 Design Thinking & Participants

To guide software development and reinforce construct validity, we conducted with middle-school teachers a formative evaluation in which we reviewed Scratch PPPs via a prototype using a design thinking approach. Design thinking is useful for investigating solutions to ill-defined problems framed in human-centric ways, often executed via a five-stage cycle: empathize, define, ideate, prototype, and test [61]. To empathize, we distributed a survey on the experience of teaching and learning with Scratch. To define, we conducted 1:1 semi-structured interviews in the style of [62]. To ideate, we facilitated sessions that iterated among open-ended idea elicitation, prototype demonstration, and discussion.

The participants included 21 teachers from informal learning venues such as codeHER and Girls Who Code, and 17 from U.S. schools. 11% had taught with Scratch for at least 2-4 years, 63% for 6-18 months, and 26% for less than 6 months. Our survey, data, and interview scripts are available at https://bit.ly/3rhdvzt.

### 5.2 Survey & Interview Results

A concerning survey result is the frequency with which students ask for help when starting in Scratch (66% often or always). To assist learners, 58% often or always provide direct instruction using several techniques: concept explanations (used by 58%); 1:1 tutoring (53%); demonstrations (47%). One way to reduce help-seeking would involve the incremental introduction of blocks; 92% would be willing or very willing to utilize such functionality.

We also explored teachers' interest in PPPs and found that 84% had never assigned them. After reviewing PPP descriptions, however, they were willing or very willing to assign PPPs (66%) and author PPPs (71%). Although this indicates enthusiasm for CT teaching through puzzle gameplay, in the semi-structured interviews, several expressed skepticisms about the educational efficacy of CT games they've assigned to students. One teacher raised concern about the lack of provision for student reflection: "A lot of the games hit algorithmic approaches, but they don't give them [students] the critical thinking moment, where they are like, why are we doing this, the real question, why, and what is my problem?". Although articulating the importance of reflection is non-trivial in the CS context as reported in [63], this teacher's approach entails designing small assignments prior to gameplay that introduce one block at a time and require students to explain their solutions. This feedback influenced the design of the selective palette and block inclusion system described in section 3 and led us to add a self-explanation pop-up at the conclusion of each puzzle in the spirit of the reflective learning journals reported in [64] to provoke reflection that strengthens learning.

**Table 1: Study protocol and data collected**

| # | Activity | Content | Data Collected |
|---|---|---|---|
| 1 | Registration | Credentials creation & condition assignment | Username & password |
| 2 | Background info | Demographics | Age, gender, education, country, programming experience & attitude, CT perceptions |
| 3 | Tutorial | 8-minute video on the learning system & *conditionals* | N/A |
| 4 | Pretest (isomorphic) | 10 multiple-choice *conditionals* questions | Pretest responses & score |
| 5 | CS CLCS | 10 CL questions with 0-10 scaled responses | Pretest CL & IL/EL/GL components |
| 6 | Puzzles | 4 puzzles on *conditionals* in 8 of 9 conditions; learning system behavior varies by condition; 4 puzzles on *sequences* in control condition | Per-puzzle time spent, time-stamped block moves & score, correctness, generated feedback, participant self-explanations |
| 7 | CS CLCS | 10 CL questions with 0-10 scaled responses | Puzzle CL & IL/EL/GL components |
| 8 | Posttest (isomorphic) | 10 multiple-choice *conditionals* questions | Posttest responses & score |
| 9 | CS CLCS | 10 CL questions with 0-10 scaled responses | Posttest CL & IL/EL/GL components |
| 10 | Concluding measurements | Motivation, programming attitude, learning system feedback, CT perceptions | TEQ & programming attitude, learning system viewpoints, CT perceptions |

**Table 2: Variation & participation across 9 study conditions**

| Cond. | CT Concept | Presentation | Distractors | Feedback |
|---|---|---|---|---|
| C1 | Conditionals | 1 palette | N | Y |
| C2 | Conditionals | 1 palette | Y | Y |
| C3 | Conditionals | Multi-palette | N | Y |
| C4 | Conditionals | Multi-palette | Y | Y |
| C5 | Conditionals | 1 palette | N | N |
| C6 | Conditionals | 1 palette | Y | N |
| C7 | Conditionals | Multi-palette | N | N |
| C8 | Conditionals | Multi-palette | Y | N |
| C9 | Sequences | 1 palette | N | Y |

# 6 SUMMATIVE EVALUATION

## 6.1 Study Design

The formative study helped prioritize development, refine learning materials, and organize a summative evaluation via a quantitative experiment between-subjects across conditions, with some within-subject measurements. As outlined in Table 1, participants: 1) created credentials in the learning system, which randomly assigned them to 1 of the 9 conditions documented in Table 2; 2) provided demographic detail; 3) reviewed an 8-minute introductory tutorial on the learning system and the CT concept of conditionals; 4, 8) took isomorphic pre/posttests; 5, 7, 9) self-reported CL after tests/training through a validated CS CL component survey (CS CLCS) [65]; 6) trained via 4 puzzles time-boxed for 8 minutes each; 2, 10) recorded CT perceptions and programming attitude via a Likert scale derived from categorized text responses by adults in [3] at study start/end; 10) responded to a validated intrinsic motivation Task Evaluation Questionnaire (TEQ) [66]. Materials are available at https://bit.ly/3pXjO9o.

## 6.2 Materials & Participants

We developed 4 *conditionals* puzzles and reused 4 on *sequences* from a previous study [16]. Following guidance in [18], we aimed to design motivating scenarios with memorable segments while providing a challenge without being tricky and leaving the participants with a positive impression. To familiarize them, we included in the instructions for the first puzzle the solution and block-use descriptions. When including distractors, we targeted misconceptions to introduce cognitive conflict and reinforce learning as in [67]. For example, an if-else block operates as a distractor in a puzzle in which only two if blocks are needed.

To create the pre/posttests, we followed the 7-step approach to developing and validating CS knowledge assessments in [68]. To test and refine our materials, we collaborated with 8 undergraduates with diverse majors and previous exposure to CS. Tests included trials of the survey content, pre/posttests, and puzzles, and think-alouds in which the tester would interact with the puzzles while verbalizing her thoughts. Although we did not further formally assess validity and reliability, these results led to refinements such as reduced ambiguity in puzzle instructions and eliminated pre/posttest questions deemed too easy or difficult.

Using Amazon Mechanical Turk and Prolific [69, 70], we recruited 624 participants with varying degrees (43% high school, 35% undergraduate, 17% graduate), and a variety of self-reported programming experience (low: 52%; medium: 34%; high: 14%). 396 men, 221 women, and 2 non-binaries comprise the population sourced from 34 countries led by the U.S. (22%), Poland (21%), and the U.K. (13%). While practical, this recruitment introduces risk to external validity, as there could be an element of self-selection in the sample, since those participating in online studies might be more adept at computing than the general populace.

### 6.3 Data Collection & Processing

We measured CL, performance, learning efficiency, and motivation by creating 7 surveys and instrumenting the learning system to: 1) record puzzle play duration; 2) trace each block moved; 3) calculate score using an algorithm inspired by the longest common subsequence approach described in [67] and documented in [16] that results in higher scores when nearest to the solution. We calculated instructional and performance efficiency using time-on-task and CL for the EI and ET values during training and transfer tasks as described in section 2. Since the data did not exhibit Shapiro-Wilk normality ($p<0.05$), we used non-parametric statistics, like Kruskal-Wallis H and Mann-Whitney U tests between-subjects, and Wilcoxon signed-rank test within-subjects. For effect sizes, we used guidelines in [71].

## 7 ANALYSIS & RESULTS

### 7.1 Cognitive Load

As expected, given random condition assignment, we did not find significant differences between conditions in self-reported CL during the pretest. Kruskal-Wallis tests did reveal significant differences between training conditions with moderate effect for IL ($H(8)=26.24$, $p<.001$, $\epsilon^2=.04$), EL ($H(8)=24.15$ $p=.002$, $\epsilon^2=.04$), and overall CL ($H(8)=29.10$ $p<.001$, $\epsilon^2=.05$). Using a Bonferroni-adjusted alpha of .006 (.05/9), we found significant differences between conditions C6 (M=5.6) vs. C9 (M=3.95) p=.037, C8 (M=5.76) vs. C9 p=0.20, and C1 (M=4.22) vs. C8 p=0.31 for IL, C1 (M=3.03) vs. C6 (M=4.19) p=.015 for EL, and C1 (M=4.32) vs. C6 (M=5.50) p=.002, C1 vs. C8 (M=5.36) p=.016, and C3 (M=4.45) vs C6 (M=5.40) p=0.22 for overall CL. The comparatively low mean IL value for C9, the control condition, suggests that participants accurately perceived fewer interacting elements in the puzzles focused on the CT concept *sequences* compared to those on *conditionals*. The IL C1 vs. C8 result corresponds with the conditions that provide the most and least scaffolding (C1: 1 palette, no distractors, feedback; C8: multi-palette, distractors, no feedback). The overall CL result for C1 vs. C8 further reveals the impact of these supports and provides evidence partially supportive of H1, indicating PPP learning systems can induce significantly lower IL and CL by activating maximum rather than minimum scaffolding.

Since the remaining noted differences varied on two axes of support each, we ran Mann-Whitney tests on condition sets for each axis and for all conditions compared to the control. We discovered significant differences with small effect related to distractors and feedback, as well as further evidence of the lower IL induced in the control compared to the treatment ($U(N_{control}=60, N_{treatment}=564)=19{,}990.5$, $z=2.32$, $p=.021$, $r=.09$, $M_{control}=3.95$, $M_{treatment}=4.92$). Participants who experienced distractors reported significantly higher IL ($U(N_{distractors}=301, N_{no\text{-}distractors}=263)=46{,}408.0$, $z=3.54$, $p<.001$, $r=.15$, $M_{distractors}=5.34$, $M_{no\text{-}distractors}=4.43$), EL ($U(N_{distractors}=301, N_{no\text{-}distractors}=263)=93{,}820.5$, $z=4.56$, $p<.001$, $r=.19$, $M_{distractors}=3.95$, $M_{no\text{-}distractors}=3.13$), and overall CL ($U(N_{distractors}=301, N_{no\text{-}distractors}=263)=47{,}873.5$, $z=4.30$, $p<.001$, $r=.18$, $M_{distractors}=5.16$, $M_{no\text{-}distractors}=4.52$), then those who did not. This suggests distractors may have been perceived sometimes as interwoven into the challenge, and sometimes as hampering focus.

Additionally, participants who experienced feedback reported with small effect significantly lower IL ($U(N_{feedback}=298, N_{no\text{-}feedback}=266)=34{,}771.0$, $z=-2.52$, $p=0.012$, $r=.11$, $M_{feedback}=4.60$, $M_{no\text{-}feedback}=5.27$), GL ($U(N_{feedback}=298, N_{no\text{-}feedback}=266)=35{,}511.5$, $z=-2.14$, $p=0.033$, $r=.09$, $M_{feedback}=5.91$, $M_{no\text{-}feedback}=6.30$), and overall CL ($U(N_{feedback}=298, N_{no\text{-}feedback}=266)=33{,}663.5$, $z=-3.09$, $p=.002$, $r=.13$, $M_{feedback}=4.62$, $M_{no\text{-}feedback}=5.12$) than those who did not. The lower IL value indicates correctness feedback might reduce the number of interacting elements perceived, due to the focus driven by prompt feedback on each block move. The lower GL value, while attractive, could represent risk that learners lean on feedback without sufficiently learning conditionals to use them in feedback's absence. We did not, however, find the lower GL negatively affected efficiency in the analysis for section 7.3. Combined, these training CL results provide only partial support for **H1**, as evidence suggests training without distractors and with feedback produces the lowest CL (**F1**); we found no significant difference from varying presentation (1-palette vs. multi-palettes).

During the posttest, there were no significant differences between individual conditions, but participants who trained with distractors reported with small effect significantly higher IL ($U(N_{distractors}=301, N_{no\text{-}distractors}=263)=43{,}811.5$, $z=2.19$, $p=.028$, $r=.09$, $M_{distractors}=4.30$, $M_{no\text{-}distractors}=3.80$), and EL ($U(N_{distractors}=301, N_{no\text{-}distractors}=263)=44{,}431.0$, $z=2.52$, $p=.012$, $r=.11$, $M_{distractors}=2.57$, $M_{no\text{-}distractors}=2.26$) than those who did not. These results could indicate less schema construction occurred during training for distractor than no-distractor participants.

### 7.2 Performance

Though we did not find significant training performance differences across all conditions when measuring the number of puzzles solved correctly, the time needed to solve them was significantly different with moderate effect ($H(8)=87.94$, $p<.001$, $\epsilon^2=.14$). After the Bonferroni adjustment, we found participants in the control on *sequences* completed training significantly faster than those in each *conditionals* condition while solving nearly the same number of puzzles correctly ($M_{control}=2.0$, $M_{treatment}=1.9$). Since solving the 4 *sequences* puzzles requires a minimum of 41 block moves compared to 33 for the *conditionals* puzzles, this result suggests the complexity of the CT concept and puzzle, rather than the length of the solution, affect time spent solving.

Other condition pairs with significant differences in time spent are: C1 vs. C4 (p=.000), C1 vs. C7 (p=.040), C4 vs. C5 (p=.000),

and C5 vs. C7 (p=.012). Participants in C1 and C5, both with an isolated palette and without distractors, needed less time than participants in C4 with multiple palettes and distractors, and C7, with multiple palettes ($M_{C1\text{-}time}$=16.6m, $M_{C5\text{-}time}$=16.1m, $M_{C4\text{-}time}$=20.8m, $M_{C7\text{-}time}$=22.9m). However, C4 and C7 participants solved more puzzles correctly than those in C5, and C7 participants outperformed those in C1 ($M_{C1\text{-}correct}$=1.9, $M_{C5\text{-}correct}$=1.5, $M_{C4\text{-}correct}$=1.8, $M_{C7\text{-}correct}$=2.7). To further investigate these correctness differences, we removed the control from the between-conditions calculations, and found a significant difference with moderate effect across just treatment conditions (H(7)=91.70, p=.000, $\epsilon^2$=.16). The condition-pair result remaining significant after Bonferroni adjustment is C5 vs. C7 (p=.020). This finding suggests that when the presentation varies while distractors and feedback variation are held constant (deactivated), multi-palette participants solve more puzzles correctly than those with an isolated palette. However, we note that despite the random condition assignment, substantially fewer completed the experiment in C7 compared to the mean across conditions ($M_{C7}$=37, $M_{C1\text{-}9}$=69). This discrepancy represents a risk to internal validity, as it is possible some participants found the lack of support in C7 too challenging, and dropped their participation, leaving in the condition population a more CT-adept assembly.

We further pursued this finding via Mann-Whitney tests on the group of conditions with an isolated palette vs. the group with multiple palettes, but did not quite find a significant correctness difference ($U(N_{1\text{-}palette}$=317, $N_{multi\text{-}palette}$=247)=35,609.0, z=-1.89, p=.059, r=.08, $M_{1\text{-}palette}$=1.8, $M_{multi\text{-}palette}$=2.0). We did, though, see significant differences with small effect in the time solving between these two groups, with those with multiple palettes spending 13% more time than those with an isolated palette ($U(N_{1\text{-}palette}$=317, $N_{multi\text{-}palette}$=247)=31,545.0, z=-3.96, p<.001, r=.17, $M_{1\text{-}palette}$=17.4m, $M_{multi\text{-}palette}$=19.7m). A similar time-spent difference exists between the groups of conditions with distractors vs. those without, with distractor participants spending 7% more time than those without distractors ($U(N_{distractors}$=301, $N_{no\text{-}distractors}$=263)=45,798.0, z=3.22, p=.001, r=.14, $M_{distractors}$=19.0m, $M_{no\text{-}distractors}$=17.8m), but differences in correctness are limited between these groups, with participants without distractors solving marginally more puzzles correctly on average in less time.

During the transfer phase, participants in each treatment condition solved more posttest than pretest questions correctly. As a full treatment population, they did so at a significant level with a small effect within-subject per a Wilcoxon test (z=3.4, p<.001, r=.15, pretest: $M_{treatment}$=7.8, posttest $M_{treatment}$=8.0). We found the largest increase in pre/posttest scores of M=.4 from participants who trained in C3 (multi-palette, no distractors, with feedback), indicating that goal-driven search through multiple palettes and guidance from correctness feedback might lead to the greatest performance increase in multiple-choice transfers tasks. However, the difference in pre/posttest scores between treatment conditions was not significant (H(7)=2.66, p=.92, $\epsilon^2$=.00), meaning further experimentation would be required to vet that possibility. The lack of transfer performance disparity between PPP conditions generally replicates findings in [16, 41], and is similar to findings on PPP inter-problem and intra-problem adaption in [33].

## 7.3 Efficiency

Given the training duration differences between the *sequences* and *conditionals* puzzles, and the dependency on standardized values in the EI, ET, and P instructional efficiency (IE) and performance efficiency (PE) calculations introduced in 2.3, we focus efficiency analysis on differences discovered between treatment conditions. Though we did not find any significant PE differences, when using time spent as an estimate of mental effort during training, we found a significant difference with a moderate effect in IE (H(7)=26.32, p<.001, $\epsilon^2$=.04), with condition pairs C1 vs. C4 (p=.003) and C4 vs. C5 (p=.001) remaining significant after the Bonferroni adjustment. This indicates that C5 and C1, with the two highest IE values ($M_{C5}$=.25, $M_{C1}$=.21), and each with an isolated palette and no distractors, led to more efficient learning than C4 ($M_{C4}$=-.31), with multiple palettes, distractors, and feedback. This result provides partial support for **H3**. The low mean IE value for C7 ($M_{C7}$=-.59) suggests that training with multiple palettes without distractors and without feedback also degrades learning efficiency, though the C5 vs. C7 (p=.102) and C1 vs. C7 (p=.307) pairwise comparisons were not significant, due to the smaller participant population in C7 noted in section 7.2. Nonetheless, these results offer evidence that multi-palette PPPs decrease IE compared to isolated-palette PPPs.

We also compared groups of conditions along the axes of variation, which led to no PE findings, but did reveal significant IE differences for each axis. The group of conditions with an isolated palette, with small effect, resulted in significantly higher IE than those with multiple palettes ($U(N_{1\text{-}palette}$=317, $N_{multi\text{-}palette}$=247)=44,503.0, z=2.79, p=.005, r=.08, $M_{1\text{-}palette}$=.11, $M_{multi\text{-}palette}$=-.13), which adds support to the IE finding favorable to an isolated palette between individual conditions. We also found, with small effect, significantly higher IE for the no-distractor group compared to the distractor group when measuring mental effort during training both by time ($U(N_{distractors}$=301, $N_{no\text{-}distractors}$=263)=34,019.0, z=-2.88, p=.004, r=.12, $M_{distractors}$=-.06, $M_{no\text{-}distractors}$=.07) and by CL ($U(N_{distractors}$=301, $N_{no\text{-}distractors}$=263) =33,228.5, z=-3.29, p=<.001, r=.14, $M_{distractors}$=-.11, $M_{no\text{-}distractors}$=.13). Lastly, we analyzed the condition sets with feedback activated vs. deactivated, and found that activated feedback led to substantially higher, but not quite significant, IE when measuring mental effort via CL ($U(N_{feedback}$=298, $N_{no\text{-}feedback}$=266)=42,922.5, z=1.70, p=.080, r=.07, $M_{feedback}$=-.00, $M_{no\text{-}feedback}$=-.08).

These results indicate IE is higher when training involves an isolated palette, no distractors, and feedback when compared to multiple palettes, distractors, and no feedback. They nearly fully support **H3**, but since the feedback result did not reach significance, the F3 finding for maximum efficiency excludes it to focus on an isolated block palette and no distractors. The distractors finding aligns with [41], which similarly found decreased learning efficiency in PPPs with distractors. To the best of our knowledge, PPP learning efficiency of isolated or multiple palettes and feedback activation or deactivation have not previously been reported in the literature.

## 7.4 Motivation

**Table 3: Selected within-subject programming attitude change**

| # | Programming is... | Feedback ON | Distractors OFF | 1-palette ON |
|---|---|---|---|---|
| 1 | fun | M=.49** | M=0.56** | M=0.4** |
| 2 | enjoyable | M=.59** | M=0.6** | M=0.58** |
| 3 | important to know | M=.33** | M=0.34** | M=0.37** |
| 4 | easy to start | M=.77** | M=0.97** | M=0.79** |
| 5 | too difficult to understand | M=.58** | M=0.62** | M=0.49** |
| 6 | boring | M=.26** | M=0.33** | M=0.24** |
| 7 | too time consuming | M=.21* | M=0.32** | M=0.32** |

Negative valence (5-7) calculated with (AFTER -BEFORE)*-1, all others (1-4) with AFTER-BEFORE. *p<.05, **p<.01

*7.4.1 Quantitative Results.* To analyze motivation quantitatively, we scored the TEQ and calculated within-subject change in programming attitude and CT perception from study start to end. 3 participants from the TEQ and 7 from the attitude/perceptions tasks were excluded due to participant response insufficiencies ($N_{TEQ}$=621, $N_{attitude/perception}$=617). Though there were not significant differences in TEQ results between conditions for the interest/enjoyment, pressure/tension, and perceived choice subscales, there were with moderate effect for the perceived competence subscale (H(8)=24.78, p=.002, $\epsilon^2$=.04). After Bonferroni adjustment, the condition pair remaining significant was C1 (M=4.65) vs C6 (M=3.63) p=.018, meaning when the presentation (isolated palette) and distractor status (deactivated) were held constant, feedback activation produced higher perceived competence, which supports **H2**.

When comparing programming attitude and CT perception change across sets of conditions, we found significant differences in each of the axes of variation. Participants in the condition set with feedback reported with small effect significantly higher increase in the Likert scale response to the statement: I believe I could successfully learn computational thinking (U($N_{feedback}$=295, $N_{no\text{-}feedback}$=263)=42,777.5, z=2.17, p=.030, r=.09, $M_{feedback}$=.27, $M_{no\text{-}feedback}$=.10). This result supports **H2** and **F2** and might suggest feedback has positive implications for CT self-efficacy. Those in the condition set with an isolated palette with small effect reported significant larger decrease in response: programming is too time consuming (U($N_{1\text{-}palette}$=314, $N_{multi\text{-}palette}$=244)=42,243.5, z=2.14, p=.033, r=.09, $M_{1\text{-}palette}$=.32, $M_{multi\text{-}palette}$=.09), which reflects the 13% less time spent in training noted in 7.2. Lastly, those in the condition set without distractors with small effect reported significantly higher increase in response to: programming is easy to start (U($N_{distractors}$=296, $N_{no=distractors}$=262) =34,347, z=-2.38, p=.017, r=.10, $M_{distractors}$=.61, $M_{no\text{-}distractors}$=.97), indicating that distractors induced perceptions of higher difficulty.

Additionally, we found many significant positive attitude changes from study start to end within-subject per condition set; selected results are presented in Table 3. Although these results indicate attitude improvement and support **H2**, the absence of longitudinal data represents a threat to internal validity, since we cannot claim the change at study conclusion persists.

*7.4.2 Qualitative Results.* To supplement the quantitative results, we sought qualitative feedback by requesting that participants describe their attitude toward programming after the learning experience, as well as their perspective on the puzzle presentation, distractors, and feedback. Especially for those who self-reported low prior programming experience, we recorded more positive attitudinal outcomes for participants who trained with feedback, supportive of **H2** and **F2**. Several reflected a sense of empowerment: 1) "I believe that programming is a complex topic to begin learning without help, it is simple in its basic form, but I think that with enough practice and will to learn, anyone can learn programming"; 2) "it seems much more accessible now"; 3) "it seems like something that I actually could do if I took the time to learn starting out with the basics it is not only for those who are innately gifted or technologically advanced, I can code as well." 4) "I always thought that programming was something that no "normal" person could do out of the blue. This proved me wrong."; 5) "it seems more interesting and maybe more accessible - before it felt as if only "chosen ones" could do it well."

Others appreciated the motivational and gameful effects: 1) "I think this is a great way to get you excited and interested in programming... this would be even enjoyable for the kids and in the meantime they unconsciously learn coding."; 2) "The puzzles were actually very enjoyable, it felt like a game; 3) "I loved it. I think I will continue learning more about programming by enrolling in other programming courses."; 4) "I find it a lot easier right now. I wanted to start it for some time but I'm a little bit lazy but this learning experience might give me the boost to get into it finally"; 5) "I thoroughly enjoyed doing this. I thought it was all going to be white data on black screen. I really liked the step by step colourful simplicity of the learning process."; 6) "It's made me want to have another go at it as this was such a good way to learn rather than in lectures I had many years ago where it was just slide after slide and then trying stuff... I think this especially would be a great way to get kids interested in programming in school as it showed how it can be fun and satisfying without being overly complex. I really enjoyed this!; 7) "The instant gratification of seeing the products of your efforts on-screen spurs you on"; 8) "It's very addictive and fun:)".

For those with multiple palettes, we found more hesitant perspectives: 1) "programming scares me a bit less now"; 2) "I think it was a great but exhausting experience"; 3) "I realized just how much I didn't know about programming, and how difficult it can

be. I would have never guessed I'd struggle as much as I did, but I consider it a valuable experience." Those who trained with feedback, however, noted its positive effect on motivation and efficiency: 1) "i liked receiving feedback, it was like a teacher guiding me, if it did not verify anything i had to search by myself"; 2) "feedback was good as meant i wasnt too far gone past a mistake before correcting it"; 3) "i was excited that i got it right and sometime feel not putting enough effort if i got i wrong. so i put more effort in getting it right."; 4) "On the first puzzle I had some red boxes pop up so I knew which moves I had done wrong and I could change things around until I got confirmation from the feedback that it was correct. It helps to know you're on the right track otherwise it might feel you're just endlessly trying things without really knowing if it's correct or not."

Those who did not receive feedback reported using more traditional testing techniques: 1) "I tried to verify whether I made the correct move by clicking the green arrow and seeing my animation based on the moves I made."; 2) "I tried to play the animation with the green flag and see if that gives feedback."; Feedback was not appreciated by all, however, as small set noted discomfort: 1) "it made me a little anxious when my moves were incorrect."; 2) "when I received information that my movement was incorrect, I got stressed"; 3) "I was happy when I saw that I made the correct choice, however, when thy system turned to red and it said that something is incorrect I began to feel anxiety and I was angry at myself because I did not know what I did wrong and I just didn't understand why it's wrong and that bothered me."; 4) "I was worried and tried to understand what I did wrong that it says it was incorrect and how I can make it work." These distressed feedback perceptions will lead us to explore throttled feedback options to limit disruption to those negatively affected.

The motivational impact of distractors was mixed. Some valued the simplicity of an isolated palette and lack of distractors: 1) "Having a restricted amount of pieces to work with makes things easier."; 2) "I liked not seeing any distractor blocks it made me more efficient and could keep me focused."; 3) "I would have easily been more confused if additional blocks were presented to me. I am very thankful that wasn't the case for me because I probably wouldn't have accomplished anything." Others embraced the challenge distractors offered: 1) "Distractors are helpful in my opinion, because once selected it trigger the though "ok why is this not okay?" and then you would compare it with the other blocks understanding the mistake."; 2) "If there were only "correct" blocks it would be more easyer and this is not good, because to learn programming you don't make only "correct" things, you need to make some mistake to understand what are you really doing."; 3) "Enabled blocks which weren't part of the puzzle solution helped me judge and differentiate between the relevant conditionals that were required to solve the puzzle."; 4) "I liked that there were some similar blocks that were not used, because that allowed me to learn about different scenarios where I might prefer one block over another in the future. . . That way I can simultaneously learn about a few different blocks instead of only the ones I am directly using."; 5) "I thought that having some "dummy" blocks to throw me off the scent was really effective, and would help a younger learner to focus more on the problem."

**Table 4: PPP element variation findings summary**

| Element | CL | Efficiency | Motivation |
|---|---|---|---|
| 1-palette | - | Increase** | Increase** |
| Distractors | Increase** | Decrease** | Decrease** |
| Feedback | Decrease* | Increase | Increase** |

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

## 7.5 Findings Summary

We conclude the analysis by summarizing findings for each varied PPP element in Table 4

# 8 CONCLUSION & FUTURE WORK

Our survey of, and interviews with, grade 6-9 teachers revealed the substantial teacher support necessary to help CT learners in Scratch, as well as their limited prior exposure to, but willingness to try, PPPs. This led us to extend Scratch with gameful PPP functionality focused on individual CT concepts. By varying elements of PPPs across nine conditions in a study of 624 adults, we found PPPs with feedback and without distractors produce lowest CL, PPPs with feedback produce highest CT motivation, and PPPs with an isolated block palette and without distractors produce highest CT learning efficiency. The study analysis offers PPP developers insight to advance efficient CT education for all.

While these results expose opportunities to advance CT learning via augmentations to popular block-based programming environments like Scratch, we recognize external validity would be strengthened were additional CT concepts studied, and other block-based environments included. We might also strengthen construct validity by evaluating how fading of supportive PPP elements within learning progressions helps learners transition toward deepening CT understanding in open-ended projects. In future work, we intend to address these issues while also exploring how block-based environments might further equip PPP content developers with tools to customize and differentiate auto-generated feedback for learners. With continued study across CT concepts, functionality, and populations, we aim to identify paths towards reliably efficient, effective, and equitable CT learning.

# ACKNOWLEDGMENTS

The Programming Systems Lab is supported in part by DARPA N6600121C4018, NSF CCF-1815494 and NSF CNS-1563555.

# REFERENCES

[1] J. Wing. Computational Thinking. Comms. of the ACM 2006.
[2] C. Schulte, M. Knobelsdorf. Attitudes towards computer science-computing experiences as a starting point and barrier to computer science. ACM ICER, 2007; pp 27-38.
[3] P. Charters, *et al.* Challenging stereotypes and changing attitudes: the effect of a brief programming encounter on adults' attitudes toward programming. ACM SIGCSE, 2014; pp 653-658.
[4] D. Ravitch. Reign of error: the hoax of the privatization movement and the danger to America's public schools; Vintage, 2013.
[5] C. Wilson, *et al.* Running on Empty: The Failure to Teach K-12 CS in the Digital Age; CSTA, 2010.
[6] N. Selwyn, *et al.* High-tech, hard work: an investigation of teachers' work in. Learning, Media and Technology 2017.
[7] L. A. DeLyser. A community model of CSforALL. ITiCSE, 2018.

[8] C. Lewis, *et al.* Alignment of goals and perceptions of computing predicts students' sense of belonging in computing. ITiCSE, 2019.
[9] E. Aivaloglou, F. Hermans. Early programming education and career orientation. ACM SIGCSE, 2019; pp 679-685.
[10] J. Wang, *et al.* Gender differences in factors influencing pursuit of CS and related fields. ACM ITiCSE, 2015.
[11] J. McBroom, *et al.* Understanding gender differences to improve equity in computer programming education. ACE, 2020; pp 185-194.
[12] OECD. Enrolment by field. https://stats.oecd.org/.
[13] J. B. Bush, *et al.* Drag and drop programming experiences and equity. ACM SIGCSE, 2020; pp 664-670.
[14] D. J. Malan, H. H. Leitner. Scratch for budding computer scientists. ACM SIGCSE Bulletin 2007, 39 (1).
[15] J. Maloney, *et al.* The Scratch programming language and environment. ACM Transactions on Computing Education 2010, 4.
[16] J. Bender, *et al.* Integrating Parsons Puzzles with Scratch. ICCE, 2021.
[17] D. Parsons, P. Haden. Parson's programming puzzles: a fun and effective learning tool for first programming courses. Australian Conference on Computing Education, 2006.
[18] K. J. Harms, *et al.* Enabling independent learning of programming concepts through programming completion puzzles. IEEE VL & HCC, 2015; pp 271-279.
[19] B. J. Ericson, *et al.* Solving parsons problems versus fixing and writing code. Koli Calling CER, 2017.
[20] R. Zhi, *et al.* Evaluating the effectiveness of parsons problems for block-based programming. ACM ICER, 2019.
[21] P. J. Rich, *et al.* Coding in K-8. AECT Tech Trends 2019, 63 (3).
[22] M. M. McGill, A. Decker. Tools, languages, and environments used in primary and secondary computing education. ACM ITiCSE, 2020.
[23] L. Zhang, J. Nouri. A systematic review of learning computational thinking through scratch in K-9. Computers & Education 2019, 141.
[24] C. Scaffidi, C. Chambers. Skill progression demonstrated by users in the Scratch animation environment. International journal of Human-Computer Interaction 2012, 28 (6), 383-398.
[25] I. Harel, S. Papert. Constructionism; Ablex Publishing, 1991.
[26] Y. Dong, *et al.* Defining tinkering behavior in open-ended block-based programming assignments. ACM SIGCSE, 2019.
[27] K. Brennan, M. Resnick. New frameworks for studying and assessing the development of computational thinking. AERA, 2012.
[28] D. Franklin, *et al.* Scratch Encore. ACM SIGCSE, 2020.
[29] J. Salac, *et al.* TIPP&SEE: a learning strategy to guide students through use-modify Scratch activities. SIGCSE, 2020.
[30] M. Tsur, N. Rusk. Scratch microworlds. ACM SIGCSE, 2018.
[31] K. A. Brennan. Best of both worlds: issues of structure and agency in computational creation, in and out of school; Dissert.; MIT, 2013.
[32] J. Garner, *et al.* Mastery learning in CS education. ACE, 2019.
[33] B. J. Ericson, *et al.* Evaluating the efficiency and effectiveness of adaptive parsons problems. ICER, 2018.
[34] Y. Du, *et al.* A review of research on parsons problems. ACE, 2020.
[35] E. Bubacher, *et al.* Where the rubber meets the road. Transfomring Learning, Enabling Learners, 2016.
[36] D. Weintrop, *et al.* Block-based comprehension: exploring and explaining student outcomes from a read-only block-based exam. ACM ITiCSE, 2019; pp 1218-1224.
[37] K. J. Harms, *et al.* Learning programming from tutorials and code puzzles. IEEE VL & HCC, 2016; pp 59-67.
[38] J. Sweller. Cognitive Load Theory; CU Press, 2010.
[39] R. K. Atkinson, *et al.* Transitioning from studying examples to solving problems. Educational Psychology 2003.
[40] R. Moreno, *et al.* Optimizing worked-example instruction in EE: role of fading & feedback during problem-solving practice. EE 2009.
[41] K. J. Harms, *et al.* Distractors in parsons problems decrease learning efficiency for young novice programmers. ICER, 2016.
[42] S. Garner. An exploration of how a technology-facilitated part-complete solution method supports the learning of computer programming. Informing Science & IT, 2007.
[43] P. Denny, *et al.* Evaluating a new exam question: parsons problems. ACM ICER, 2008; pp 113-124.
[44] J. Helminen, *et al.* How do students solve parsons programming problems?–execution-based vs. line-based feedback. Learning and Teaching in Computing and Engineering, 2013.
[45] G. Raubenheimer, *et al.* Toward empirical analysis of pedagogical feedback in computer programming learning environments. ACE, 2021; pp 189-195.
[46] T. van Gog, F. Paas. Instructional efficiency: revisiting the original construct in educational research. Ed. Psychologist, 2008; pp 16-26.
[47] I. Karvelas, *et al.* The effects of compilation mechanisms and error message presentation on novice programmer behavior. ACM SIGCSE, 2020; pp 759-765.
[48] F. Tahir, *et al.* Investigating the effects of gamifying SQL-Tutor. ICCE, 2020; pp 416-425.
[49] S. Marwan, *et al.* Adaptive immediate feedback can improve novice programming engagement and intention to persist in computer science. ACM ICER, 2020; pp 194-203.
[50] J. Sulaiman, *et al.* SAGE-RA: A Reference Architecture to Advance the Teaching and Learning of Computational Thinking. Embedding AI in Education Policy and Practice for Southeast Asia, 2019.
[51] R. C. Holt, *et al.* SP/k: a system for teaching computer programming. Communications of the ACM 1977, 20 (5), 301-309.
[52] A. Kumar. Epplets. ACM SIGCSE, 2018.
[53] C. Sepúlveda-Díaz, *et al.* Lessons learned from introducting preteens in parent-led homeschooling to CT. ACM SIGCSE, 2020.
[54] L. Zhang, *et al.* Progression of CT skills in Swedish compulsory schools with block-based programming. ACE, 2020; pp 66-75.
[55] B. Munasinghe, *et al.* Teachers' understanding of technical terms in a computational thinking curriculum. ACE, 2021; pp 106-114.
[56] A. Lamprou, A. Repenning. Teaching how to teach computational thinking. ACM ITiCSE, 2018; pp 69-74.
[57] P. Haduong, K. Brennan. Helping K–12 teachers get unstuck with scratch. ACM SIGCSE, 2019; pp 1095-1101.
[58] A. N. Kumar. Helping students solve Parsons puzzles better. ACM ITiCSE, 2019; pp 65-70.
[59] A. N. Kumar. Mnemonic variable names in Parsons puzzles. ACM CompEd, 2019; pp 120-126.
[60] R. Zhi, *et al.* Exploring instructional support design in an educational game for K-12 computing education. ACM ITiCSE, 2018.
[61] R. Buchanan. Wicked problems in design thinking. DI 1992.
[62] S. Kvale. InterViews: An Introduction to Qualitative Research Interviewing; SAGE: Thousand Oaks, CA, 1996.
[63] J. W. Coffey. A study of the use of a reflective activity to improve students' software design capabilities. SIGCSE, 2017.
[64] J. Whalley, H. Ogier. Learning journals in creative programming assessments: exposing bugs, issues, and misconeptions. ACE, 2020.
[65] B. B. Morrison, *et al.* Measuring cognitive load in introductory CS: adaptation of an instrument. ICER, 2014; pp 131-138.
[66] Self-Determination Theory. https://selfdeterminationtheory.org/intrinsic-motivation-inventory/.
[67] V. Karavirta, *et al.* A mobile learning application for parsons problems with automatic feedback. Koli Calling CER, 2012.
[68] P. S. Buffum, *et al.* A practical guide to developing and validating computer science knowledge assessments with application to middle school. ACM SIGCSE, 2015.
[69] Amazon. Amazon Mechnical Turk. https://www.mturk.com/.
[70] Prolific. Prolific. https://www.prolific.co/.
[71] C. O. Fritz, *et al.* Effect size estimates. Experimental Psych. 2012.