To Read or to Rotate? Comparing the Effects of Technical Reading Training and Spatial Skills Training on Novice Programming Ability

Madeline Endres
endremad@umich.edu
University of Michigan, CSE
USA

Madison Fansher
mfansher@umich.edu
University of Michigan, Psychology
USA

Priti Shah
priti@umich.edu
University of Michigan, Psychology
USA

Westley Weimer
weimerw@umich.edu
University of Michigan, CSE
USA

ABSTRACT

Understanding how to best support and train novice programmers is a critical component of producing better and more diverse software engineers. In this paper, we present the results of a controlled 11-week longitudinal study with 57 CS1 students comparing two skill-based interventions to improve programming performance. The first intervention involves spatial training, an established baseline known to be helpful in engineering contexts. The second intervention is a novel CS-focused technical reading training.

In our reading training, we teach strategies for summarizing scientific papers and understanding scientific charts and figures; most of the covered readings were CS1-accessible portions of computer science research papers. For the spatial training, we use a standardized training curriculum previously found to improve programming skills by focusing on spatial ability (i.e., the ability to mentally manipulate objects). We first replicate findings that both reading ability and spatial ability correlate with programming success. Significantly, however, we find that those in our reading training exhibit larger programming ability gains than those in the standard spatial training ($p = 0.02$, $f^2 = 0.10$). We also find that reading trained participants perform particularly well on programming problems that require tracing through code ($p = 0.03$, $f^2 = 0.10$). Our results suggest that technical reading training could be beneficial for novice programmers. Finally, we discuss the implications of our results for future CS1 interventions, the possibility for non-programming based training to positively impact developers, and future directions for software engineering education research.

CCS CONCEPTS

• **Social and professional topics** → **Software engineering education**; **CS1**.

KEYWORDS

Transfer Training, CS1, Technical Reading, Spatial Ability

ACM Reference Format:
Madeline Endres, Madison Fansher, Priti Shah, and Westley Weimer. 2021. To Read or to Rotate? Comparing the Effects of Technical Reading Training and Spatial Skills Training on Novice Programming Ability. In *Proceedings of the 29th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE '21), August 23–28, 2021, Athens, Greece.* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3468264.3468583

1 INTRODUCTION

For many aspiring software engineers, successfully completing a first computer science course (CS1) is an essential step toward a technology-focused career. In recent years, the software engineering research community has demonstrated an increased interest in both supporting novice programmers [18, 28, 51, 67] and understanding what cognitive skills are most important for software engineering success [18, 22, 32, 59, 60]. However, CS1 students often struggle [4, 36], and less affluent students with weaker preparatory education struggle disproportionately [46]. This may contribute to a lack of diversity endemic in many modern software workplaces [52]. In this paper, we investigate software engineering training methods for novices at the university level. The techniques in this paper combine insights from cognitive studies of software engineering with CS pedagogy through the comparison of two skills-based CS1 interventions: spatial ability training and technical reading training. Specifically, *we propose a novel computer science focused technical reading training* for novice software engineers, and we *compare its effectiveness to that of an established spatial ability curriculum.*

One reason under-prepared novice programmers struggle is that they may have less incoming facility with cognitive skills helpful for software engineering. Students' incoming math, reading, and spatial abilities correlate with CS1 success [6, 21, 34]. Such skills also relate to key software engineering tasks such as data structure manipulation [32] and code review [22]. Consequently, one potential intervention for supporting struggling novices is facilitating *learning transfer* to programming from a concurrently taught supplemental training course [7], where learning transfer refers to the use of skills in a separate context from where they were learned.

ESEC/FSE '21, August 23–28, 2021, Athens, Greece

ACM ISBN 978-1-4503-8562-6/21/08…$15.00
https://doi.org/10.1145/3468264.3468583

One of the most common skill-based training interventions proposed and tested for STEM education is spatial ability training [7, 15, 65, 70]. Spatial ability refers to the ability to mentally manipulate objects, and it is a cognitive skill directly connected with problem solving [31], a necessary component for software engineering expertise [74]. Supplemental training designed to help improve spatial ability has been shown to both improve outcomes in several key engineering classes and also increase engineering degree retention [63]. There is also evidence that spatial ability training directly improves novice programming outcomes [7, 15]. The connection between spatial skills and software engineering is amplified by the finding that tree-based data-structure manipulation and spatial ability problems activate similar brain regions [32].

We observe, however, that there are many aspects of software that are not obviously spatial. It is possible, therefore, that technical reading ability may sometimes be more important for initial software engineering success. Technical reading ability is a key component of software engineering. For instance, programmers generally spend significant time switching between reading project specifications and documentation [13]. Similarly, over half of developers consider readability to be the most important property of code [61, Sec. 4.1], time spent reviewing code is among the top three factors for assessing productivity and is more important than time spent writing code [43, Fig. 2], and managers and developers rank readability in the top three development analytic [9, Fig. 4]. At a cognitive level, recent medical imaging studies have found that with more programming expertise, brain activation patterns while reading code are more similar to patterns while reading prose [18, 22], and that language aptitude can explain a significant portion of learning outcomes for programming course participants [48].

In this paper, we propose a novel CS-focused technical reading training, and we use an 11-week randomized trial with 57 participants to compare the efficacy of two CS1 interventions: our novel technical reading training and an established standardized spatial ability training. Our proposed technical reading training is CS-focused to better cover CS-specific reading tasks, such as reading API documentation. Reading transfer training interventions have been carried out in other fields, but they are rare for STEM education. To the best of our knowledge, we are the first to test for transfer between technical reading ability and computer science.

We find that reading training participants had significantly larger programming gains than those in the spatial training ($p = 0.02$). We also find that the reading training's benefit is most significant for tasks that require tracing code. Furthermore, we observe that reading ability tends to correlate more strongly with final programming ability than spatial ability. This paper's contributions include:

- A novel CS-focused technical reading intervention.
- Results of an 11-week randomized control trial comparing the effects of our reading training and an established spatial training on novice programming ability.
- The finding that reading training participants exhibit significantly larger CS1 programming gains than spatial participants, primarily during code-tracing tasks.
- A replication of prior findings that both spatial ability and reading ability correlate with programming performance.
- An analysis of cognitive and demographic features predictive of CS1 programming ability.
- A discussion of the implications of our findings on CS1 education and software engineering in general.

2 RELATED WORK

In this section, we give a brief summary of relevant research. We define technical reading and discuss reading training and transfer. We then outline connections between software engineering, program comprehension, and reading ability. Finally, we define spatial ability and describe its connections with computer science.

Technical Reading Ability: Closely related to general reading ability, *Technical Reading Ability* is a person's ability to read and understand technical or scientific texts [55]. Both reading and technical reading can be predictive of success in various fields. For instance, when investigating elementary school students' facility with math word problems, Vilenius-Tuohimaa *et al.* found that both technical and general reading ability were strongly related to student success [71]. Similarly, reading was found to be the second most important academic area for predicting success in nursing school behind science and ahead of mathematics [76].

There is evidence that technical reading training may contribute to improved performance in various fields. For instance, a writing-intensive technical reading course improved biology students' perceived understanding of primary scientific papers and their ability to communicate science [8]. A large body of work explores how to teach reading (see Grabe for a summary [25]). Grabe proposes key guidelines for developing reading curricula, including "emphasizing vocabulary learning," "promoting strategic reading," "activating background knowledge in appropriate ways," and developing "intrinsic motivation for reading" [25]. To both increase student motivation and also to deliberately promote transfer, we develop our own CS-focused technical reading training (see Section 3.1).

Technical Reading and Computer Science: There is evidence that reading ability correlates with programming success [26, 34, 39, 44]. For instance, Leeper *et al.* found that, while effort and comfort level were the strongest predictors of success, verbal SAT scores were more predictive than math SAT scores of computer science course grades [34]. Furthermore, Shute found that scores on algebra word problems predicted programming skill acquisition [58].

Others connect programming to reading or tracing through a program and describing its function in natural language [39, 41, 44]. That is, there is a positive correlation between verbally contextualizing programs "in plain English" and programming ability. Mayer found that "asking learners to put technical information in their own words (through making comparisons) results in broader learning" that supports transfer to computer science [41]. Furthermore, Fedorenko *et al.* argue that learning to program is akin to learning a language and thus teaching programming can be "informed by pedagogies for developing linguistic fluency" [20].

Beyond direct connections between programming ability and reading ability, many essential software engineering tasks, such as code review, code summarization, or managing documentation, require technical reading ability. For example, professional software developers use a significant portion of their time on the web reading documentation and debugging forums [77]. Furthermore, software developers often use prose summaries of code, either in the form of documentation or inline comments, to facilitate program comprehension and communication with other programmers [23]. Techniques for automatically generating prose summaries of code reduce time spent writing documentation, keep up with rapidly changing code bases, and aid developer comprehension [27, 42].

Programming and reading also relate on a cognitive level. Siegmund *et al.* identified five brain areas that activate when reading programs, most of which are also associated with language [59]. Similarly, Floyd *et al.* found that while code review and prose review were distinct neurological processes, they became less distinct with more programming experience [22]. This pattern was further supported by Endres *et al.*, who directly compared visualization, reading, and programming with novices at the cognitive level [18].

Researchers have also explored reading and program comprehension using eye-tracking. For example, Busjahn *et al.* used eye-tracking results to argue for using natural language reading as a basis for understanding code comprehension [11]. Rodeghero *et al.* found that, when possible, programmers use keywords to capture high-level program information rather than reading code details [54]. Similarly, Rodeghero and McMillan found that developers prefer to skim code and follow common reading patterns [53]. Furthermore, Busjahn *et al.* found that developers generally read code non-linearly, a pattern that increases with expertise [10]. These results inform the design of our reading training (see Section 3.1).

In an educational context, Gunawardena *et al.* observed "there may be a stable correlation between student's ability to successfully comprehend and detect errors in a document and their course performance" [26]. Similarly, Hebig *et al.* conducted focus groups with masters students discussing their perceptions of various software comprehension techniques, finding generally positive views of many reading-related comprehension aides such as systematic code reading, documentation, and analysis dashboards [29]. These lines of preliminary results invite a more controlled study.

We are not aware of any previous studies directly exploring transfer training between technical reading and programming ability. A main contribution of this paper, therefore, is the development and evaluation of the first (to our knowledge) CS-focused technical reading software engineering intervention.

Spatial Ability and Computer Science: *Spatial ability* refers to a person's capacity to understand and reason about spatial relationships among objects, and is a blanket term for skills such as mental rotation, mental folding, spatial perception, and spatial pattern recognition [40]. There is a large body of research on spatial ability and its correlates. Generally, spatial ability correlates with gender and socioeconomic status [12, 72]; on average, men have higher spatial ability than women, and spatial ability and affluence are positively correlated. Spatial ability positively correlates with performance in a variety of fields including mathematics [30, 73], natural sciences [5, 69, 78], and engineering [1].

Several studies have found evidence of a medium to strong positive correlation between spatial ability and programming [21, 33, 45, 47]. For instance, Parker *et al.* found that spatial ability was a better mediating variable for socioeconomic gaps in computer science than access to computing [45]. Additionally, Parkinson and Cutts found that "spatial skills typically increase as the level of academic achievement in computer science increases" [47].

Beyond correlational studies, researchers have used various methods to connect spatial ability with programming. For example, Huang *et al.* used medical imaging to find that similar parts of the brain are recruited to solve mental rotation problems and tree-based data structure problems, indicating that core software engineering tasks use visiospatial cognitive processes [32]. Additionally, Margulieux proposed "spatial encoding strategy" (SpES), a theory about the cognitive processes behind the transferability between spatial ability and programming [40]. SpES posits that improving spatial ability helps develop general strategies for encoding oriented "mental representations of non-verbal information," strategies that can be applied to programming problems. We hypothesize that such strategies also apply to software engineering problems.

Spatial ability is malleable, meaning it improves through training; a meta-analysis [70] found that spatial training increases performance on a variety of tasks. For example, Pribyl *et al.* found that spatial training improved organic chemistry students' ability to "mentally manipulate two-dimensional representations of a molecule" [49]. Also, spatial training may reduce, or even eliminate, observed gender gaps in spatial ability [35]. The important correlates with spatial ability coupled with its malleability encourage us to consider a spatial intervention to help train software engineers.

Recently, there have been two studies establishing causality between spatial ability training and computer science success. Cooper *et al.* ran an exploratory study with 38 high school programmers, finding that those who participated in spatial training performed better on a final programming assessment [15]. Significantly, Bockmon *et al.* replicated the Cooper study at four different universities with a much larger sample size ($n = 345$) [7]. They found that on a final programming assessment, introductory students who participated in a spatial training intervention significantly outperformed students who did not participate. All participation in the spatial intervention was voluntary, leaving open the possibly of self-selection biases affecting their results. Even so, the study provides compelling evidence for the feasibility of spatial training transferring to CS1, and thus software engineering, performance.

Due to established correlation with computer science outcomes and evidence of transfer to programming ability, we use spatial training as one of our two skills-based interventions. In our experiment, spatial training functions as a baseline for analyzing the efficacy of our reading treatment. Both Cooper *et al.* and Bockmon *et al.* used spatial training materials based on interventions created by Sorby and Baartmans [64], materials that have also been found to improve performance in general engineering classes [63, 65]. We also use Sorby and Baartmans's materials.

3 TRAINING MATERIALS

In this section, we discuss the structure and content of the two interventions. In Section 3.1, we present our novel CS-focused technical reading intervention (Reading Treatment), and in Section 3.2 we describe the standardized spatial training (Spatial Treatment).

3.1 Technical Reading Training

For the Reading Treatment, we designed a novel computer science focused technical reading curriculum. We had several key design goals. First, for the purposes of direct comparison with the Spatial

Treatment (see Section 3.2), we required that our Reading Treatment consist of nine two-hour weekly training sessions and include workbook practice problems, group work, and physical props — snap-blocks for the Spatial Treatment and vocabulary flashcards for the Reading Treatment. We also required that the Reading Treatment use active learning techniques such as think-pair-share, group work, and in-class practice problems, as these have been found to enhance learning [50]. Finally we required our training be CS-focused to increase participant engagement and programming relevance. To accomplish this, we integrated computer science research papers and API documentation into the course.

Overall Session Structure: Each session followed the same general lesson plan: a 20-minute ice-breaker and workbook warm-up, general (not CS specific) vocabulary, a short lecture covering a technical reading strategy, and finally reading and analysis practice. This practice typically accounted for half of each training session.

Warm-up and Vocabulary: The initial ice-breaker and workbook warm-up served to get students on task and excited for the rest of the session [17]. For the workbook, students completed pages from *Reading Comprehension Skills and Strategies: Level 8*, a commercially-available workbook [56]. We included vocabulary practice due to the established benefit of emphasizing vocabulary when teaching reading [25]. Vocabulary words were selected from a curated list of common GRE words. Generally, we emphasized words that we believed were prevalent in technical or scientific writing. For the lecture portion, we focused on teaching established technical reading strategies.

Lecture Topic Selection: During the lecture portion, we covered various technical reading strategies. The majority of these strategies focused on using structural cues to quickly and accurately scan texts to retrieve and understand key points. Topics included focusing on outlines when reading to improve comprehension, understanding figures and charts in scientific writing, and strategies for understanding persuasive technical proposals (e.g., Heilmeier's Catechism). Our focus on teaching structural cues and patterns to efficiently skim texts was motivated by findings that experienced programmers tend to read code non-linearly, focusing on high level features [10, 53].

CS-Paper Selection: All but two of the reading and analysis practice sections involved computer science research papers selected using several criteria. First, papers were selected from the reading lists of various software engineering and computer science education courses at a top-tier public university. All papers were published in reputable computer science journals or conferences. Second, to ensure the material was understandable for CS1 students, we deliberately selected accessible research papers that were interesting to first year students; in particular, many readings involved computer science education so the content was relatable. Finally, for more complex readings, students were only asked to read a curated subset of the paper such as the introduction, related work, and conclusion. We asked students to summarize readings and to write and share short reading responses; previous work has found that asking learners to put technical information in their own words results in increased comprehension and broader learning for transfer [41]. The other two sessions contained general review (roots and affixes) or API documentation reading strategies.

Training Materials: For replication purposes, we have made our reading training materials publicly available in our replication package.[1] This replication package contains session slide decks and a list of all lecture topics and all CS-papers covered. We also provide study recruitment materials and data analysis scripts.

3.2 Spatial Training

For our Spatial Treatment, we used a spatial ability course developed at Michigan Technical University by Sorby and Baartmans [64]. Intended to help incoming engineers improve their spatial ability, Sorby and Baartmans's curriculum has been shown to improve not only students' spatial skills [65], but also their grades in several key engineering courses [63]. The course consists of 10 modules and covers the following topics: surfaces and solids of revolution, combining solid objects, isomorphic sketching, orthographic sketching, orthographic projections, flat pattern folding, 3D shape rotation around single and multiple axes, object reflection or symmetry, and mental cutting. Provided teaching materials included training videos, software, and individual practice workbooks.

We taught the material in Sorby and Baartmans's curriculum in nine two-hour weekly sessions. Each session involved showing the provided lecture videos, working through the software in pairs, and completing workbook problems. We also gave each participant snap-blocks, as recommended by the provided training materials, to help them visualize challenging workbook problems. To ensure student engagement, we held the training sessions in-person on campus until COVID-19 necessitated remote instruction. We discuss the impact of COVID-19 on our experiment in more detail in Section 4.

4 EXPERIMENTAL DESIGN

In this section, we cover our experimental design and protocol.

Design Overview: To compare the effects of our CS-focused technical reading training and spatial training,[2] we conducted an 11-week controlled study with 57 participants. Participants were enrolled in the same CS1 course at the University of Michigan during Spring 2020, and they were randomly assigned into either the Spatial Treatment or the Reading Treatment. These two Treatments required participants to make the same weekly time commitment for the same number of weeks, and all participants were compensated the same monetary amount. Participants were not aware of their Treatment group when they took the pre-test. We measured training effectiveness using a validated CS1 assessment [46].

Both Treatments had nine two-hour in-person weekly training sessions with the same instructors. The study started in the the fourth week of term and lasted until the end of the semester. Participants had to complete at least 6/9 training sessions to be included in our analysis. Participants also attended two additional two-hour assessment sessions, a pre-test the week before the start of the study and a post-test the week after the last training session. At the two assessments, participants took spatial ability, reading ability, and programming ability tests. They also completed a demographics questionnaire during the pre-test and a qualitative survey during the post-test. We describe the assessments in detail in Section 5.

[1] Replication package: https://github.com/CelloCorgi/FSE2021_To_Read_or_to_Rotate.

[2] While we initially speculated that the Spatial Treatment might be more effective than the Reading Treatment due to historical results, we focus our discussion and framing on the Reading Treatment's effectiveness given our surprising results.

Design Motivation: An alternate study design for evaluating our reading training's effectiveness would have been to compare programming gains of the participants to those in a "no treatment" group (i.e., students in the same CS1 course who were not in the reading treatment). We choose instead to compare between two interventions: our CS-focused technical reading training and an extant spatial training of the same duration and intensity. The main factor behind this choice was to mitigate self-selection bias. In studies of educational interventions with supplementary instruction, self-selection into the treatment group can introduce significant bias, even after controlling for demographics factors [16, 38]. Thus, we designed our study such that participants first self-selected into the study and only then were they randomly assigned into a treatment group.

We also considered comparing both interventions against a weaker placebo course as a control, but rejected it as participant drop-out rate (and thus our potential statistical power) was a real concern. We hypothesized that drop-out would be significantly higher with any control appearing unrelated to CS. Finally, we note that we emailed a post-test invitation to all participants who were pre-tested but later dropped out of the study to form a potential additional no-treatment comparison group. Unfortunately, likely due to COVID-19 and exam timing, only five participants responded. While we observed lower programming gains in this group than those in either treatment, five responses is not enough to support meaningful analysis.

Participant Recruitment: Participants were recruited from the same CS1 course at the University of Michigan, a large public university. A prerequisite for declaring a computer science major, the course is in C++ and Python. We recruited using email, forum posts, and in-class presentations. Participants had to be over 18, be able to attend at least six sessions, have no prior programming experience, and only be enrolled in one programming course. Recruitment occurred during the third week of the fifteen-week semester.

Out of 736 students in CS1, 187 students completed the pre-screening. Of those, 151 met the eligibility requirements, and 97 completed the pre-test. These 97 participants were then assigned randomly to one of the two Treatments. For each two-hour session attended, participants were compensated $20 in cash ($220 for participants who attended all 11 weeks). Ultimately, 57 valid participants (29 in the Reading Treatment and 28 in the Spatial Treatment) completed at least 6/9 training sessions and took the post-test, a 58.7% study completion rate for pre-test takers. Four additional students who took the post-test and attended the requisite number of training sessions were eliminated due to invalid post-test scores; all four rushed through assessments, completing them faster than two standard deviations below the mean. We note that the majority of participant drop-out occurred in weeks four and five of the study, coinciding with intensification of the COVID-19 pandemic (see Section 4). Table 1 contains demographic breakdowns of the final 57 participants.

Table 1: Profile of study participants by Treatment type. We only include racial and ethnic categories with at least 5 study participants. The pre-test scores are all average raw scores.

	Spatial	Reading
Total Participants	28	29
Female Participants	23	17
Male Participants	5	12
Native English Speakers	21	20
Native Chinese Speakers	4	8
White / Caucasian	10	10
Asian / Pacific Islander	15	12
Hispanic American	2	3
Pre-test PFT (out of 20)	13.1	15.1
Pre-test PSVT:R II (out of 30)	17.5	21.6
Pre-test GRE (out of 25)	8.0	9.6
Pre-test Programming (out of 12)	3.1	2.8

Population Homogeneity: We also checked for differences between our Treatment populations. While we assigned participants randomly and all participants had no prior programming experience, one group could have greater reading, spatial, or programming skills than the other, yielding a potential advantage in CS1. We find no significant differences between the incoming reading scores or the incoming programming scores of the two Treatments. We do, however, find a significant difference between incoming spatial ability on both spatial assessments (PFT: $p = 0.01$, PSVT:R II: $p = 0.004$). The average raw pre-test scores are presented in Table 1. We account for this in our mathematical analyses (e.g., by considering programming gains, see Figure 3) as well as in Limitations and Threats to Validity (see Section 9).

To ensure participants did not have prior programming experience, we checked for programming ability during recruitment and pre-screening. During recruitment, we emphasized that participants could have no prior experience including minimal exposure to textual or visual programming languages such as Scratch. During pre-screening, participants indicated if they "had any prior programming experience" with either "Yes," "No," or "Other/unsure." Respondents were only included in the study if they chose "No" outright. Potential participants also indicated previous course enrollment from a list. Students in courses with "programming" or "code" in their description were excluded. Finally, we observe that, on our programming pre-test, participants scored 24.6%, only slightly more than the 20% expected with random guessing.

COVID-19 Adjustments: Our experiment was concurrent with the COVID-19 pandemic, and our institution suspended in-person research activities during the study's fifth week. Therefore, we moved the remainder of the training sessions and the post-test to a virtual format; students attended a proctored two-hour video conference each week in lieu of the in-person training session. They then emailed scans of their completed work to the research team. For the post-test, the participants took online Qualtrics versions of each assessment. Like the virtual sessions, the virtual post-test was proctored over a video-conference. Beyond session format, the COVID-19 epidemic affected study retention. During the last week with full in-person sessions before university-wide COVID-19 measures, the study had 82 attendees, while in the first full-week of virtual sessions, the study had only 61 participants, a 25.6% drop.

5 EXPERIMENTAL INSTRUMENTS

In this section, we describe instruments used during the study's pre- and post-tests to collect demographics and assess participants' spatial, reading, and programming abilities.

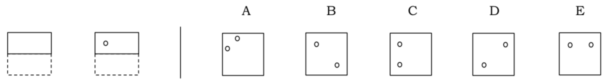


Figure 1: PFT: Participants select the choice which corresponds to the paper on the left unfolded. Answer is "C."

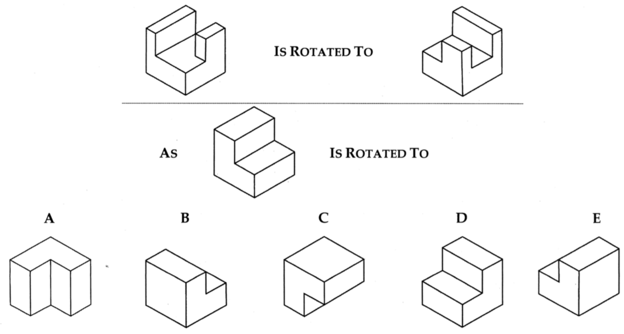


Figure 2: PSVT:R II: Participants select the shape that is rotated in the same way as the top shape. Answer is "D."

Spatial Ability Assessments: We administered two spatial ability assessments: the *Paper Folding Test* (PFT) [75] and the *Revised Purdue Spatial Visualization Test* (PSVT:R II) [79]. Both are validated standard assessments of different facets of spatial ability. The PFT measures mental folding and contains 20 multiple choice questions split into two halves, each timed at three minutes. Participants took the PFT during the pre- and post-tests. An example problem is shown in Figure 1. The PSVT:R II measures mental rotation and contains 30 multiple choice questions of increasing difficulty, and takes 20 minutes. Participants took the PSVT:R II during the pre- and post-tests. An example problem is shown in Figure 2.

Reading Assessment: To assess reading ability, we gave students verbal sections of the *Graduate Record Examination* (GRE) [19], a required exam for admission to many graduate programs. We chose verbal sections from the official GRE practice test, using sections from the old (pre-2010) GRE format to minimize the chance of participants taking a test they had seen before. Each GRE verbal section had 25 multiple choice questions and a 35 minute time limit. The test includes vocabulary questions and reading comprehension questions. Students were randomly assigned one of two GRE verbal sections during the pre-test and took the other GRE section during the post-test to avoid re-testing effects. We checked for difficulty differences between the two test versions by comparing the means and medians of both versions' pre- and post-test scores. We did not find any significant difference in difficulty between the two versions. We note that the GRE is a general measure of reading ability rather than a measure of CS-focused technical reading. While there is a CS version of the GRE, it primarily measures programming ability rather than technical reading. To the best of our knowledge, there does not yet exist a validated measure of CS-focused reading.

Programming Assessment: To assess programming ability, we used the Second CS1 Assessment (SCS1) developed by Parker *et al.* [46]. Written in pseudocode, the SCS1 is a validated language-agnostic measure of CS1 programming ability. It is also the same programming assessment used by Bockmon *et al.* for evaluating their spatial training intervention [7]. An updated version of the FCS1 [66], the SCS1 consists of 27 multiple choice questions evenly divided into three question types: definition questions which ask students to recall the function of a programming construct, trace-based questions which require students to mentally walk through code before choosing the answer, and code-completion questions which ask students to choose a code snippet that causes a given program to have a specified behavior when inserted into a specified program location. The SCS1 has a one-hour time limit. Along with the full-length version, Parker *et al.* created a shorter 12-question subset. Due to students' limited initial programming ability, we used this shortened version during the pre-test and had participants complete the full SCS1 during the post-test.

Demographic and Qualitative Instruments: We also collected a variety of demographic and qualitative data. During the pre-test, students took a demographic survey for gender, race, ethnicity, native language, intended major(s), and computing attitudes. We also included a version of the Financial Affluence Survey III, a validated measure estimating socioeconomic status [68]. During the post-test, we had participants self-report anticipated CS1 course grades and other measures of programming ability. We also asked students for feedback on the Treatments including which modules they found the most challenging or helpful and if they thought that the training helped them in CS1. Finally, we asked for short-answer reflections on their experiences in the study.

6 TRAINING EFFECTIVENESS VALIDATION

We first analyzed the effectiveness of both Treatments in their respective domains. Especially due to COVID-19, it is important to verify the interventions' effectiveness. We used a multiple regression approach, with pre-test score as a predictor to control for individual variation in spatial and reading ability, and included Treatment Group as a predictor as well.[3]

We observed main effects of pre-test score for all three measures (PSVT:R, PFT, and GRE) suggesting that pre-test scores significantly predicted participants' post-test scores (see Table 2). We did not, however, find significant differences between Treatment Groups for the PSVT:R, PFT, nor the GRE Reading Assessment at post-test (see Table 2); we would expect a main effect of Treatment Group for each of these regression analyses. While it was surprising that our interventions did not appear to improve spatial ability nor GRE reading scores, closer examination of the data makes clear why we do not see significant improvements. For one, the students in our study had relatively high spatial ability. For example, in the study upon which we base our spatial ability training [63], the mean PSVT:R pre-test score of intervention participants was 52%, while the average pre-test PSVT:R score of our participants was 65%. A one-sample t-test comparing our pre-test data on the PSVT:R to a .522 suggested that our pre-test scores were significantly higher than those in the original study ($t(54) = 4.99, p < .001$).

Furthermore, our reading intervention could target different skills than those assessed by the GRE. We note that the GRE is a

[3] For all regressions, we report (1) the regression coefficient B, the direction and strength of the relationship between the predictor and dependent variable; and (2) the uncertainty around that estimate $SE(B)$, the outcome of a t-test $t(x)$ with x degrees of freedom, and the significance level of this test.

Table 2: Training Validation: Regression analysis on the effectiveness of the training interventions.

Post-test Score Predictor	Estimate (B)	SE(B)	t(54)	P-value
PSVT:R II				
Intercept	0.29	0.07	4.14	<0.001
Pre-test	0.63	0.09	6.87	<0.001
Training Type	0.01	0.04	0.24	0.810
PFT				
Intercept	0.16	0.07	2.24	0.030
Pre-test	0.85	0.09	9.43	<0.001
Training Type	−0.01	0.03	−0.33	0.750
GRE				
Intercept	0.18	0.04	4.15	<0.001
Pre-test	0.02	< 0.01	6.35	<0.001
Training Type	−0.04	0.04	−1.13	0.270

general assessment of reading ability and potentially not the best measure of skills taught in a CS-focused Reading Treatment. To the best of our knowledge, there does not yet exist a validated measure of CS-focused reading ability that does not also test programming ability. We hope the efficacy of our proposed Reading Treatment may help facilitate the future development of such a tool.

7 EXPERIMENTAL RESULTS

We now present our analysis of the data collected during the experiment described in Section 4. This experiment compares two potential CS1 skill-training interventions: Spatial Ability Training and our CS-focused Technical Reading Training. We center our investigation around the following five questions:

- **RQ1**—Efficacy: Did Reading Treatment participants perform better than Spatial participants on the final programming test?
- **RQ2**—Question Type: Were the effects the Treatments more pronounced for some programming question types than others?
- **RQ3**—Subpopulation: Were participant subpopulations better supported by either Treatment?
- **RQ4**—Correlation: Which participant demographic- and skill-based traits were most predictive of, and/or correlated with, programming success regardless of Treatment?
- **RQ5**—Perception: How do participants subjectively describe their experiences in both Treatments?

In this paper, we use the phrase *Final Programming Outcomes* to refer to scores on the final programming assessment.

Notes on Statistical Methods: Most statistics were conducted with R statistical software, primarily using the `lm` function for regression analysis. All analysis scripts, including tests for violations of statistical assumptions and graph generation with `ggplot2`, are available in the replication package

For each of the first three research questions, we specified multiple linear regression models. We specified separate multiple linear regression models for each question containing only the variables that we report with the results for each research question. For example, the model for RQ1 contains pre-test programming scores (to control for participant variation) and participant Treatment Group. We did not specify an overarching model between research questions that includes all of the individual differences explored in RQ4 for two reasons. First, there is no effect of these variables on treatment effectiveness (as shown in RQ3, see Section 7.3). Second, adding them would increase model complexity in RQ1 and RQ2 and possibly introduce a multicollinearity issue [2], as confirmed by the variance inflation factors we observed for each model.

Beyond multiple linear regression models, we also compute a correlation matrix for RQ4 and perform chi-square tests for RQ5. To measure the effect size of our results, we compute Cohen's f^2 [14]. Finally, we note that correction for multiple statistical tests is necessary when multiple dependent variables are used to search for an effect on at least some variable as a function of an independent variable. As a result, we employ false discovery rate correction when calculating the correlation matrix for RQ4. For RQ1, however, we have only one dependent variable of interest. Similarly, RQ2 acts as a breakdown of this dependent variable and thus serves to provide more information about what students learn. As a result, it is not necessary to control for multiple comparisons in RQ1 and RQ2. Finally, RQ3 and RQ5 contain no statistically significant tests even without correction, making further correction unnecessary.

7.1 RQ1: Treatment Efficacy

To answer our main question of whether the reading training would improve programming abilities, we compared Final Programming Outcomes between participants in the Spatial and Reading Treatments. Using a multiple regression model with pre-test programming score to control for individual variation and Treatment Group as predictors, we found a significant main effect of pre-test score ($B = .37$, $SE(B) = .14$, $t(54) = 2.68$, $p < .01$), suggesting that participants' pre-test score predicted their post-test score.

More importantly, we also found a significant main effect of Treatment Group ($B = -0.09$, $SE(B) = 0.14$, $t(54) = -2.33$, $p = .02$): the students in the Reading Treatment performed significantly better on the post-test programming test than students in the Spatial Treatment (see Figure 3). This is our primary result, and it indicates that technical reading ability may facilitate programming for novice software engineers more than spatial ability in some cases. We discuss further implications of this result in Section 8.

> Reading Treatment participants perform better than Spatial Treatment participants on Final Programming Outcomes ($p = 0.02$), a small effect ($f^2 = 0.10$).

7.2 RQ2: Treatment Effects by Question Type

In this section, we analyze if the Reading Treatment's benefit varies based on programming question type. As mentioned in Section 5, the SCS1 has three types of questions: definitional, tracing, and code-completion [46]. We computed a multiple regression model for each type of question, with pre-test on the subtype questions and Treatment Group as predictors in order to determine if the Reading Treatment significantly improved more on specific question subtypes in comparison to the Spatial Treatment. We found

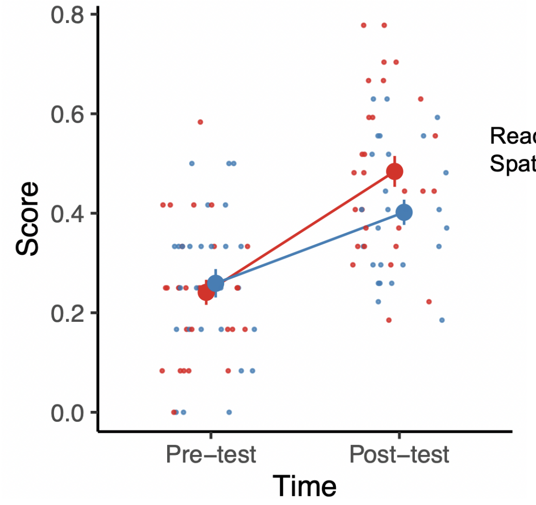


Figure 3: Percentage Programming Gains by Treatment: Reading Treatment participants exhibit significantly larger gains than Spatial Treatment participants.

evidence that the Reading Treatment improved more on the tracing questions than the Spatial Treatment ($B = -1.04, SE(B) = 0.45, t(54) = -2.31, p = .03$), a small sized effect ($f^2 = 0.10$). However, the Reading Treatment did not improve more on the definitional ($B = 0.41, SE(B) = 0.23, t(54) = 1.76, p = .33$) nor code-completion ($B = .27, SE(B) = .32, t(54) = -1.61, p = 0.11$) questions (see Figure 4). We consider hypotheses for the effect of the Reading Treatment on Tracing questions in Section 8.

> The Reading Treatment was especially helpful for code-tracing questions ($p = 0.03$, $f^2 = 0.10$). There were no significant performance differences on definitional ($p = 0.33$) nor code-completion ($p = 0.11$) questions.

7.3 RQ3: Participant Subpopulation Effects

In this section, we analyze variations in both Treatments' effectiveness by participant sub-population. To avoid spurious effects due to multiple comparisons, we performed a limited number of sub-population analyses. In particular, we considered incoming spatial ability, incoming reading ability, native language, and gender. To examine whether one treatment was more effective than the other for different subpopulations, we ran a series of multiple regression analyses. In each analysis we consider post-test programming score as the dependent variable and include the individual difference of interest, pre-test programming score, Treatment Group, and the individual difference × Treatment Group interaction as predictors.

If one treatment were more effective for improving programming scores than another for a specific sub-population, we would expect to see a significant interaction. However, we were unable to see any variations in Treatment effectiveness by subpopulation for any of the models (see Table 3). This is not surprising given the sizes of our subpopulations. It is difficult to analyze individual difference data without a large sample size. Future work might replicate the current study with a larger sample size to determine if sub-population variation in treatment effectiveness exists.

Table 3: Subpopulation analysis: results of regression tests for individual differences.

Interaction: Treatment Group * X	Estimate (B)	SE(B)	t(53)	*p*-value
Incoming PSVT:R II	−0.04	0.20	−0.18	0.86
Incoming PFT	−0.27	0.24	−1.12	0.27
Incoming Reading	−0.06	0.19	−0.30	0.76
Native Language	−0.13	0.08	−0.30	0.94
Gender	−0.09	0.09	−0.99	0.33

> We did not find evidence that variation in the effectiveness of the Treatments existed by participant sub-population when considering incoming spatial ability, incoming reading ability, native language, and gender.

7.4 RQ4: Treatment-Agnostic Correlations

We also analyze which skills-based and demographic features across both Treatments predict Final Programming Outcomes using correlations to capture how assessment scores relate to each other. Figure 5 contains a Spearman's rank correlation matrix for pre- and post-test scores and select demographic features for all 57 study participants.[4] In particular, it contains the spatial ability measures (PFT and PSVT:R II), reading test (GRE), programming test (SCS1), gender, and native language. We report only significant correlations that pass false discovery rate correction ($p < 0.05$, $q < 0.05$).

Perhaps unsurprisingly, the pre- and post-tests for both spatial measures were all positively correlated ($0.63 < r < 0.78$). We note that cross-assessment (PFT vs. PSVT:R II) correlations trend lower than pre- post-test correlations on the same assessment. The pre- and post-test reading scores also correlate ($r = 0.62$). We do not observe any significant correlations between either spatial test with the reading assessment.

We also observe that all reading and spatial assessments other than the pre-test PFT correlated significantly with Final Programming Outcomes (see row 8 in Figure 5). The correlations with post-tests for reading and PSVT:R II were particularly strong, both with $r \geq 0.50$. This indicates that while reading and spatial ability are separate skills, they are both important for software engineering. We also note that the PSVT:R II tended to correlate more strongly with coding than the PFT. It appears that mental rotation is more connected to introductory programming than mental folding.

Additionally, we observe a trend that reading may correlate more with programming than spatial ability; reading was both the most-correlated pre-test and the most-correlated post-test with Final Programming Outcomes ($r = 0.47$ and 0.56 respectively), and when considering only code-replacement questions, only reading was significantly correlated with programming. This trend may indicate that reading is more predictive of software engineering success than spatial ability. However, replication with larger statistical power

[4] We chose Spearman's rank correlation instead of Pearson's because Spearman's captures both linear and non-linear monotonic relationships.

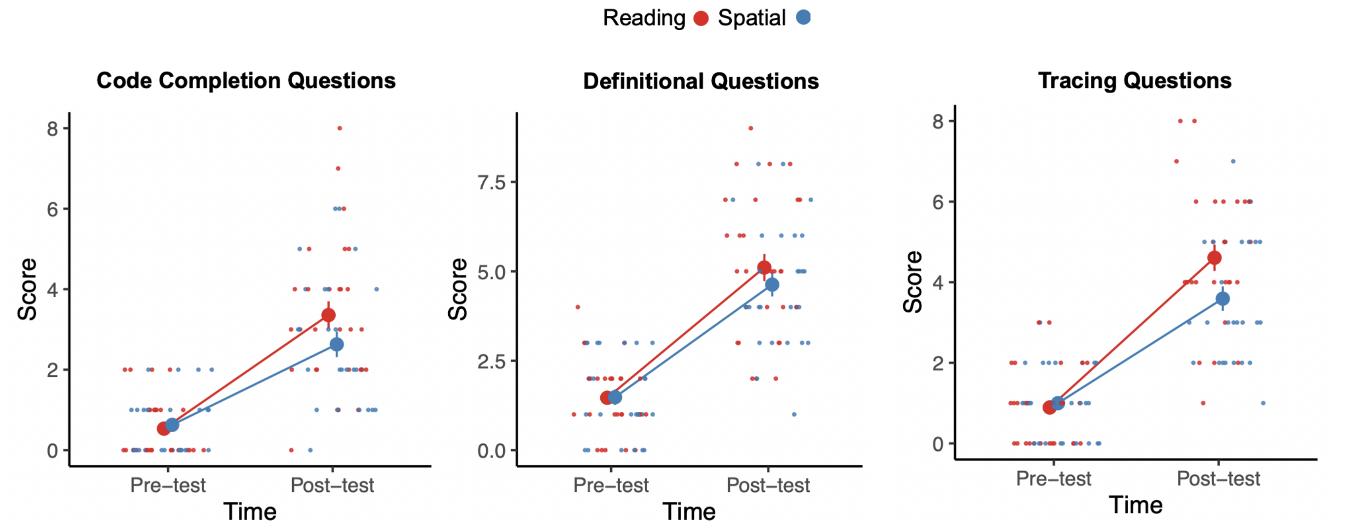
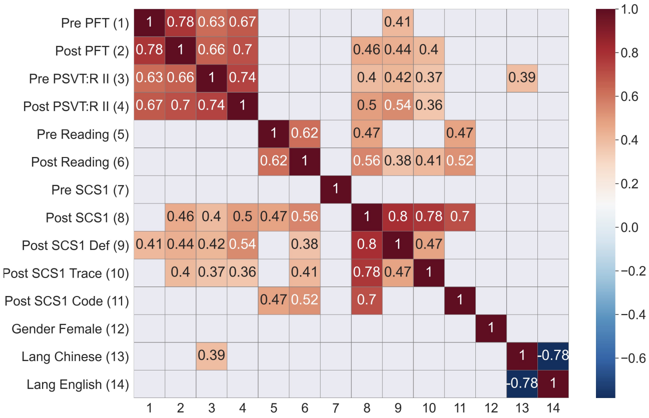


Figure 4: Question-type specific programming gains by Treatment: Reading Treatment participants exhibit significantly larger gains than Spatial Treatment participants on the Tracing questions but not on the Code Completion or Definitional questions. For each graph, score is out of a maximum of 9.



	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Pre PFT (1)	1	0.78	0.63	0.67					0.41					
Post PFT (2)	0.78	1	0.66	0.7				0.46	0.44	0.4				
Pre PSVT:R II (3)	0.63	0.66	1	0.74				0.4	0.42	0.37			0.39	
Post PSVT:R II (4)	0.67	0.7	0.74	1				0.5	0.54	0.36				
Pre Reading (5)					1	0.62		0.47			0.47			
Post Reading (6)					0.62	1		0.56	0.38	0.41	0.52			
Pre SCS1 (7)							1							
Post SCS1 (8)		0.46	0.4	0.5	0.47	0.56		1	0.8	0.78	0.7			
Post SCS1 Def (9)	0.41	0.44	0.42	0.54		0.38		0.8	1	0.47				
Post SCS1 Trace (10)		0.4	0.37	0.36		0.41		0.78	0.47	1				
Post SCS1 Code (11)					0.47	0.52		0.7			1			
Gender Female (12)												1		
Lang Chinese (13)			0.39										1	-0.78
Lang English (14)													-0.78	1

Figure 5: Correlation Matrix: contains significant Spearman's rank correlations for pre-tests, post-tests, and select demographic traits for all participants.

is required to confirm our observation. We did not observe any significant correlations between gender or native language with spatial ability, reading ability, nor programming scores.

> Spatial ability and reading ability, while distinct tasks, are both correlated with programming ability. We also observe a trend that reading ability may be more correlated with programming ability than spatial ability.

7.5 RQ5: Perception

Finally, we present a brief analysis of participants' experiences in each Treatment. After the skills-based post-test, participants filled out a questionnaire to capture self-reported assessments of training efficacy and programming competency (see Section 5). Participants also wrote short-answer explanations for their self-reported assessments. We focus on two questions: (1) "Did participating in the Transfer Training Study increase or decrease your desire to major or minor in computer science?" and (2) "Do you personally feel that the training sessions helped you with CS1? Why or why not?."

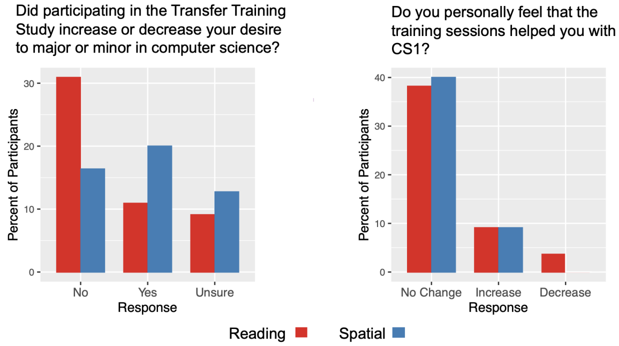


Figure 6: Participant Experiences: participant response counts regarding their subjective experiences in both Treatments

Perception Question 1 — Major Declaration: In both Treatments, a majority of participants reported that our study did not change their desire to major or minor in computer science (75.1% of Reading vs. 81.5% of Spatial). Similar proportions of each Treatment reported that participating actually increased their desire to continue (18.5% of Spatial vs. 17.9% of Reading). A Chi-square test compared the proportion of different responses to this question between groups, and there were no significant differences ($\chi^2(2) = 2.01, p = .37$).

Perception Question 2 — Perceived Helpfulness: We also compared the proportion of responses between Treatments for whether students thought the training helped with CS1. We find no statistically significant differences between groups ($\chi^2(2) = 4.25, p = .12$). Although this test did not reach significance, the Spatial Treatment appeared to be perceived to be more effective than the Reading Treatment (40.2% vs 17.9%). All but one Spatial Treatment participant who described the training as useful also described their programming activities as spatial tasks. As a particularly demonstrative example, one subject explained that they "really like to visualize how the computer processes the information. This often requires imagining data traveling in space from one place to another and learning more about spatial thinking, I thought, really helped me in doing so." Students who did not find the Spatial Treatment helpful generally reported that they saw no clear connection between the topics.

While less common, several Reading Treatment participants also found the training helpful for CS1. For instance, one participant thought that the Reading Treatment "helped me to become more logical" while another participant commented that "it gave me an insight to computer science that I have been looking for." For those students who found the Reading Treatment less helpful, a common theme was that they wanted more hands-on programming practice directly related to CS1 (≈ 70%). For instance, one student found that "it wasn't really helpful for [CS1] because we didn't practice coding concepts." Other students wanted more in-depth coverage of topics such as API documentation and less focus on vocabulary.

We find this difference between student perception of training helpfulness and observed quantitative helpfulness intriguing. We encourage future studies to investigate this effect further using a methodology, such as interviews, that enables the collection of more robust qualitative data.

> Although not statistically significant, students perceived the Spatial Treatment to be more helpful than the Reading Treatment, a discrepancy between student perception and quantitative effect. Treatments had no significant negative effect on students' intentions to continue CS.

8 DISCUSSION AND IMPLICATIONS

We now consider the implications of our results and directions for future research. We first discuss our primary, perhaps surprising, result that first-year computer science students may benefit more from our CS-focused technical Reading Treatment than a standardized Spatial Treatment. A diverse, and sometimes contradictory, array of models have been proposed as cognitive theories for learning programming. Some focus primarily on math or visuospatial processes [40], while others build on models explaining comprehension of natural language texts [62]. In their survey of proposed program comprehension models through the lens of CS-education, Schulte *et al.* posit that reading strategies may be a distinct and important element of the knowledge base required to learn programming [57]. Our finding supports this supposition, lending support to program comprehension models that emphasize reading strategies. Our work also motivates additional research on the relationship between technical reading strategies and programming.

We do not claim, however, that spatial ability is unhelpful for software engineering. In fact, we observe a positive correlation between spatial ability and programming scores. Rather, we claim that technical reading may in some cases be a *more* transferable skill, especially when a population already has high incoming spatial ability. We also note that while both spatial ability and reading ability correlate with programming, they are at most weakly correlated with each other. Therefore, spatial ability and technical reading may be distinct skills that are both critical for software engineering.

One potential direction, therefore, involves teasing out which features of software engineering are more spatial in nature and which are more aligned with technical reading. For instance, Huang *et al.*'s finding that tree-based data structures activate visiospatial brain areas while array-based data structures do not [32] may indicate that spatial ability becomes more important for software engineers later in their development; tree-based data structures are more often taught in CS2, not CS1. Subsequent neuroimaging studies found that code reading was less associated with spatial ability than was code writing [37]. Furthermore, there is evidence that more experienced programmers have stronger spatial ability [47].

In contrast, an alternate hypothesis is that technical reading may become *even more* important for experienced software engineers. For novices, the ability to trace through code and describe its function in natural language is highly predictive of programming ability [39, 44]. While useful for novices, code tracing also remains essential for experienced developers; software engineers are frequently required to read and understand other programmers' code to both contribute to large multi-programmer projects [24] and also for code reviews [3]. Interestingly, our Reading Treatment's effect was largest for questions that required code tracing, a result that underscores the connection between code tracing and natural language facility. Therefore, exploring technical reading-based training for more expert programmers is one future direction.

In this context, our results may appear less surprising given that the CS1 testing primarily focused on code comprehension and tracing (see Section 5) rather than code writing (cf. [37]), and that others have found links between CS performance and comprehension (e.g., [39, 41, 44]).

As discussed in Section 6, one reason our Spatial participants did not see significant spatial gains compared to those in the Reading Treatment may be that our participants started CS1 with relatively high spatial ability compared to other spatial ability training studies. Another future research direction, therefore, involves investigating these treatments on students with weaker incoming cognitive skills, both spatial- and reading-related. It is possible that low-spatial students will benefit more from the Spatial Treatment than the Reading Treatment; our participants may already have "enough" spatial ability for the programming tasks at hand. However, it is also possible that novice programmers with low spatial ability may use different, more Reading Treatment amenable, program comprehension strategies than their highly spatial-skilled counterparts. If, as posited by SpES, spatial ability helps students develop general strategies for mentally encoding programming problems [40], then it is possible that low-spatial programmers compensate by using alternate problem-solving strategies better supported by technical reading.

Better understanding programming strategies used by software engineers is critical for providing better programming support. Our finding that CS-focused technical reading training may improve programming skill has implications for future CS1 instruction and for training more experienced software engineers. We hope future replication will investigate more directly which aspects of the reading intervention are the most helpful, and we also hope to adapt the transfer training curriculum for experts instead of novices; it is an open question how technical reading training can be best adapted to aid more expert programmers. Ultimately, we hope our results motivate the rigorous investigation of the relationships between technical reading ability, spatial ability, and programming for novice and experienced software engineers alike.

9 LIMITATIONS AND THREATS TO VALIDITY

In this section, we address various limitations and threats to the validity. In particular, we address threats to our experimental control, generalizability concerns, threats concerning our training validation step, and limitations imposed by COVID-19. We also discuss our efforts to mitigate these threats.

One threat to validity is that despite random assignment of participants to Treatments, our populations were not homogeneous in incoming spatial ability. Reading Treatment participants' average incoming spatial ability was higher than that of Spatial Treatment participants (see Section 4). We mitigate this by using linear regression models that account for pre-existing individual differences.

Furthermore, as noted in Section 6, we did not observe significant differences between treatment groups for gains on the spatial and reading assessments. This raises the concern that our interventions could be ineffective at teaching spatial ability and technical reading. However, also as indicated in Section 6, the lack of difference in spatial ability between groups is likely due to a ceiling effect as our population already had high incoming spatial ability scores. As for the lack of difference of reading scores between treatments, we note the limitations in using a generic reading assessment, the GRE, as a proxy for CS-focused technical reading ability. We note that this threat does not lesson the core result that our Reading Treatment benefited students more than the Spatial Treatment. However, we acknowledge it is possible that our choice of test may affect our within-domain Reading Treatment assessment.

Our work also faces generalizability limitations. While our overall results were statistically significant, our population of 57 students is only a small portion of all students taking a single CS1 course at a single American university. As a result, our results may not generalize to other CS1 curricula. Also, all students in the study had time to attend multiple two-hour training sessions. It is possible our results will not generalize to CS1 students who would not elect to participate in such a study. Furthermore, our participants also started CS1 with comparatively high spatial ability. Therefore, our results may not generalize to low-spatial populations.

Additionally, our study was conducted during the COVID-19 pandemic. COVID-19 did not directly impact our praxis until the fifth week, when we transferred training sessions from in-person to virtual. We mitigated the effects of transitioning online by requiring all training material be completed while in a video chat with a proctor, sending participants physical training materials when possible, and having participants email proctors scans of their work. Even so, COVID-19 impacted the generalizability and power of our results. For instance, the online sessions contained fewer active learning activities, potentially lowering student engagement and learning [50]. Furthermore, participant dropout may have skewed our population. That is, students with the means and time to continue after the start of COVID-19 may be different than the initial study population. While any such difference is likely to affect both treatment groups equally, it may affect the generalizability of our findings to all CS1 students.

Finally, we note that while our results are intriguing, our effect sizes are small. We encourage future researchers to attempt to replicate our findings given these small effects and the limitations of our research

10 CONCLUSION

In this paper, we propose a CS-focused technical reading intervention for novice software engineers. We evaluate our intervention using a controlled 11-week longitudinal study with 57 participants where we compare CS1 programming outcomes between our Reading Treatment and a standardized Spatial Treatment. We find evidence that **Reading Treatment participants exhibit larger programming gains than Spatial Treatment participants** ($p = 0.02$), and that training type has a small size effect on final programming score ($f^2 = 0.10$). We also observe that our **Reading Treatment is most helpful for programming problems that require code tracing** ($p = 0.03$); the difference between the Spatial and Reading Treatment scores on definitional and code-completion problems is more modest. Finally, we find that while both spatial ability and reading ability correlate with CS1 programming ability, they are not strongly correlated with each other, indicating that both may be discrete cognitive skills important for software engineering success.

While not without limitations, our work remains a positive step toward combining insights from the cognitive study of software engineering with insights from computer science pedagogy. This interdisciplinary research has the potential to help create software tools that better support all programmers regardless of their incoming skills and strengths. It also may help improve software engineering curricula, including improving curricula for retraining an adults into more software-focused careers. We therefore hope our work spurs future research on the multifaceted connections between technical reading, spatial ability, and software engineering.

ACKNOWLEDGEMENTS

We acknowledge the partial support of the NSF (CCF 1908633, CCF 1763674) as well as both the Center for Research on Learning and Teaching and the Center for Academic Innovation at the University of Michigan. Additionally, we thank our research assistants Anne Fitzpatrick, Annie Li, Serena Chan, and Zachary Karas for their help leading training sessions and grading assessments. We further thank Amber Solomon, Mark Guzdial, Miranda Parker, and So Yoon Yoon for their helpful discussions and willingness to share validated assessments. Finally, we especially and sincerely thank Dr. Sheryl Sorby, without whom this work would not have been possible, for her generosity in sharing her spatial training curriculum materials.

REFERENCES

[1] Maizam Alias, Thomas R Black, and David E Gray. 2002. Effect of instruction on spatial visualization ability in civil engineering students. *International Education Journal* 3, 1 (2002), 1–12.

[2] Michael Patrick Allen. 1997. *The problem of multicollinearity.* Springer US, Boston, MA, 176–180. https://doi.org/10.1007/978-0-585-25657-3_37

[3] Alberto Bacchelli and Christian Bird. 2013. Expectations, outcomes, and challenges of modern code review. In *35th International Conference on Software Engineering, ICSE '13, San Francisco, CA, USA, May 18-26, 2013*, David Notkin, Betty H. C. Cheng, and Klaus Pohl (Eds.). IEEE Computer Society, 712–721. https://doi.org/10.1109/ICSE.2013.6606617

[4] Theresa Beaubouef and John Mason. 2005. Why the high attrition rate for computer science students: some thoughts and observations. *ACM SIGCSE Bull.* 37, 2 (2005), 103–106. https://doi.org/10.1145/1083431.1083474

[5] Alice A Black. 2005. Spatial ability and earth science conceptual understanding. *Journal of Geoscience Education* 53, 4 (2005), 402–414.

[6] Angela Blums, Jay Belsky, Kevin Grimm, and Zhe Chen. 2017. Building links between early socioeconomic status, cognitive ability, and math and science achievement. *Journal of Cognition and Development* 18, 1 (2017), 16–40.

[7] Ryan Bockmon, Stephen Cooper, William Koperski, Jonathan Gratch, Sheryl Sorby, and Mohsen Dorodchi. 2020. A CS1 Spatial Skills Intervention and the Impact on Introductory Programming Abilities. In *Proceedings of the 51st ACM Technical Symposium on Computer Science Education, SIGCSE 2020, Portland, OR, USA, March 11-14, 2020*, Jian Zhang, Mark Sherriff, Sarah Heckman, Pamela A. Cutter, and Alvaro E. Monge (Eds.). ACM, 766–772. https://doi.org/10.1145/3328778.3366829

[8] Sara E Brownell, Jordan V Price, and Lawrence Steinman. 2013. A writing-intensive course improves biology undergraduates' perception and confidence of their abilities to read scientific literature and communicate science. *Advances in Physiology Education* 37, 1 (2013), 70–79.

[9] Raymond P. L. Buse and Thomas Zimmermann. 2012. Information needs for software development analytics. In *34th International Conference on Software Engineering, ICSE 2012, June 2-9, 2012, Zurich, Switzerland*, Martin Glinz, Gail C. Murphy, and Mauro Pezzè (Eds.). IEEE Computer Society, 987–996. https://doi.org/10.1109/ICSE.2012.6227122

[10] Teresa Busjahn, Roman Bednarik, Andrew Begel, Martha E. Crosby, James H. Paterson, Carsten Schulte, Bonita Sharif, and Sascha Tamm. 2015. Eye movements in code reading: relaxing the linear order. In *Proceedings of the 2015 IEEE 23rd International Conference on Program Comprehension, ICPC 2015, Florence/Firenze, Italy, May 16-24, 2015*, Andrea De Lucia, Christian Bird, and Rocco Oliveto (Eds.). IEEE Computer Society, 255–265. https://doi.org/10.1109/ICPC.2015.36

[11] Teresa Busjahn, Carsten Schulte, and Andreas Busjahn. 2011. Analysis of code reading to gain more insight in program comprehension. In *11th Koli Calling International Conference on Computing Education Research, Koli Calling '11, Koli, Finland, November 17-20, 2011*, Ari Korhonen and Robert McCartney (Eds.). ACM, 1–9. https://doi.org/10.1145/2094131.2094133

[12] Beth M Casey, Eric Dearing, Marina Vasilyeva, Colleen M Ganley, and Michele Tine. 2011. Spatial and numerical predictors of measurement performance: The moderating effects of community income and gender. *Journal of educational psychology* 103, 2 (2011), 296. https://doi.org/10.1037/a0022516

[13] Souti Chattopadhyay, Nicholas Nelson, Yenifer Ramirez Gonzalez, Annel Amelia Leon, Rahul Pandita, and Anita Sarma. 2019. Latent patterns in activities: a field study of how developers manage context. In *Proceedings of the 41st International Conference on Software Engineering, ICSE 2019, Montreal, QC, Canada, May 25-31, 2019*, Joanne M. Atlee, Tevfik Bultan, and Jon Whittle (Eds.). IEEE / ACM, 373–383. https://doi.org/10.1109/ICSE.2019.00051

[14] Jacob Cohen. 1988. *Statistical power analysis for the behavioral sciences.* Academic press. https://doi.org/10.4324/9780203771587

[15] Stephen Cooper, Karen Wang, Maya Israni, and Sheryl Sorby. 2015. Spatial Skills Training in Introductory Computing. In *International Computing Education Research.* 13–20. https://doi.org/10.1145/2787622.2787728

[16] Phillip Dawson, Jacques van der Meer, Jane Skalicky, and Kym Cowley. 2014. On the effectiveness of supplemental instruction: A systematic review of supplemental instruction and peer-assisted study sessions literature between 2001 and 2010. *Review of educational research* 84, 4 (2014), 609–639.

[17] Jim Eison. 2010. Using active learning instructional strategies to create excitement and enhance learning. *Jurnal Pendidikantentang Strategi Pembelajaran Aktif (Active Learning) Books* 2, 1 (2010), 1–10.

[18] Madeline Endres, Zachary Karas, Xiaosu Hu, Ioulia Kovelman, and Westley Weimer. 2021. Relating Reading, Visualization, and Coding for New Programmers: A Neuroimaging Study. In *43rd IEEE/ACM International Conference on Software Engineering, ICSE 2021, Madrid, Spain, 22-30 May 2021.* IEEE, 600–612. https://doi.org/10.1109/ICSE43902.2021.00062

[19] ETS.org. 2020. *GRE Home.* ETS. Retrieved July 15, 2020 from https://www.ets.org/gre

[20] Evelina Fedorenko, Anna Ivanova, Riva Dhamala, and Marina Umaschi Bers. 2019. The language of programming: a cognitive perspective. *Trends in cognitive sciences* 23, 7 (2019), 525–528. https://doi.org/10.1016/j.tics.2019.04.010

[21] Sally Fincher, Anthony Robins, Bob Baker, Ilona Box, Quintin Cutts, Michael de Raadt, Patricia Haden, John Hamer, Margaret Hamilton, Raymond Lister, et al. 2006. Predictors of success in a first programming course. In *Proceedings of the 8th Australasian Computing Education Conference (ACE 2006)*, Vol. 52. Australian Computer Society Inc., 189–196.

[22] Benjamin Floyd, Tyler Santander, and Westley Weimer. 2017. Decoding the representation of code in the brain: an fMRI study of code review and expertise. In *Proceedings of the 39th International Conference on Software Engineering, ICSE 2017, Buenos Aires, Argentina, May 20-28, 2017*, Sebastián Uchitel, Alessandro Orso, and Martin P. Robillard (Eds.). IEEE / ACM, 175–186. https://doi.org/10.1109/ICSE.2017.24

[23] Andrew Forward and Timothy Lethbridge. 2002. The relevance of software documentation, tools and technologies: a survey. In *Proceedings of the 2002 ACM Symposium on Document Engineering, McLean, Virginia, USA, November 8-9, 2002.* ACM, 26–33. https://doi.org/10.1145/585058.585065

[24] Thomas Fritz, Jingwen Ou, Gail C. Murphy, and Emerson R. Murphy-Hill. 2010. A degree-of-knowledge model to capture source code familiarity. In *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering - Volume 1, ICSE 2010, Cape Town, South Africa, 1-8 May 2010*, Jeff Kramer, Judith Bishop, Premkumar T. Devanbu, and Sebastián Uchitel (Eds.). ACM, 385–394. https://doi.org/10.1145/1806799.1806856

[25] William Grabe. 2004. 3. Research on teaching reading. *Annual review of applied linguistics* 24 (2004), 44.

[26] Ananda Gunawardena, Aaron Tan, and David Kaufer. 2010. Encouraging reading and collaboration using classroom salon. In *Proceedings of the fifteenth annual conference on Innovation and technology in computer science education.* 254–258.

[27] Sonia Haiduc, Jairo Aponte, and Andrian Marcus. 2010. Supporting program comprehension with source code summarization. In *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering - Volume 2, ICSE 2010, Cape Town, South Africa, 1-8 May 2010*, Jeff Kramer, Judith Bishop, Premkumar T. Devanbu, and Sebastián Uchitel (Eds.). ACM, 223–226. https://doi.org/10.1145/1810295.1810335

[28] Christoph Hannebauer, Marc Hesenius, and Volker Gruhn. 2018. Does syntax highlighting help programming novices? *Empir. Softw. Eng.* 23, 5 (2018), 2795–2828. https://doi.org/10.1007/s10664-017-9579-0

[29] Regina Hebig, Truong Ho-Quang, Rodi Jolak, Jan Schröder, Humberto Linero, Magnus Ågren, and Salome Honest Maro. 2020. How do Students Experience and Judge Software Comprehension Techniques?. In *ICPC '20: 28th International Conference on Program Comprehension, Seoul, Republic of Korea, July 13-15, 2020.* ACM, 425–435. https://doi.org/10.1145/3387904.3389283

[30] Mary Hegarty and Maria Kozhevnikov. 1999. Types of visual–spatial representations and mathematical problem solving. *Journal of educational psychology* 91, 4 (1999), 684.

[31] Sherry Hsi, Marcia C Linn, and John E Bell. 1997. The role of spatial reasoning in engineering and the design of spatial instruction. *Journal of engineering education* 86, 2 (1997), 151–158.

[32] Yu Huang, Xinyu Liu, Ryan Krueger, Tyler Santander, Xiaosu Hu, Kevin Leach, and Westley Weimer. 2019. Distilling neural representations of data structure manipulation using fMRI and fNIRS. In *International Conference on Software Engineering.* 396–407. https://doi.org/10.1109/ICSE.2019.00053

[33] Sue Jane Jones and Gary E. Burnett. 2007. Spatial Ability and Learning to Program. In *Proceedings of the 19th Annual Workshop of the Psychology of Programming Interest Group, PPIG 2007, Joensuu, Finland, July 2-3, 2007.* Psychology of Programming Interest Group, 19. http://ppig.org/library/paper/spatial-ability-and-learning-program

[34] Philip R. Ventura Jr. 2005. Identifying predictors of success for an objects-first CS1. *Comput. Sci. Educ.* 15, 3 (2005), 223–243. https://doi.org/10.1080/08993400500224419

[35] Steven J Kass, Robert H Ahlers, and Melissa Dugger. 1998. Eliminating gender differences through practice in an applied visual spatial task. *Human Performance* 11, 4 (1998), 337–349.

[36] Päivi Kinnunen and Lauri Malmi. 2006. Why students drop out CS1 course?. In *International Computing Education Research Workshop 2006, ICER '06, Canterbury, UK, September 9-10, 2006*, Richard J. Anderson, Sally Fincher, and Mark Guzdial (Eds.). ACM, 97–108. https://doi.org/10.1145/1151588.1151604

[37] Ryan Krueger, Yu Huang, Xinyu Liu, Tyler Santander, Westley Weimer, and Kevin Leach. 2020. Neurological divide: an fMRI study of prose and code writing. In *ICSE '20: 42nd International Conference on Software Engineering, Seoul, South Korea, 27 June - 19 July, 2020*, Gregg Rothermel and Doo-Hwan Bae (Eds.). ACM, 678–690. https://doi.org/10.1145/3377811.3380348

[38] Robert E Larzelere, Brett R Kuhn, and Byron Johnson. 2004. The intervention selection bias: an underrecognized confound in intervention research. *Psychological bulletin* 130, 2 (2004), 289.

[39] Mike Lopez, Jacqueline L. Whalley, Phil Robbins, and Raymond Lister. 2008. Relationships between reading, tracing and writing skills in introductory programming. In *International Computing Education Research Workshop, ICER '08,*

Sydney, Australia, September 6-7, 2008, Michael E. Caspersen, Raymond Lister, and Mike Clancy (Eds.). ACM, 101–112. https://doi.org/10.1145/1404520.1404531

[40] Lauren E. Margulieux. 2019. Spatial Encoding Strategy Theory: The Relationship between Spatial Skill and STEM Achievement. In *Proceedings of the 2019 ACM Conference on International Computing Education Research* (Toronto ON, Canada) *(ICER '19)*. 81–90. https://doi.org/10.1145/3291279.3339414

[41] Richard E Mayer. 1981. The psychology of how novices learn computer programming. *ACM Computing Surveys (CSUR)* 13, 1 (1981), 121–141.

[42] Paul W McBurney and Collin McMillan. 2014. Automatic documentation generation via source code summarization of method context. In *Proceedings of the 22nd International Conference on Program Comprehension*. 279–290.

[43] André N. Meyer, Thomas Fritz, Gail C. Murphy, and Thomas Zimmermann. 2014. Software developers' perceptions of productivity. In *Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering, (FSE-22), Hong Kong, China, November 16 - 22, 2014*, Shing-Chi Cheung, Alessandro Orso, and Margaret-Anne D. Storey (Eds.). ACM, 19–29. https://doi.org/10.1145/2635868.2635892

[44] Laurie Murphy, Sue Fitzgerald, Raymond Lister, and Renée McCauley. 2012. Ability to'explain in plain english'linked to proficiency in computer-based programming. In *Proceedings of the ninth annual international conference on International computing education research*. 111–118.

[45] Miranda Parker, Amber Solomon, Brianna Pritchett, David Illingworth, Lauren Margulieux, and Mark Guzdial. 2018. Socioeconomic Status and Computer Science Achievement: Spatial Ability as a Mediating Variable in a Novel Model of Understanding. In *Proceedings of the 2018 ACM Conference on International Computing Education Research*. 97–105. https://doi.org/10.1145/3230977.3230987

[46] Miranda C. Parker, Mark Guzdial, and Shelly Engleman. 2016. Replication, Validation, and Use of a Language Independent CS1 Knowledge Assessment. In *Proceedings of the 2016 ACM Conference on International Computing Education Research* (Melbourne, VIC, Australia) *(ICER '16)*. 93–101. https://doi.org/10.1145/2960310.2960316

[47] Jack Parkinson and Quintin I. Cutts. 2018. Investigating the Relationship Between Spatial Skills and Computer Science. In *Proceedings of the 2018 ACM Conference on International Computing Education Research, ICER 2018, Espoo, Finland, August 13-15, 2018*, Lauri Malmi, Ari Korhonen, Robert McCartney, and Andrew Petersen (Eds.). ACM, 106–114. https://doi.org/10.1145/3230977.3230990

[48] Chantel S Prat, Tara M Madhyastha, Malayka J Mottarella, and Chu-Hsuan Kuo. 2020. Relating natural language aptitude to individual differences in learning programming languages. *Scientific reports* 10, 1 (2020), 1–10. https://doi.org/10.1038/s41598-020-60661-8

[49] Jeffrey R Pribyl and George M Bodner. 1987. Spatial ability and its role in organic chemistry: A study of four organic courses. *Journal of research in science teaching* 24, 3 (1987), 229–240.

[50] Michael Prince. 2004. Does active learning work? A review of the research. *Journal of engineering education* 93, 3 (2004), 223–231.

[51] Václav Rajlich. 2013. Teaching developer skills in the first software engineering course. In *35th International Conference on Software Engineering, ICSE '13, San Francisco, CA, USA, May 18-26, 2013*, David Notkin, Betty H. C. Cheng, and Klaus Pohl (Eds.). IEEE Computer Society, 1109–1116. https://doi.org/10.1109/ICSE.2013.6606661

[52] Alison D Rayome. 2018. *5 eye-opening statistics about minorities in tech.* TechRepublic. Retrieved July 15, 2020 from https://www.techrepublic.com/article/5-eye-opening-statistics-about-minorities-in-tech/

[53] Paige Rodeghero and Collin McMillan. 2015. An Empirical Study on the Patterns of Eye Movement during Summarization Tasks. In *2015 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement, ESEM 2015, Beijing, China, October 22-23, 2015*. IEEE Computer Society, 11–20. https://doi.org/10.1109/ESEM.2015.7321188

[54] Paige Rodeghero, Collin McMillan, Paul W. McBurney, Nigel Bosch, and Sidney K. D'Mello. 2014. Improving automated source code summarization via an eye-tracking study of programmers. In *36th International Conference on Software Engineering, ICSE '14, Hyderabad, India - May 31 - June 07, 2014*, Pankaj Jalote, Lionel C. Briand, and André van der Hoek (Eds.). ACM, 390–401. https://doi.org/10.1145/2568225.2568247

[55] Debopriyo Roy. 2008. Using concept maps for information conceptualization and schematization in technical reading and writing courses: A case study for computer science majors in Japan. In *2008 IEEE International Professional Communication Conference*. IEEE, 1–12.

[56] Saddleback Publishing. 2002. *Reading Comprehension Skills and Strategies Level 8*. Saddleback Publishing. https://books.google.com/books?id=cMJ4ypi2_HIC

[57] Carsten Schulte, Tony Clear, Ahmad Taherkhani, Teresa Busjahn, and James H Paterson. 2010. An introduction to program comprehension for computer science educators. In *Proceedings of the 2010 ITiCSE working group reports*. 65–86.

[58] Valerie J Shute. 1991. Who is likely to acquire programming skills? *Journal of educational Computing research* 7, 1 (1991), 1–24. https://doi.org/10.2190/VQJD-T1YD-5WVB-RYPJ

[59] Janet Siegmund, Christian Kästner, Sven Apel, Chris Parnin, Anja Bethmann, Thomas Leich, Gunter Saake, and André Brechmann. 2014. Understanding understanding source code with functional magnetic resonance imaging. In *36th International Conference on Software Engineering, ICSE '14, Hyderabad, India - May 31 - June 07, 2014*, Pankaj Jalote, Lionel C. Briand, and André van der Hoek (Eds.). ACM, 378–389. https://doi.org/10.1145/2568225.2568252

[60] Janet Siegmund, Norman Peitek, Chris Parnin, Sven Apel, Johannes C. Hofmeister, Christian Kästner, Andrew Begel, Anja Bethmann, and André Brechmann. 2017. Measuring neural efficiency of program comprehension. In *Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering, ESEC/FSE 2017, Paderborn, Germany, September 4-8, 2017*, Eric Bodden, Wilhelm Schäfer, Arie van Deursen, and Andrea Zisman (Eds.). ACM, 140–150. https://doi.org/10.1145/3106237.3106268

[61] Edward K Smith, Christian Bird, and Thomas Zimmermann. 2016. Beliefs, practices, and personalities of software engineers: a survey in a large software company. In *Proceedings of the 9th International Workshop on Cooperative and Human Aspects of Software Engineering*. 15–18.

[62] Elliot Soloway and Kate Ehrlich. 1984. Empirical Studies of Programming Knowledge. *IEEE Trans. Software Eng.* 10, 5 (1984), 595–609. https://doi.org/10.1109/TSE.1984.5010283

[63] Sheryl Sorby, Norma Veurink, and Scott Streiner. 2018. Does spatial skills instruction improve STEM outcomes? The answer is 'yes'. *Learning and Individual Differences* 67 (2018), 209–222. https://doi.org/10.1016/j.lindif.2018.09.001

[64] Sheryl A Sorby and Beverly J Baartmans. 2000. The development and assessment of a course for enhancing the 3-D spatial visualization skills of first year engineering students. *Journal of Engineering Education* 89, 3 (2000), 301–307.

[65] Sheryl A. Sorby, Edmund Nevin, Avril Behan, Eileen Mageean, and Sarah Sheridan. 2014. Spatial skills as predictors of success in first-year engineering. In *IEEE Frontiers in Education Conference*. 1–7. https://doi.org/10.1109/FIE.2014.7044005

[66] Allison Elliott Tew and Mark Guzdial. 2011. The FCS1: A Language Independent Assessment of CS1 Knowledge. In *Proceedings of the 42nd ACM Technical Symposium on Computer Science Education* (Dallas, TX, USA) *(SIGCSE '11)*. 111–116. https://doi.org/10.1145/1953163.1953200

[67] Nikolai Tillmann, Jonathan de Halleux, Tao Xie, Sumit Gulwani, and Judith Bishop. 2013. Teaching and learning programming and software engineering via interactive gaming. In *35th International Conference on Software Engineering, ICSE '13, San Francisco, CA, USA, May 18-26, 2013*, David Notkin, Betty H. C. Cheng, and Klaus Pohl (Eds.). IEEE Computer Society, 1117–1126. https://doi.org/10.1109/ICSE.2013.6606662

[68] Torbjørn Torsheim, Franco Cavallo, Kate Ann Levin, Christina Schnohr, Joanna Mazur, Birgit Niclasen, Candace Currie, FAS Development Study Group, et al. 2016. Psychometric validation of the revised family affluence scale: a latent variable approach. *Child Indicators Research* 9, 3 (2016), 771–784.

[69] Dyanne M Tracy. 1987. Toys, spatial ability, and science and mathematics achievement: Are they related? *Sex roles* 17, 3-4 (1987), 115–138. https://doi.org/10.1007/BF00287620

[70] David H Uttal, Nathaniel G Meadow, Elizabeth Tipton, Linda L Hand, Alison R Alden, Christopher Warren, and Nora S Newcombe. 2013. The malleability of spatial skills: A meta-analysis of training studies. *Psychological bulletin* 139, 2 (2013), 352. https://doi.org/10.1037/a0028446

[71] Piia Maria Vilenius-Tuohimaa, Kaisa Aunola, and Jari-Erik Nurmi. 2008. The association between mathematical word problems and reading comprehension. *Educational Psychology* 28, 4 (2008), 409–426.

[72] Daniel Voyer, Susan Voyer, and M Philip Bryden. 1995. Magnitude of sex differences in spatial abilities: a meta-analysis and consideration of critical variables. *Psychological bulletin* 117, 2 (1995), 250. https://doi.org/10.1037/0033-2909.117.2.250

[73] Jonathan Wai, David Lubinski, and Camilla P Benbow. 2009. Spatial ability for STEM domains: Aligning over 50 years of cumulative psychological knowledge solidifies its importance. *Journal of Educational Psychology* 101, 4 (2009), 817. https://doi.org/10.1037/a0016127

[74] Yingxu Wang and Vincent Chiew. 2010. On the cognitive process of human problem solving. *Cogn. Syst. Res.* 11, 1 (2010), 81–92. https://doi.org/10.1016/j.cogsys.2008.08.003

[75] John Winslow, Ruth B Ekstrom, and Leighton A Price. 1963. *Kit of reference tests for cognitive factors.* Educational Testing Service.

[76] Amanda A Wolkowitz and Jeffrey A Kelley. 2010. Academic predictors of success in a nursing program. *Journal of Nursing Education* 49, 9 (2010), 498–503. https://doi.org/10.3928/01484834-20100524-09

[77] Xin Xia, Lingfeng Bao, David Lo, Zhenchang Xing, Ahmed E. Hassan, and Shanping Li. 2018. Measuring Program Comprehension: A Large-Scale Field Study with Professionals. *IEEE Trans. Software Eng.* 44, 10 (2018), 951–976. https://doi.org/10.1109/TSE.2017.2734091

[78] Eun-Mi Yang, Thomas Andre, Thomas J Greenbowe, and Lena Tibell. 2003. Spatial ability and the impact of visualization/animation on learning electrochemistry. *International Journal of Science Education* 25, 3 (2003), 329–349. https://doi.org/10.1080/09500690210126784

[79] So Y. Yoon. 2011. *Psychometric properties of the Revised Purdue Spatial Visualization Tests: Visualization of Rotations (the Revised PSVT:R).* Ph.D. Dissertation. Purdue University.