TU Delft

Delft University of Technology

Document Version
Final published version

Licence
CC BY

Citation (APA)
Noviello, Y., Sibia, N., Birillo, A., Overklift Vaupel Klein, T., Liut, M., & Migut, G. (2026). AI-Generated Traces for Novice Programmers: Learning Effects and Learner Differences in a Multi-Institutional Study. In *ITiCSE 2026 - Proceedings of the 31st ACM Conference on Innovation and Technology in Computer Science Education V. 1* (pp. 135-141). Association for Computing Machinery (ACM). https://doi.org/10.1145/3803400.3809346

Important note
To cite this publication, please use the final published version (if applicable).
Please check the document version above.

Copyright
In case the licence states “Dutch Copyright Act (Article 25fa)”, this publication was made available Green Open Access via the TU Delft Institutional Repository pursuant to Dutch Copyright Act (Article 25fa, the Taverne amendment). This provision does not affect copyright ownership.
Unless copyright is transferred by contract or statute, it remains with the copyright holder.

Sharing and reuse
Other than for strictly personal use, it is not permitted to download, forward or distribute the text or part of it, without the consent of the author(s) and/or copyright holder(s), unless the work is under an open content license such as Creative Commons.

Takedown policy
Please contact us and provide details if you believe this document breaches copyrights.
We will remove access to the work immediately and investigate your claim.

This work is downloaded from Delft University of Technology.

AI-Generated Traces for Novice Programmers: Learning Effects and Learner Differences in a Multi-Institutional Study

Yuri Noviello
Delft University of Technology
Delft, Netherlands
y.noviello@tudelft.nl

Naaz Sibia
University of Toronto
Toronto, Canada
naaz.sibia@utoronto.ca

Anastasiia Birillo
JetBrains Research
Belgrade, Serbia
anastasia.birillo@jetbrains.com

Thomas Overklift Vaupel Klein
Delft University of Technology
Delft, Netherlands
t.a.r.overkliftvaupelklein@tudelft.nl

Michael Liut
University of Toronto Mississauga
Mississauga, Canada
michael.liut@utoronto.ca

Gosia Migut
Delft University of Technology
Delft, Netherlands
m.a.migut@tudelft.nl

Abstract

Introductory programming (CS1) courses often struggle to support students' understanding of program execution. While visualizations can make execution processes explicit, their effectiveness depends on design and context, and empirical evidence for AI-generated visualizations remains limited. We propose Generated Animated Traces (*GATs*), AI-generated, analogy-based, narrated animations that coordinate source code, execution state, and conceptual analogies. We conduct a study at two institutions in CS1 courses (Python, $N = 961$; Java $N = 151$) comparing *GATs* to textual explanations. We measure immediate learning performance and experience, end-of-course engagement and exam performance. Results show that *GATs* can yield selective benefits for immediate learning, but benefits are context-dependent and short-term. We observe that *GATs*' influence on performance is moderated by learner engagement profiles. This finding underscores the importance of personalized approaches.

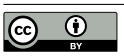
CCS Concepts

• **Social and professional topics** → **Computing education**; • **Applied computing** → **Computer-assisted instruction**.

Keywords

Visualization; LLM-generated Videos; Personalization; Abstraction

ACM Reference Format:
Yuri Noviello, Naaz Sibia, Anastasiia Birillo, Thomas Overklift Vaupel Klein, Michael Liut, and Gosia Migut. 2026. AI-Generated Traces for Novice Programmers: Learning Effects and Learner Differences in a Multi-Institutional Study. In *Proceedings of the 31st ACM Conference on Innovation and Technology in Computer Science Education V. 1 (ITiCSE 2026), July 10–15, 2026, Madrid, Spain.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3803400.3809346



This work is licensed under a Creative Commons Attribution 4.0 International License.
ITiCSE 2026, Madrid, Spain
© 2026 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-2634-7/2026/07
https://doi.org/10.1145/3803400.3809346

1 Introduction

Understanding how programs execute is a central challenge in introductory programming (CS1) courses. Novice programmers may learn to produce correct outputs while still lacking robust mental models of how variables, memory, and control flow interact at runtime, which limits transfer and long-term learning [39].

A common instructional response is to make program execution more explicit through visual explanations, such as traces, diagrams [10, 42], animations that present multiple views of execution [40], or domain-related analogies [38]. Prior work suggests that coordinating verbal and non-verbal channels can support comprehension [34], while dynamic visualizations with attention guidance help learners focus on task-relevant elements [11]. Advances in Large Language Models (LLMs) now make it possible to automatically generate such instructional materials at scale, creating new opportunities to deploy rich visual explanations consistently across large-enrollment courses [18, 32]. However, the educational impact of such materials remains unclear. Empirical evidence is limited on when, for whom, and under what conditions AI-generated instructional materials support learning, particularly across institutional contexts and learner profiles [18]. Prior work further suggests that instructional interventions rarely benefit all learners equally, and that average effects may obscure meaningful differences in engagement and learning outcomes [17].

In this work, we investigate Generated Animated Traces (*GATs*), an AI-generated, analogy-based, narrated animation that synchronizes a code snippet, a memory diagram, and a conceptual analogy within a single presentation to explicitly represent how program state and control flow evolve during execution. Rather than assuming benefits from coordinated views, we empirically evaluate how GATs compare to carefully matched step-by-step textual explanations across two CS1 courses with different languages and course structures. We address three research questions:

- **RQ1:** How do *GATs* affect (a) immediate post-intervention programming task performance and (b) course summative performance, compared to textual explanations?
- **RQ2:** How does students' learning experience differ between *GATs* and textual explanations, as measured by self-reported frustration, cognitive load, and situational interest?
- **RQ3:** To what extent are the effects of *GATs* moderated by learner engagement profiles?

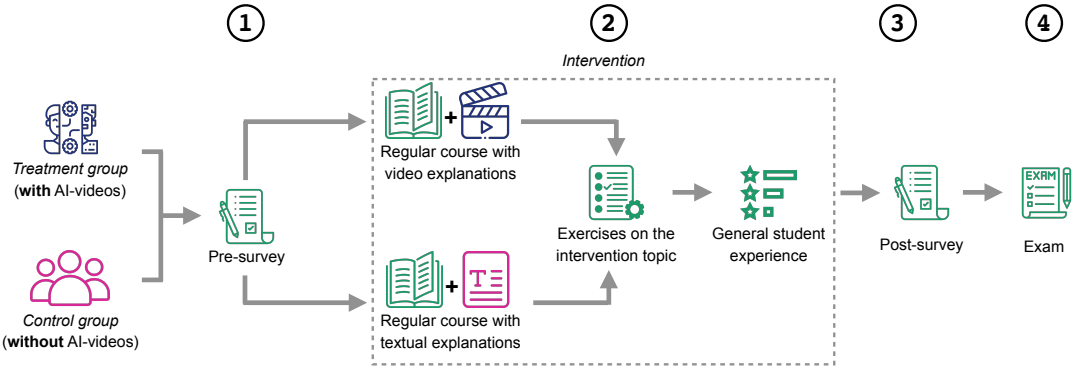


Figure 1: Study pipeline. Unique materials for the *treatment group* are marked in blue, and unique materials for the *control group* are marked in pink. Shared materials for both groups are marked in green.

In this work we provide the first systematic evaluation of GATs for CS1 across: (1) two institutions with different languages and structures, (2) multiple instruments (immediate/summative performance, learning experience, engagement), and (3) learner heterogeneity via engagement profiles. Our findings show that the impact of *GATs* is context- and learner-dependent, underscoring the need to move beyond average effects in CS1 instruction.

2 Related Work

Learning to program requires more than producing correct outputs. It requires building mental models that link code to runtime behavior, including variable state, control flow, and memory operations [22, 40]. In CS1, instruction is commonly scaffolded through notional machines, idealized models of how programming languages execute, that help learners reason about scope, state changes, and execution order [13]. Effective learning, therefore, depends on supporting abstraction rather than surface-level code tracing alone.

Program visualizations are widely used instructional tools for making program execution explicit. Tools such as Python Tutor expose step-by-step execution traces and runtime state [16], which can support tracing accuracy. However, highly detailed visualizations may anchor novices to low-level mechanics, limiting abstraction and transfer [12, 40]. Learners may successfully follow traces without developing deeper conceptual understanding.

However, learners do not automatically integrate multiple representations when exposed to program visualizations. In an eye-tracking study, novices often attend selectively to a single view unless explicitly guided [2]. Synthesizing this literature, Sorva concludes that visualization effectiveness depends not on visual richness alone, but on how representations are coordinated [40]. Similarly, to avoid redundancy and cognitive overload, the DeFT framework emphasizes that multiple external representations require careful design [1] and should guide attention to task-relevant elements [30, 41]. Also, the expertise-reversal effects suggest that instructional supports that are helpful for novices may hinder more advanced learners [24]. These findings underscore that adding instructional detail does not inherently improve learning outcomes.

In addition to program visualizations, analogies are frequently used in CS1 to ground abstract execution concepts in familiar domains. Well-designed analogies have been shown to reduce cognitive load and improve conceptual understanding [7, 14]. However, their effectiveness is highly design-sensitive, and misaligned analogies can mislead learners or impede abstraction [5]. At the same time, the analogy-based videos may increase engagement even when performance gains are limited [45], highlighting a distinction between learning experience and measured outcomes.

As an extension of visualization- and analogy-based instruction, recent studies indicate that AI-generated teaching materials, when pedagogically sound, can be as effective as human-produced materials [31, 44], although they tend to score lower on social presence [18]. Also, AI can assist learners in generating personalized analogies [4, 19]. As AI lowers the cost of producing rich instructional media for various needs, there is growing interest in deploying such materials at scale [18]. However, empirical evidence remains limited on when, for whom, and under what conditions AI-generated instructional materials support learning, particularly across institutional contexts and learner profiles. Addressing this gap requires evaluations that move beyond average effects to consider learner heterogeneity and multiple learning outcomes.

3 Methods

3.1 Study design

Overview. In 2025, we conducted two similar studies at *Delft University of Technology* (TU Delft) and the *University of Toronto Mississauga* (UofT), both research-intensive publicly funded institutions. Students at both sites were enrolled in an undergraduate introductory programming course (CS1), using Java at TU Delft and Python at UofT, and were randomly assigned to either *treatment* (AI-generated videos) or *control* (textual explanations) groups (Figure 1).

Instruments. At the start of the course, students completed a pre-survey on *baseline programming experience* (Figure 1 **(1)**). Both groups participated in interventions on selected topics (Figure 1 **(2)**). Immediately following each intervention students completed topic-aligned exercises, used to measure *short-term learning performance* (Figure 1 **(2)**). The exercises included programming task (contributing 80% to the immediate performance) and 2 MCQs (multiple-choice questions, each contributing 10% to the immediate performance). One MCQ was about code tracing (identify output of a given program), another one was on conceptual understanding (describe execution of a given program). Students also answered

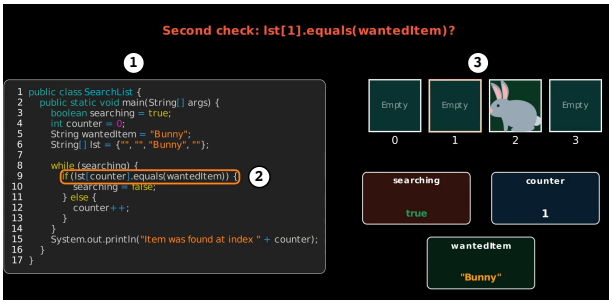


Figure 2: Example of a GAT provided to the treatment group: (1) a code snippet; (2) a highlighted code line indicating the program state to visualize; (3) a conceptual analogy to explain the highlighted code line.

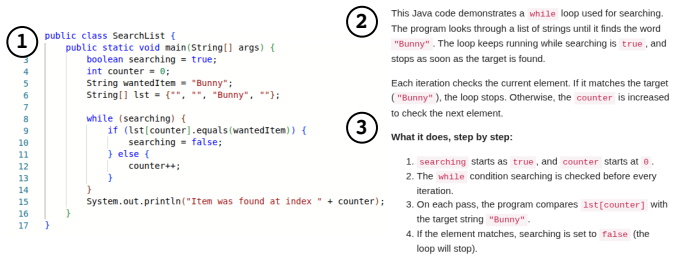


Figure 3: Example of a textual explanation provided to the control group: (1) the code snippet; (2) a general description of the code; (3) step-by-step execution order. The figure shows only the first few steps due to space limitations.

Likert-scale questions about *self-reported cognitive load, frustration, and situational interest*, adapted from established self-report instruments (Paas mental effort, NASA-TLX, and IMI items) [20, 33, 37] (Figure 1 **(2)**). At the end of the course students completed a post-survey assessing *self-reported engagement and elaboration strategies*, using CAP (Passive, Active, Constructive) engagement instruments [8, 9] and MSLQ elaboration [36] (Figure 1 **(3)**). *Long-term summative performance* was assessed via the final exam, using the same questions across institutions, adapted only for programming language (Figure 1 **(4)**).

Reproducibility. All study materials (surveys, videos, explanations and exercises) are included in the replication package [1].

Data Privacy & Ethics. Ethical approval was obtained from both institutions. To ensure instructional equivalence, the control group received textual explanations identical in content to those used in the treatment videos. Students were informed about the study purpose and data usage practices. To protect privacy, all datasets were anonymized prior to analysis.

3.1.1 Study 1 at TU Delft. The study took place in a 9-week CS1 course in the first quarter, taken by incoming CS majors and bridge-to-master's students. Taught in Java, it covered core topics (e.g., control flow, methods, OOP, file I/O, streams, unit testing) via lectures and live coding. Students completed 8 optional weekly labs, practice exams (with up to 5% bonus), and were assessed through a theoretical exam (50%) and a programming exam (50%). Interventions were embedded in labs, with voluntary participation incentivized by a raffle of ten €25 gift cards.

3.1.2 Study 2 at UofT. The study was conducted in a first-semester CS1 course for first-year students entering a competitive CS major, with an interdisciplinary cohort. The 12-week, Python-based course used a flipped classroom [6] and covered core topics (e.g., control flow, functions, OOP, file I/O, sorting, complexity, regular expressions). Assessment included a term test (20%), final exam (35%), eleven weekly labs (3% each), and LMS-based preparatory exercises (12%). Interventions were mandatory within exercises, with a 2% bonus for completing pre- and post-surveys.

3.1.3 Intervention topics. We selected five (5) topics for the interventions: *while-loop*, *arraylist*, *hashmap*, *file reading*, and *file writing*. These topics were selected because they are foundational CS1 concepts taught in both courses, aligning with CS2023 guidelines, but are also known to have common misconceptions among students [15]. For each topic, we had a matching test/exam question. In study 1 at TU Delft, we omitted *file writing* in the final exam due to logistical issues. In study 2 at UofT we deployed a selection of three (3) interventions: *while-loop*, *file reading*, and *file writing*. Unfortunately, *arraylist* and *hashmap* had to be removed for two reasons. First, the UofT deployment occurred in a larger, more heterogeneous population with mandatory participation, raising concerns about participant burden and engagement quality with extensive intervention materials. Second, practical constraints in course scheduling and assessment alignment made it challenging to meaningfully integrate all five topics within the existing curriculum structure. We selected topics that: (1) represented core CS1 concepts, (2) aligned with available exam items, and (3) were most likely to reveal effects given the population characteristics.

3.2 Interventions

For each topic an intervention consisted of an instructional artifact: video explanations (*GATs*) for the treatment group and *textual explanations* for the control group. The underlying code snippets and instructional logic were consistent across both groups, thus the intervention manipulated only the *presentation modality*.

3.2.1 Generated Animated Traces (GATs). A GAT is an AI-generated, analogy-based, narrated animation designed to support representational fluency by synchronizing two distinct views:

Source Code View (Figure 2 **(1)**) with the currently executed line(s) highlighted (Figure 2 **(2)**) to guide student attention;

Conceptual Analogy View that uses a visual analogy to foster abstraction (e.g., searching for a *bunny* in physical boxes to illustrate a *while* loop) (Figure 2 **(3)**). The analogy view is augmented with variable value updates, data structure contents, and control-flow transitions directly within the analogical context.

GATs were produced through a human in the loop pipeline. Initial scripts and execution traces were generated by an AI-driven system, that produces analogy-oriented explanations and automatically renders them as manim-based animations [32]. Subtitles for all movements were generated automatically, and a voiceover was added to provide audio narration. Final videos were lightly edited

[1] Replication package: https://osf.io/kb9uv

by the first author to refine timing, correct layout issues, and ensure pedagogical clarity and alignment with course instruction.

3.2.2 Textual explanations. Textual explanations consisted of a structured walkthrough of the *same code* used in the corresponding GAT (Figure 3). Each explanation included the code snippet (Figure 3 **(1)**), a high-level description of its purpose (Figure 3 **(2)**), and a step-by-step execution narrative describing actions and resulting state changes (Figure 3 **(3)**). To ensure comparability, textual explanations were matched exactly to GATs in learning objectives, conceptual scope, and line-by-line execution order.

3.3 Data Analysis

Our analysis plan was designed to estimate the effects of explanation modality on students' immediate and longer-term learning performance and experience. We analyzed TU Delft and UofT data separately. We did not pool data across institutions because the deployments differ in programming language and topic coverage.

Design Integrity and Attrition Checks. We used independent t-tests to check baseline equivalence in prior programming experience and Mann–Whitney U tests to assess differential attrition and modality-related dropout bias [28].

Modeling Immediate Impact and Experience. Immediate performance outcomes were analyzed using *Linear Mixed Models* [35] with student-level random intercepts to account for repeated interventions per student. Models included *Topic* and prior experience as covariates. We estimated pooled and topic-specific effects and applied this approach to immediate performance and experience instruments (cognitive load, frustration, and situational interest).

Long-Term Performance and Psychometric Modeling. Exam outcomes were analyzed at the student level using *Analysis of Covariance (ANCOVA)* [21], controlling for prior experience. Post-survey instruments (MSLQ elaboration and CAP engagement dimensions) were analyzed using analogous models to assess how GATs influenced students' end-of-course cognitive strategies.

Engagement Profiles Clustering. To address learner heterogeneity, we conducted an exploratory engagement profile analysis using *k-means* clustering on self-reported CAP instruments, in line with prior engagement profile analyses Wong et al. [43]. This allowed us to test if the efficacy of GATs was moderated by a student's specific *engagement profile*, moving beyond simple average effects.

Error Control and Significance Reporting. Due to the large number of dependent variables, we controlled the *False Discovery Rate (FDR)* using the *Benjamini-Hochberg (BH)* procedure [3] within predefined instruments domains. We report effect estimates, standard errors, and both raw and adjusted p-values.

4 Results

To validate the experimental design, we performed sanity checks on randomization and retention. For both institutions, no significant differences were found between the treatment and control groups for baseline equivalence (using self-reported prior programming experience) and differential attrition (via the number of completed interventions). The results of the analysis are presented in (Table 1).

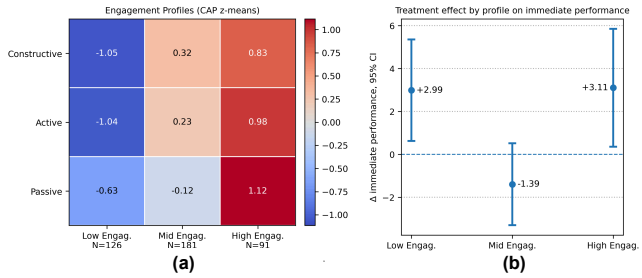


Figure 4: Engagement profiles and moderated treatment effects. (a) Heatmap shows mean standardized CAP engagement scores per cluster. (b) Within-profile treatment effects on immediate performance with 95% confidence intervals.

4.1 Immediate learning performance

Study 1 at TU Delft: *GATs* significantly improved immediate performance across topics (Table 1, row 21), mainly driven by the *While-loop* intervention (Table 1, row 1), where the treatment group outperformed the control group. Other topics showed positive but non-significant differences that failed correction.

Study 2 at UofT: No significant overall differences in immediate performance were found between conditions. Topic-specific results also showed no effects after correction, though *File Writing* had the largest raw difference favoring *GATs* (Table 1, row 5).

4.2 Immediate learning experience

Study 1 at TU Delft: No reliable differences were found for frustration or cognitive load, across topics. Situational interest showed a weak positive trend favoring *GATs*, but it did not survive correction. Topic-specific analyses found no significant effects, with the largest difference in situational interest on *File Reading*.

Study 2 at UofT: *GATs* reduced frustration in raw tests for pooled experience measures, but this did not survive FDR correction (Table 1, row 22). No reliable differences were found for cognitive load or situational interest (Table 1, rows 23–24). At the topic level, raw tests showed lower frustration for *While-loop* (Table 1, row 6) and reduced situational interest for *File Writing* (Table 1, row 20), but neither remained significant after correction.

4.3 Long-term exam outcomes

Across both **TU Delft** and **UofT**, ANCOVA models controlling for prior experience showed no significant condition differences on summative exam outcomes, with all adjusted p-values exceeding conventional significance thresholds (Table 1, rows 25-29).

4.4 Post-survey scales

Study 1 at TU Delft: The post-survey analysis yielded no significant effects (Table 1, rows 30-33). Raw analyses suggested higher passive engagement (Table 1, row 32) under *GATs*, but this pattern was not robust after FDR correction.

Study 2 at UofT: *GATs* produced a reliable increase in *Constructive engagement* (Table 1, row 30). No other engagement instruments showed effects surviving correction, and *MSLQ elaboration* did not differ between conditions (Table 1, row 31-33).

	Instrument	Topic / Context	N	Δ	p	p_{adj}	N	Δ	p	p_{adj}
Topic-specific effects				**TU Delft**				**UofT**		
1	Immediate performance	*While-loop*	138	8.227	**0.003**	**0.015**	927	-0.118	0.880	0.879
2	Immediate performance	*ArrayList*	99	1.347	0.651	0.651	—	—	—	—
3	Immediate performance	*HashMap*	52	5.771	0.090	0.218	—	—	—	—
4	Immediate performance	*File Reading*	34	4.270	0.296	0.370	764	-0.762	0.378	0.566
5	Immediate performance	*File Writing*	20	7.933	0.131	0.218	416	2.691	**0.021**	0.062
6	Frustration	*While-loop*	138	0.073	0.769	0.993	927	-0.163	**0.043**	0.129
7	Frustration	*ArrayList*	99	0.006	0.982	0.993	—	—	—	—
8	Frustration	*HashMap*	52	0.003	0.992	0.993	—	—	—	—
9	Frustration	*File Reading*	34	-0.542	0.139	0.484	764	-0.086	0.331	0.331
10	Frustration	*File Writing*	20	-0.610	0.194	0.484	416	-0.158	0.164	0.246
11	Cognitive load	*While-loop*	138	0.087	0.632	0.676	927	-0.121	0.073	0.220
12	Cognitive load	*ArrayList*	99	0.150	0.439	0.676	—	—	—	—
13	Cognitive load	*HashMap*	52	-0.090	0.676	0.676	—	—	—	—
14	Cognitive load	*File Reading*	34	-0.299	0.241	0.657	764	-0.040	0.582	0.582
15	Cognitive load	*File Writing*	20	-0.354	0.263	0.657	416	-0.132	0.159	0.239
16	Situational interest	*While-loop*	138	0.218	0.257	0.367	927	-0.038	0.619	0.849
17	Situational interest	*ArrayList*	99	0.219	0.291	0.367	—	—	—	—
18	Situational interest	*HashMap*	52	0.161	0.489	0.489	—	—	—	—
19	Situational interest	*File Reading*	34	0.472	0.091	0.367	764	-0.016	0.849	0.849
20	Situational interest	*File Writing*	20	0.374	0.294	0.367	416	-0.211	**0.042**	0.125
Overall immediate effects (pooled across topics)										
21	Immediate performance	*Pooled*	151	5.420	**0.003**	**0.017**	961	0.173	0.760	0.761
22	Frustration	*Pooled*	151	-0.069	0.676	0.974	961	-0.135	**0.043**	0.086
23	Cognitive load	*Pooled*	151	0.004	0.974	0.974	961	-0.095	0.097	0.129
24	Situational interest	*Pooled*	151	0.240	0.079	0.196	961	-0.059	0.362	0.362
Long-term performance										
25	Exam performance	*While-loop*	135	0.746	0.390	0.780	893	0.808	0.369	0.945
26	Exam performance	*ArrayList*	98	1.656	0.201	0.780	—	—	—	—
27	Exam performance	*HashMap*	52	0.151	0.698	0.915	—	—	—	—
28	Exam performance	*File Reading*	33	0.011	0.915	0.915	763	0.005	0.945	0.945
29	Exam performance	*File Writing*	—	—	—	—	416	0.011	0.917	0.945
Post-survey outcomes										
30	Constructive engagement	*Post-survey*	16	0.460	0.369	0.443	398	0.212	**0.016**	**0.049**
31	Active engagement	*Post-survey*	16	0.891	0.072	0.144	398	0.139	0.125	0.125
32	Passive engagement	*Post-survey*	16	0.743	**0.019**	0.076	398	0.136	0.058	0.087
33	MSLQ elaboration	*Post-survey*	16	0.202	0.292	0.438	398	0.054	0.488	0.488

Table 1: Summary of intervention effects for TU Delft and UofT. Δ denotes the estimated difference between conditions (*GAT* – Text). *N* is the number of students included in each analysis after excluding invalid responses. p_{adj} values are FDR-adjusted within each site and outcome family. Bold indicates $p < .05$ and $p_{adj} < .05$. — indicates not available for that site.

4.5 Engagement profiles analysis

We conducted an exploratory engagement profile analysis based on students' CAP engagement scores. Due to the limited sample size at TU Delft ($N = 16$), the analysis was conducted only for UofT.

We applied k-means clustering ($k = 3$) to standardized CAP engagement dimensions (Passive, Active, Constructive), yielding three profiles that differed primarily in overall engagement magnitude, with each profile exhibiting consistently low-, mid-, or high-engagement scores (Figure 4 **(a)**). These engagement profiles were used to test whether the effects of *GATs* varied across them. We found no evidence of profiles' moderation for exam outcomes, cognitive load, frustration and situational interest. In contrast, profiles significantly moderated immediate learning performances ($p = .005$, $p_{adj} = .024$). Specifically, *GATs* benefited low- and high-engagement profiles, while mid-engagement profile showed a small performance decrement relative to textual explanations (Figure 4 **(b)**). No other profile interactions were observed.

5 Discussion

5.1 Effects on Learning Performance

Our findings show that AI-generated visual explanations: (1) do not harm short and long-term performance and (2) can improve immediate performance, but that these benefits are context-dependent and short-term. In the Java-based course, *GATs* produced a significant improvement in immediate performance when pooled across topics, driven primarily by the *While-loop* intervention. This suggests that

GATs are particularly effective for conceptually demanding topics where learners must reason about invisible execution processes, such as iteration and state change. In contrast, no reliable pooled immediate performance effects were observed in the Python-based course. Although some topic-level differences favored *GATs* in raw analyses, these effects did not survive correction. This divergence highlights the importance of context-specific factors, including programming language, participation structure, and course organization. Prior work suggests that Java may impose higher syntactic and semantic overhead for novices than Python, potentially increasing intrinsic cognitive load and creating greater opportunity for representational scaffolds to provide benefit [23, 26, 27].

Additionally, at UofT, participation was required (and grade-linked), which may dilute the effects if some students engage minimally with the materials. At TU Delft, participation was optional, which may yield a smaller but more engaged sample.

Across both institutions, we found no evidence that GATs influenced long-term performance. This pattern aligns with prior CS education work showing that isolated instructional interventions often improve local understanding without reliably transferring to delayed assessments [29]. Repeated integration of visual explanations may be necessary to influence long-term outcomes.

5.2 Effects on Experience and Engagement

GATs effects on immediate learning experience were modest compared to textual explanations. Across both deployments, pooled effects on frustration, cognitive load, and situational interest were small and did not survive correction. This may reflect the strength of the control condition: textual explanations were carefully constructed to provide a clear, step-by-step execution narrative, leaving limited room for additional reductions in perceived difficulty.

However, post-survey results paint a more nuanced picture. In the larger study at UofT, *GATs* produced a reliable increase in *Constructive engagement* at the end of course, while Passive and Active engagement and MSLQ elaboration showed no effect. Constructive engagement reflects sense-making behaviors such as generating inferences and connecting representations [8]. This suggests that the primary experiential benefit of *GATs* may not lie in making tasks feel easier in the moment, but rather in shaping how students engage with explanations over time. The absence of consistent immediate experience effects alongside the increase in Constructive engagement may suggest that while *GATs* may not substantially reduce frustration or cognitive load beyond a strong textual baseline, they may still encourage deeper interpretive strategies, particularly in large, heterogeneous courses where sustained engagement is difficult to support at scale.

5.3 Moderation by Engagement Profiles

An important finding of this work is that average effects may mask meaningful learner differences. Our exploratory analysis suggested that engagement profiles significantly moderated the effects of *GATs* on immediate performance.

Low- and high-engagement students benefited from *GATs* in immediate performance, while mid-engagement students showed a small performance decrement relative to textual explanations. Arguably, *GATs* serve different functions depending on learners' baseline engagement profiles. For low-engagement students, *GATs* may act as an *attention-guided scaffold*, making key state transitions explicit and reducing reliance on self-generated tracing.

In contrast, mid-engagement students who may already extract a coherent execution narrative from structured text, may experience redundancy or coordination costs when additional visual channels are introduced. This pattern is consistent with expertise-reversal effects observed in instructional design, where added scaffolding can hinder learners who no longer need it [24, 25].

Although exploratory and based on self-report measures, these findings suggest the importance of personalization. A single representation is unlikely to benefit all learners equally, and adaptive delivery based on, for example, engagement profiles may be crucial to realize the full potential of automated visual explanations.

5.4 Limitations

Our findings should be interpreted in light of several limitations. First, the two deployments differed in programming language, participation incentives, and topic coverage; accordingly, we do not draw direct cross-institutional conclusions. Second, the interventions targeted a limited set of topics, and observed effects did not transfer to long-term exam performance, suggesting more sustained integration. Third, the engagement-profile moderation analysis was exploratory and relied on self-report engagement measures, so these results should be treated as suggestive rather than confirmatory.

6 Conclusion

This study evaluated the effect of AI-generated instructional visualizations to support novice programmers across two institutional contexts. Comparing *GATs* with carefully matched textual explanations, we examined their impact on intermediate and long-term performance, learning experience, and engagement. Our results show that *GATs* can yield selective benefits for immediate learning, but these effects are context-dependent (topic, institution) and do not transfer to long-term exam performance.

At the same time, in one institution, *GATs* were associated with increased constructive engagement at the end of the course, and exploratory analysis revealed that learners' engagement profiles moderated the effectiveness of *GATs* for the immediate learning performance. Combining these findings suggests that the value of AI-generated visualizations lies not in uniformly improving outcomes, but in supporting specific learners and instructional moments. Future work should investigate adaptive, learner-aware strategies and examine whether personalized use of *GATs* can lead to more durable effects on learning outcomes.

7 Acknowledgments

This work was conducted as part of the AI for Software Engineering (AI4SE) collaboration between JetBrains and Delft University of Technology. The authors gratefully acknowledge the financial support provided by JetBrains, which made this research possible. We acknowledge the support of the Natural Sciences and Engineering Research Council of Canada (NSERC) Discovery Grant #RGPIN-2024-04348 and PGS D–600673–2025.

References

[1] Shaaron Ainsworth. 2006. DeFT: A Conceptual Framework for Considering Learning With Multiple Representations. *Learning and Instruction* 16, 3 (2006), 183–198.

[2] Roman Bednarik. 2012. Expertise-dependent visual attention strategies develop over time during debugging with multiple code representations. *International Journal of Human-Computer Studies* 70, 2 (2012), 143–155.

[3] Yoav Benjamini and Yosef Hochberg. 1995. Controlling the false discovery rate: a practical and powerful approach to multiple testing. *Journal of the Royal statistical society: series B (Methodological)* 57, 1 (1995), 289–300.

[4] Seth Bernstein, Paul Denny, Juho Leinonen, Lauren Kan, Arto Hellas, Matt Littlefield, Sami Sarsa, and Stephen Macneil. 2024. "Like a Nesting Doll": Analyzing Recursion Analogies Generated by CS Students Using Large Language Models. In *Proceedings of the 2024 on Innovation and Technology in Computer Science Education V. 1*. 122–128.

[5] Briana Bettin, Linda Ott, and Julia Hiebel. 2022. Semaphore or metaphor? Exploring concurrent students' conceptions of and with analogy. In *Proceedings of the 27th ACM Conference on Innovation and Technology in Computer Science Education Vol. 1*. 200–206.

[6] Jacob Bishop and Matthew A Verleger. 2013. The flipped classroom: A survey of the research. In *2013 ASEE annual conference & exposition*. 23–1200.

[7] Yingjun Cao, Leo Porter, and Daniel Zingaro. 2016. Examining the value of analogies in introductory computing. In *Proceedings of the 2016 ACM Conference on International computing education research*. 231–239.

[8] Michelene TH Chi and Muhsin Menekse. 2015. Dialogue patterns in peer collaboration that promote learning. *Socializing intelligence through academic talk and dialogue* 1, 2 (2015), 263–274.

[9] Michelene TH Chi and Ruth Wylie. 2014. The ICAP framework: Linking cognitive engagement to active learning outcomes. *Educational psychologist* 49, 4 (2014), 219–243.

[10] Kathryn Cunningham, Sarah Blanchard, Barbara Ericson, and Mark Guzdial. 2017. Using Tracing and Sketching to Solve Programming Problems: Replicating and Extending an Analysis of What Students Draw. In *Proceedings of the 2017 ACM Conference on International Computing Education Research*. 164–172.

[11] Björn B de Koning and Halszka Jarodzka. 2017. Attention guidance strategies for supporting learning from dynamic visualizations. In *Learning from dynamic visualization: Innovations in research and application*. Springer, 255–278.

[12] Dimitri Eckert, Dion Timmermann, and Christian Kautz. 2022. Student Misconceptions About Loops in Introductory Programming Courses and the Influence of Representations. In *2022 IEEE Frontiers in Education Conference (FIE)*. IEEE, 1–5.

[13] Sally Fincher, Johan Jeuring, Craig S Miller, Peter Donaldson, Benedict Du Boulay, Matthias Hauswirth, Arto Hellas, Felienne Hermans, Colleen Lewis, Andreas Mühling, et al. 2020. Notional Machines in Computing Education: The Education of Attention. In *Proceedings of the Working Group Reports on Innovation and Technology in Computer Science Education*. 21–50.

[14] Michal Forišek and Monika Steinová. 2012. Metaphors and analogies for teaching algorithms. In *Proceedings of the 43rd ACM technical symposium on Computer Science Education*. 15–20.

[15] Rita Garcia and Michelle Craig. 2025. 20 Years Later: A Replication Study on Teaching CS1 Concepts. *ACM Trans. Comput. Educ.* 25, 2, Article 22 (June 2025), 33 pages. doi:10.1145/3730405

[16] Philip J Guo. 2013. Online Python Tutor: Embeddable Web-Based Program Visualization for Cs Education. In *Proceeding of the 44th ACM Technical Symposium on Computer Science Education*. 579–584.

[17] Philip J Guo. 2018. Non-native english speakers learning computer programming: Barriers, desires, and design opportunities. In *Proceedings of the 2018 CHI conference on human factors in computing systems*. 1–14.

[18] Tran Trieu Hai, Duong Thi Thuy Mai, and Nguyen Van Hanh. 2025. A rapid review of using AI-generated instructional videos in higher education. *Frontiers in Computer Science* 7 (2025), 1721093.

[19] Colton Harper, Jake Rance, Paul Owens, and Stephen Cooper. 2024. Tool-Driven Scaffolding of Student-Generated Analogies in CS1. In *Proceedings of the 8th Conference on Computing Education Practice*. 5–8.

[20] Sandra G Hart and Lowell E Staveland. 1988. Development of NASA-TLX (Task Load Index): Results of empirical and theoretical research. In *Advances in psychology*. Vol. 52. Elsevier, 139–183.

[21] Bradley Huitema. 2011. *The analysis of covariance and alternatives: Statistical methods for experiments, quasi-experiments, and single-case studies*. John Wiley & Sons.

[22] Philip Nicholas Johnson-Laird. 1983. *Mental Models: Towards a Cognitive Science of Language, Inference, and Consciousness*. Number 6. Harvard University Press.

[23] Erkki Kaila, Matti Luukkainen, Antti Laaksonen, and Kjell Lemström. 2023. On Changing the Curriculum Programming Language from Java to Python (Discussion Paper). In *Proceedings of the 23rd Koli Calling International Conference on Computing Education Research*. 1–7.

[24] Slava Kalyuga. 2007. Expertise reversal effect and its implications for learner-tailored instruction. *Educational Psychology Review* 19, 4 (2007), 509–539.

[25] Slava Kalyuga. 2021. The expertise reversal principle in multimedia learning. Cambridge University Press.

[26] Theodora Koulouri, Stanislao Lauria, and Robert D. Macredie. 2015. Teaching Introductory Programming: A Quantitative Evaluation of Different Approaches. *ACM Trans. Comput. Educ.* 14, 4, Article 26 (Dec. 2015), 28 pages. doi:10.1145/2662412

[27] Erno Lokkila, Athanasios Christopoulos, and Mikko-Jussi Laakso. 2023. A data-driven approach to compare the syntactic difficulty of programming languages. *Journal of Information Systems Education* 34, 1 (2023), 84–93.

[28] Henry B Mann and Donald R Whitney. 1947. On a test of whether one of two random variables is stochastically larger than the other. *The annals of mathematical statistics* (1947), 50–60.

[29] Lauren E Margulieux, Briana B Morrison, and Adrienne Decker. 2020. Reducing withdrawal and failure rates in introductory programming with subgoal labeled worked examples. *International Journal of STEM Education* 7, 1 (2020), 19.

[30] Richard E Mayer, Emily Griffith, Ilana TN Jurkowitz, and Daniel Rothman. 2008. Increased interestingness of extraneous details in a multimedia science presentation leads to decreased learning. *Journal of Experimental Psychology: Applied* 14, 4 (2008), 329.

[31] Torbjørn Netland, Oliver von Dzengelevski, Katalin Tesch, and Daniel Kwasnitschka. 2025. Comparing human-made and AI-generated teaching videos: An experimental study on learning effects. *Computers & Education* 224 (2025), 105164.

[32] Yuri Noviello, Anastasia Birillo, and Gosia Migut. 2026. ANVIL: Analogies and Video for Lecturers. In *Artificial Intelligence in Education (Lecture Notes in Computer Science)*. Springer. Accepted for publication in the proceedings of AIED 2026.

[33] Fred GWC Paas. 1992. Training Strategies for Attaining Transfer of Problem-Solving Skill in Statistics: A Cognitive-Load Approach. *Journal of Educational Psychology* 84, 4 (1992), 429.

[34] Allan Paivio. 1991. Dual coding theory: Retrospect and current status. *Canadian Journal of Psychology/Revue canadienne de psychologie* 45, 3 (1991), 255.

[35] José C Pinheiro and Douglas M Bates. 2000. *Mixed-effects models in S and S-PLUS*. Springer.

[36] Paul R Pintrich et al. 1991. A manual for the use of the Motivated Strategies for Learning Questionnaire (MSLQ). (1991).

[37] Richard M Ryan and Edward L Deci. 2024. Self-determination theory. In *Encyclopedia of quality of life and well-being research*. Springer, 6229–6235.

[38] Pawan Saxena, Sanjay Kumar Singh, and Gopal Gupta. 2023. Achieving effective learning outcomes through the use of analogies in teaching computer science. *Mathematics* 11, 15 (2023), 3340.

[39] Naaz Sibia, Valeria Ramirez Osorio, Jessica Wen, Rutwa Engineer, Angela Zavaleta Bernuy, Andrew Petersen, Michael Liut, and Carolina Nobre. 2025. From Code to Concept: Evaluating Multiple Coordinated Views in Introductory Programming. arXiv:2509.26466 [cs.HC] https://arxiv.org/abs/2509.26466

[40] Juha Sorva, Ville Karavirta, and Lauri Malmi. 2013. A review of generic program visualization systems for introductory programming education. *ACM Transactions on Computing Education (TOCE)* 13, 4 (2013), 1–64.

[41] John Sweller. 2011. Cognitive load theory. In *Psychology of Learning and Motivation*. Vol. 55. Elsevier, 37–76.

[42] Lynda Thomas, Mark Ratcliffe, and Benjy Thomasson. 2004. Scaffolding With Object Diagrams in First Year Programming Classes: Some Unexpected Results. *ACM SIGCSE Bulletin* 36, 1 (2004), 250–254.

[43] Rachel M Wong, Olusola Adesope, Chi Yang Chuang, Oluwasola S Oni, Bernie Vanwie, Prashanta Dutta, Kitana Kaiphanliam, Felicia Adesope, Oluwafemi J Ajeigbe, and Jacqueline Gartner. 2024. Engineering students engagement profiles while using low-cost desktop learning module. *IJEE International Journal of Engineering Education* (2024).

[44] Tao Xu, Yuan Liu, Yaru Jin, Yueyao Qu, Jie Bai, Wenlan Zhang, and Yun Zhou. 2025. From recorded to AI-generated instructional videos: A comparison of learning performance and experience. *British Journal of Educational Technology* 56, 4 (2025), 1463–1487.

[45] Tingting Zhu, Rutwa Engineer, Xaria Prempeh, Anna Ly, Michelle Craig, and Andrew Petersen. 2025. Comparing physical analogue and traditional videos for learning and emotional engagement. *Discover Education* 4, 1 (2025), 71.