Zavgorodniaia, Albina; Hellas, Arto; Seppälä, Otto; Sorva, Juha

Should Explanations of Program Code Use Audio, Text, or Both? A Replication Study

Published in:
Proceedings - 20th Koli Calling Conference on Computing Education Research, Koli Calling 2020

DOI:
10.1145/3428029.3428050

Published: 19/11/2020

Document Version
Peer reviewed version

Please cite the original version:
Zavgorodniaia, A., Hellas, A., Seppälä, O., & Sorva, J. (2020). Should Explanations of Program Code Use Audio, Text, or Both? A Replication Study. In *Proceedings - 20th Koli Calling Conference on Computing Education Research, Koli Calling 2020* (pp. 1-10). [5] ACM. https://doi.org/10.1145/3428029.3428050

Should Explanations of Program Code Use Audio, Text, or Both? A Replication Study

Albina Zavgorodniaia, Arto Hellas, Otto Seppälä, and Juha Sorva
{albina.zavgorodniaia,arto.hellas,otto.seppala,juha.sorva}@aalto.fi
Aalto University, Finland

ABSTRACT

Studies in educational psychology suggest that people learn better when visual learning materials are accompanied by audio explanations rather than textual ones. Research on how this *modality effect* applies to computing education is scarce and inconclusive. We explore whether modality of instruction affects learning from videos that use a series of example programs to explain how variables work in Python. Learners (n=186) were crowdsourced from the internet and randomized in three groups, who received explanations as audio, text, or both, respectively. We did not find significant differences between the groups in near transfer to code-tracing tasks or perceived cognitive load. The result affirms the need to further investigate instructional modalities in programming education. There are a number of theoretical, methodological, and instructional-design factors that may explain these and earlier findings; we trace out future research that could explore those factors.

CCS CONCEPTS

• **Social and professional topics** → Computing education.

KEYWORDS

modality effect, cognitive load, introductory programming, learning from examples, replication

ACM Reference Format:
Albina Zavgorodniaia, Arto Hellas, Otto Seppälä, and Juha Sorva. 2020. Should Explanations of Program Code Use Audio, Text, or Both? A Replication Study. In *Koli Calling '20: Proceedings of the 20th Koli Calling International Conference on Computing Education Research (Koli Calling '20), November 19–22, 2020, Koli, Finland, Finland*. ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3428029.3428050

1 INTRODUCTION

Programming students at all levels study example programs. The examples may be written in a textbook, presented by a teacher in class, or viewed as video tutorials online.

The use of video, in particular, has increased dramatically in education over the past two decades [21]. Programming teachers are creating more and more instructional videos that explain example code, using the videos in pedagogies such as the flipped classroom [4, 16, 20, 86, 91, 100, 101], sharing them with other practitioners [89], and embedding them in ebooks [22, 26]. Others have asked learners to create programming videos for peer instruction [25] or crowdsourced video-like program animations from learners [32]. For better or worse, many online courses have students spend a lot of their study time watching video lectures (e.g., [12]), while in informal learning, many learners have embraced video-based activities such as watching experts program either live or after-the-fact [33, 73]. Tools are being developed to enhance learning from programming videos and live streams and to assist in their creation [17, 46, 52].

To help learners understand example programs and to highlight key lessons, teachers commonly accompany example code with verbal explanations. The code itself is usually presented visually, but explanations of the code may rely on different sensory *modalities*; in practice, explanations are provided either in writing, as spoken words, or both.

When live-coding [92] in class, teachers usually explain examples through the audio modality; for explanations of pre-prepared example code, they may use either audio alone or a combination of audio and text (e.g., annotations on a PowerPoint slide). Instructional videos similarly tend to explain code using audio or both modalities. In traditional textbooks, explanations are textual, but ebooks may accompany or supplant text with spoken explanations; some ebooks incorporate videos. Program-animation tools illustrate code with educational diagrams or other graphics, often accompanied by textual explanations; audio support is rare in these tools [94], but not nonexistent [6, 22, 90], and of course audio may be added to an example by capturing a program animation on video.

In the present work, we explore the question of which modality is the most effective when explaining computer programs to non-disabled novice learners.

As we shall discuss in more detail below, studies in educational psychology suggest that audio explanations tend to be effective; this is known as the *modality effect*. Based on that research, we might expect that it is easier for a learner to listen to an audio explanation while looking at code than it is to shift their visual attention to an accompanying textual explanation. However, there are several qualifications to this advice from psychological research, as the superiority of audio depends on several factors, vanishing altogether or even reversing under some circumstances. Moreover, there is a dearth of research investigating whether and how that purportedly general advice applies in the programming domain.

One study directly addresses this question, however: In 2017, Morrison [66] conducted an experiment comparing textual and auditory explanations of example programs. She did not find evidence for the modality effect—i.e., no significant difference between the

Koli Calling '20, November 19–22, 2020, Koli, Finland, Finland

ACM ISBN 978-1-4503-8921-1/20/11…$15.00
https://doi.org/10.1145/3428029.3428050

modalities. Morrison notes that her findings are open to interpretation and concludes that "More studies should be conducted."

In this article, we report on a study that replicates Morrison's with a different cohort of learners crowdsourced from the internet and a different set of program examples on video. Like Morrison, we present groups of learners with audio explanations, textual explanations, or both, respectively, and find no significant differences in learning gains or cognitive-load measurements between the groups. We contribute a commentary on the possible reasons behind these findings, which include confounding factors in our study design as well as broader issues in the psychology of programming.

2 RELATED WORK

2.1 The Modality Effect

In essence, the modality effect is the finding that people learn better from a combination of pictures and audio than from a combination of pictures and text. This effect is robustly documented by experiments in educational psychology across a number of domains of learning; for reviews and meta-analysis, see [30, 43, 63, 64]. Its main implication for instruction is to replace written explanations of diagrams and other pictures with spoken narration.

The related *verbal redundancy effect* [43, 44, 63, 64] refers to the ineffectiveness of presenting the same information using multiple modalities, such as when reading out text on a PowerPoint slide. That is, only hearing instructional explanations in audio or only reading them as text is often better than both hearing and reading the same explanations. Although the verbal redundancy effect can be considered as a separate finding in its own right, it also effectively extends the modality effect by recommending that redundancy between modalities is eliminated.

2.1.1 Cognitive Foundations of the Modality Effect. As a theoretical foundation, the majority of studies on the modality effect cite *dual-processing* models of human cognition [63, 64]. These models posit that people have distinct subsystems, or "processors," for verbal and pictorial information, each with its own, very limited capacity and duration. More specifically, Baddeley's [7] influential model of working memory incorporates a *phonological loop* and a *visuospatial sketchpad* for processing words and images, respectively. Written text is processed first visually, then auditively.

In terms of dual-processing theory, looking at pictures with accompanying text is prone to overloading the visual subsystem of working memory. This is in part because there is more visual information to be processed and in part due to having to split one's attention between multiple visual cues (picture and text) and to connect them mentally. In contrast, listening to speech while looking at pictures "offloads" some of the information to the auditory subsystem and alleviates the split-attention problem. Instruction that combines pictures with speech therefore makes fuller use of working-memory capacity compared to pictures with text. (See, e.g., [43, 50, 63, 64].) The theory further suggests that presenting the same information as both speech and text (verbal redundancy) has the same danger of visual overload and, to boot, requires the learner to mentally relate the speech and the text, further burdening working memory. [43, 44, 63, 64]

Since working-memory models like Baddeley's [7] posit that text is—after being fleetingly perceived visually—retained in the phonological loop, the above dual-processing explanation for the modality effect has been criticized for overemphasizing the role of visual processing of textual explanations [81]. Another criticism is the traditional models' silence on additional modalities such as human movement and haptics [85]. Other established models of working memory exist besides Baddeley's dual-processing model; Cowan's model [19] is an example, and Adams et al. [1] review others. Building on that earlier work, Sepp et al. [85] have recently proposed a working-memory model that—rather than positing separate "processors"—centers on a focus of attention that shifts between competing sensory and non-sensory foci of various modalities and has a limited scope. The modality effect would then be explained by a combination of factors including interference between multiple visual foci (pictures and text) and people's superior retention of auditory over visual information (inner "echo") [85].

What these different explanations of the modality effect have in common is that they highlight the severe constraints on humans' conscious processing of new information and suggest that using a complementary combination of modalities makes it easier to work around those constraints.

2.1.2 Factors that Moderate the Modality Effect. Although there is evidence for the modality effect from dozens of studies, the effect does not occur universally in all circumstances. Instead, there are a number of "boundary conditions" [64] on when we may expect a modality effect. Unless these conditions are met, it may be equally effective—or even better—to accompany pictures with text than with audio. We list some of the main boundary conditions below.

First, for the modality effect to occur, the pictorial and verbal information must be *mutually unintelligible* [43]. If the words merely describe what is obvious from the picture, or vice versa, the redundant explanations merely add working-memory load and may hinder learning. Moreover, whether the pictures and words are intelligible on their own depends on prior knowledge, so domain-experts are able to deal with some materials no matter the modality.

Second, the modality effect is likelier and stronger on *complex learning materials*, which means materials that feature interacting or interdependent elements [30, 43, 63, 64]. For example, a cause-and-effect description of a system is more complex than a list of isolated facts. Of course, if the materials are *too* complex, there will be little learning irrespective of modality. Like mutual intelligibility, complexity depends on expertise.

Third, explicit *signaling* may be required for the modality effect to appear [63, 64, 78, 82]. Unless the connections between parts of the picture and their spoken explanations are very obvious, the learner has to visually search for the relevant parts while keeping track of the words. This additional burden can negate the effect.

Fourth, the modality effect may only apply to explanations that are meaningfully portioned in relatively *short segments* [43, 50, 63, 64, 77]. If each segment is long or incoherent, spoken explanations are prone to being poorer than written ones (a reverse modality effect). With longer segments, there is also a higher risk of missing an important but transient piece of a spoken explanation, whereas text allows glancing back.

Fifth, and relatedly, *pacing* matters [30, 63, 64, 77]. Especially with novice learners, the modality effect is more likely when the learner cannot control when to proceed to the next piece of the

explanation (e.g., in a mass lecture or a non-stop video). If there are pauses and the learner gets to control when to resume, the modality effect may vanish or reverse, as there is more time for reading and integrating text with pictures. On the other hand, relying on learners to pause a video appropriately is liable to fail, as many learners do not pause often enough [11, 27]. Learner control of pacing is particularly important if there is no signaling or if the explanatory segments are long. Very slow-paced presentations may not exhibit the modality effect.

Sixth, and finally, if the explanations use *unfamiliar words*, such as when the learner is not a native speaker of the language of instruction, text can be relatively helpful. Under such circumstances, even the otherwise inadvisable combination of identical speech and text can be beneficial as subtitling. [43, 63, 64]

2.2 Cognitive Load Theory

The modality effect is often discussed in terms of *cognitive load theory* [97, 98], which explains how working-memory limitations hamper the acquisition of complex knowledge in long-term memory. Cognitive-load research has identified various "effects" that guide instructional design to exploit the human cognitive apparatus. The worked-example effect, which promotes studying examples over solving problems, is one example; the modality effect is another.

Cognitive load is the intensity of cognitive activity required by a learning situation [45]. It is estimated in terms of *element interactivity*: the number of related elements the learner must simultaneously and consciously hold in working memory. Since working memory is extremely limited, it is easily overloaded, causing learning to fail.

Overall cognitive load can be analytically separated in two [42, 96, 98]: *intrinsic load* and *extraneous load*. The former is unavoidable without compromising learning objectives; the latter is avoidable load that is caused by suboptimal materials and activities.

Earlier formulations of cognitive load theory postulated a third, distinct load component, *germane load*, representing additional load that leads to learning. The theory has since been reformulated [42, 96, 98], but the germane-load concept still features in, for example, measurement instruments (including one that we use below).

2.2.1 *Cognitive Load and the Modality Effect.*

In terms of cognitive load theory, accompanying pictures with text increases extraneous load, as it unnecessarily requires the learner to search for relations between visual elements [68]. When the learning goal is non-trivial given the learner's prior knowledge—i.e., when intrinsic load is high—the increase in extraneous load easily results in a failure to learn. Audio explanations of pictures, signaling, and segmenting reduce extraneous load. Verbal redundancy from identical text and speech further increases extraneous load. Empirical evidence links the modality and verbal redundancy effects to higher cognitive-load measurements [43].

2.2.2 *Measuring Cognitive Load.*

Researchers have come up with an assortment of methods for approximating cognitive load empirically; for recent reviews, see [47, 98, 106]. The most common approach is subjective post-hoc self-assessment: learners are asked to rate one or more aspects of a learning activity that they just engaged in, such as their perceived mental effort during the activity.

One notable example is the self-assessment instrument by Leppink et al. [54], which consists of ten items such as:

- "The activity covered formulas that I perceived as very complex." (One of three items targeting intrinsic load.)
- "The instructions and/or explanations during the activity were very unclear." (One of three items targeting extraneous load.)
- "The activity really enhanced my knowledge and understanding of statistics." (One of four items targeting germane load.)

Leppink and colleagues' instrument has been since adapted from the original domain of statistics to various others, including a programming-specific adaptation by Morrison et al. [67]. In CER, Morrison and colleagues' version of the instrument has been used in a variety of studies (e.g., [23, 36, 59, 66, 92, 99]). There is evidence of the instrument's internal reliability and discriminant validity [54, 67, 105], but doubts remain over its construct validity, especially with respect to the items that target germane load [40, 53, 55, 62, 105].

2.3 The Modality Effect in Programming

There is little research on the modality effect in computing education. In this article, we replicate an experiment by Morrison [66], which is to our knowledge the only one to address the question of whether written or spoken explanations of program code result in better learning.

Morrison [66] recruited participants from introductory-level computer science courses at multiple U.S. universities, filtering out participants that scored over 67% on a pre-test on programming. The 61 participants were divided in three conditions that received explanations of program code as audio only, text only, or both, respectively. Each group viewed three videos of different lengths (5, 23, and 12 minutes); the videos were the same for each condition, except for the presentation modality. The hypothesis was that since "people see code segments more as a diagram rather than text to be read" [66], a modality effect could be expected.

Each video stated a problem with a "real-world" context (e.g., generating an invoice), presented an example program as a solution, and traced through the program's execution. The central programming concepts of the three videos were assignment, nested selection, and iteration, respectively. The programs were written in Python, and two of them had textual comments embedded in the code. The videos were system-paced (except for a single pause in the middle that ended when the learner wished) and visually signaled the parts of the program code that were being verbally explained. After each video, the participants answered the Morrison et al. [67] cognitive load questionnaire and a post-test.

Morrison's [66] post-tests asked the participants to verify that they understood the preceding program's purpose (which had been stated on the video), to recall the purpose of selected parts of the program, to trace segments of the original program, and to transfer their knowledge to a different but similar program.

Against expectations, Morrison found no support for the modality and verbal redundancy effects. The differences between the three conditions in post-test performance were not statistically significant, nor were the differences in the groups' responses to the cognitive-load survey. What is more, the descriptive differences between the conditions had a mixed pattern that did not suggest a modality effect; for example, the best average performance on transfer questions was that of the text-only group.

Morrison [65, 66] discusses various possible reasons for the lack of a modality effect. Some of these reasons arise from the particular materials used; for example, since transfer performance on the post-test was poor overall, and since the videos covered a lot of content, the lessons may have been too complex to be learned. Another possibility is that code is not viewed "like a picture," at least by novices; we will return to this topic in Discussion.

The lack of evidence for a modality effect in Morrison's [66] study has been cited as a possible indication that programming is more complex than other disciplines and that pedagogies that work in science education do not work in computer science [104, 107]. Nevertheless, as Morrison writes, "it is possible that the modality effect does not hold within learning programming, but we cannot conclude that based on this study alone." [66]

3 RESEARCH GOALS AND HYPOTHESES

Our paper's driving question is in the title: "Should explanations of program code use audio, text, or both?" In other words: "Do the modality and verbal redundancy effects apply to explanations of program code?" More specifically, we evaluate these hypotheses:

H1 *Learners who receive audio explanations as they study example code learn more than learners who receive the same explanations as text.*

- H1a *The audio learners will outperform the text learners on a post-test of conceptual knowledge and program tracing.*
- H1b *The text learners will report higher extraneous load and similar intrinsic load compared to the audio learners.*

H2 *Learners who receive identical explanations of example code simultaneously as audio and text learn less than learners who receive the same explanations as audio only or text only.*

- H2a *The text-only learners and the audio-only learners will both outperform the audio-with-text learners on the post-test.*
- H2b *The audio-with-text learners will report higher extraneous load and similar intrinsic load compared to the learners in the single-modality conditions.*

We replicate Morrison's [66] programming-domain study with a larger sample size, a different cohort of learners, and a different set of video tutorials. We also replicate, by extension, the modality studies in other domains that Morrison replicated. In doing so, we respond to requests for more replication studies in CER [35] and for more varied replications of multimedia-learning principles in different domains and with learners other than the typical cohort of young college-student volunteers [64, Chapter 20].

Aside from being an experiment on the modality effect, our project is intended as a methodological exploration of crowdsourcing learners for studies of learning to program. We report on that aspect of our work in a separate article [37].

4 METHODS

We conducted a randomized controlled experiment where each participant, a beginner in programming, received instruction with either audio or text narration or both. The participants' learning and cognitive load were then assessed to test the above hypotheses.

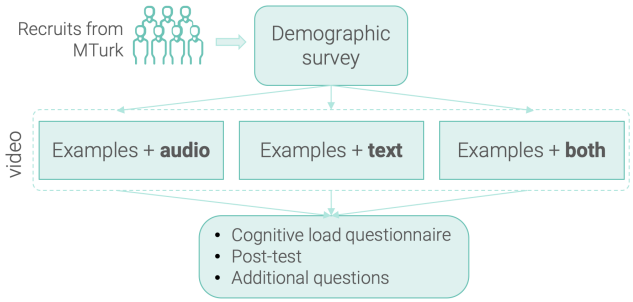

Figure 1: The structure of the experiment

The participants were crowdsourced from Amazon Mechanical Turk (www.mturk.com). The experiment then took place online on a web site under our control, in April 2020.

4.1 Procedure

The experiment consisted of three parts: a demographic survey, an instructional video, and finally a cognitive-load questionnaire and a post-test on what was learned (Figure 1). We allotted 45 minutes for the whole experiment, with one hour as the hard limit.

The demographic survey included a question about programming experience blended into a series of questions about other kinds of skills, activities, and typical background information. This allowed us to filter responses and focus on participants with low prior knowledge while keeping potential participants uninformed about the upcoming task.

Each participant was randomly assigned to one of three conditions: audio explanations (*Audio*), textual explanations (*Text*), or simultaneous audio and textual explanations (*Both*). They were then shown an instructional video on programming that matched the assigned condition; the videos were otherwise identical in content.

After viewing the video, the participants were asked to fill out a cognitive-load questionnaire [67], then to complete a post-test on programming, and finally to respond to a few questions about their interests, computing-related anxiety, and thoughts about the session they had just participated in.

4.2 Instructional Materials

Like Morrison [66], we showed the learners videos that explain programs written in a text-based programming language. Unlike her, we used only one video per group, rather than a series of videos.

4.2.1 Video Contents. As suggested by Morrison [65, 66], we attempt to demonstrate the modality effect on simpler content than hers. Our video was a beginner-level lesson on reading and tracing imperative Python programs and focused on a limited set of key concepts: expressions, values, and assignment and print statements.

At 24 minutes, the video was roughly equal in length to the longest of Morrison's [66] three videos. Unlike hers, our video did not cover a single relatively complex program but a sequence of tiny programs that gradually introduced the key concepts. In total, there were twelve programs, ranging from two to seven lines of code in length (median 3.5). The programs were accompanied by verbal explanations consisting of 2377 words in total.

Our programs differed from Morrison's in that they were decontextualized examples of programming constructs rather than

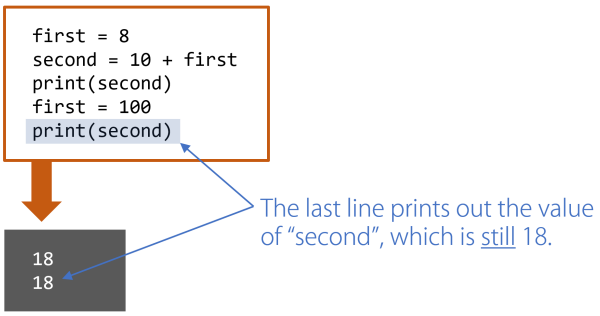

Figure 2: One of the example programs from the videos, its output, and an accompanying textual explanation. The relevant parts of the code and output are signaled to the learner.

solutions to contextualized problems.[1] Our code also did not feature any in-code comments; all explanations were external to the code. No control structures were used, so the control flow was linear. The examples featured only integer data.

To "portion" intrinsic cognitive load, two of the longer programs were extensions of earlier ones: they were identical to the preceding example except for additional statements at the end.

For each program, the relevant concepts were explained, the program's behavior was traced in detail, and its output was shown. Fig. 2 shows a screenshot. Overall, the videos demonstrated a way of reasoning about the behavior and output of such code.

The programs and explanations were designed to attend to several misconceptions that beginners have about variables in programming (e.g., inappropriate analogies with school mathematics; variables storing multiple values [93]). Moreover, two of the twelve example programs had deliberate errors—an uninitialized variable and swapped assignment syntax–which were explained to the learners. This aligns with the advice from learning-from-examples research to use erroneous examples, although our videos merely explained the errors to learnerss rather than requiring learners to explain them, as in some studies [8, 39].

4.2.2 Presentation Design. Following the advice from research on the modality effect (Section 2.1.2 above), the videos visually signaled which part of the program code and the corresponding output was being explained at each point. Figure 2 illustrates how this looked for the Text and Both groups; the Audio group saw similar highlights but without the linked explanatory texts.

Given our focus on explaining program code, we deliberately avoided using any instructional visualizations of program execution (such as diagrams of variable values) beyond showing each program's output and the signaling highlights.

Since the modality effect is likelier under system-paced conditions (Section 2.1.2 above), we configured the videos so that once the *Play* button was pressed, it was not possible to stop, pause, slow down, speed up, or rewind. Log data of playback was collected so that "cheating" participants could be excluded from the analysis.

For the Audio and Both groups, the explanations were narrated in English by a fluent but non-native English-speaker. We did not display an image of the speaker or model the writing of the programs on a real device (and so did not follow *that* advice on effective video design [63, 64]). The narrator vocally stressed key ideas and words. Overall the presentation was calm and matter-of-fact in tone rather than obviously enthusiastic (cf. [9, 56]). The parts that were strongly stressed in speech were also highlighted in the textual explanations. (The underlined word in Figure 2 is an example.)

The narration was paced at approximately 150 words per minute, which is a recommended rate for multimedia instruction [103] and similar to the pace in other modality studies [50, 80, 95]. The time for reading each slide in the Text condition was equal to that in the Audio and Both conditions.

There were no pauses longer than a few seconds, but we attempted to pace the narration with "natural" pauses in speech, leaving a little bit of time for the learner to integrate what they saw and/or heard. The textual explanations of code arrived onscreen in chunks of one, two, or occasionally three simple sentences.

4.3 Participants

The participants were recruited and paid via Amazon Mechanical Turk (MTurk). MTurk has emerged as a platform for crowdsourcing research participants and is increasingly used in CER as well (e.g. [34, 51, 61]). MTurk's utility for scientific data collection has been assessed in multiple studies, and the platform has been shown capable of providing fairly rich and diverse data about people [41, 69]. Buchheit et al. [13] suggest that, in the accounting domain, MTurk workers' task performance is similar to undergraduates'; the authors conclude that MTurkers are good research participants for tasks that require general reasoning and problem-solving ability but no previous knowledge. However, the platform is also known for variable data quality [14, 38], so potential participants have to be carefully screened.

During screening, we initially accepted only workers who had completed at least 100 tasks on MTurk and had had at least 98% of their tasks approved by the task requester. With the screening already underway, we chose to raise those constraints to 500 and 99%, respectively. To avoid selection bias, we did not mention programming in the task description.

A total of 307 MTurk workers took part in the experiment. Of them, 61 were excluded for a variety of reasons such as partially missing responses or suspect data (e.g., unnaturally fast responses, responding using linear patterns). Of the 246 remaining workers, 60 reported having more than a little experience in programming; since our focus is on novices, we excluded them from the analysis. This left us with 186 participants. (24 of these 186 were recruited during the initial screening phase with looser constraints on participation.)

We observed no significant differences in the post-test results and cognitive-load scores between the participants who reported no prior programming experience ($n = 131$) and those who reported "a little" experience ($n = 55$). We chose to analyze both these groups together as a single cohort.

Of the 186 participants, 83 self-identified as men and 103 as women, with no-one picking the other options. Most (158) were from the U.S., fourteen were from India, and the remainder from various countries. English was 168 participants' native language; the rest spoke a mix of languages. Seven of the participants were

[1] The purposelessness of these example code fragments was briefly brought up in the videos, which explained that the computer follows any instructions literally and systematically, so it is up to the programmer to make interesting use of the constructs.

between 18–24 years of age, 64 were between 25–34, 56 between 35–44, 38 between 45–54, and 21 were at least 55 years old. 109 of the participants held a bachelor's degree, 23 a master's, 2 a doctoral degree, and 52 were school graduates. 123 participants reported having some background in the humanities, 44 in natural and technical sciences, and 19 reported no education beyond primary school.

After random division into groups, the Audio, Text, and Both conditions had 59, 64, and 63 participants, respectively.

4.4 Cognitive-Load Questionnaire

We estimated cognitive load by asking the the participants to answer the cognitive-load self-assessment instrument for the programming domain, which that Morrison et al. [67] had adapted from Leppink and colleagues' original [54]. This is the same questionnaire that Morrison [66] used in her modality study.

We confirmed the instrument's internal reliability and discriminant validity by replicating Morrison and colleagues' [67] analyses on our data set (as detailed in [105]).

A minor difference between our instrument and Morrison and colleagues' is that we used a ten-point scale (from one to ten) while they used an eleven-point one.

4.5 Post-Test

4.5.1 Questions. The post-test included seven transfer questions where the participants had to apply their knowledge from the video to tracing tiny programs. These were followed by a "theoretical" multiple-choice question about how variables work.

The transfer questions each presented a piece of Python code and asked the participant to predict its behavior. Participants were asked for two things: 1) whether the code would run without error; and 2) if so, what the program's output is, and if not, what the reason for the failure is. These answers were given in free form.

The programs in the transfer tasks were similar but not identical to the ones on the preceding video. Two of the seven programs (the second and fourth) were written to produce en error. The other five programs would print out at least one line of output.

4.5.2 Scoring. The seven transfer questions were each given a binary mark of 1 (correct) or 0 (incorrect) according to the following criteria. If the program ran without error and produced an output:

- One point was awarded if the participant predicted an error-free run and the complete, correct output.
- Zero points were awarded if the participant's justification for the output was incorrect, clearly incomplete (e.g., only one of two outputs), or missing entirely.

If the program did result in an error:

- One point was awarded if the participant predicted that an error occurs and correctly explained why.
- Zero points were awarded if the participant's justification for the error was incorrect or missing.

The multiple-choice question presented seven statements about variables, which the participant had to evaluate as being true or false on the basis of the video. There were three true statements and four false ones. The maximum score for this question was three: one point was awarded for each true statement correctly identified, and one point was taken away for each false statement incorrectly identified as true (down to a minimum of zero).

5 RESULTS

5.1 Post-Test Performance

The groups' total scores on the post-test did not differ significantly (one-way ANOVA: $F = 0.6$, $p = 0.6$, $d = 0.006$). Similarly, no significant difference in time-on-task was observed ($F = 0.4$, $p = 0.7$, $d = 0.004$). The scores were not normally distributed.

The means, medians, and standard deviations of the post-test totals and times-on-task are shown in Table 1. Table 2 details, for each modality group and each question on the post-test, the percentage of participants who answered that question correctly. Figure 3 shows the distribution of post-test totals by group.

Table 1: Post-test totals and times-on-task.

		Mean	Median	Stdev
Audio (n=59)	Score (from 10)	4.8	5.0	2.5
	Time (seconds)	380	292	290
Text (n=64)	Score	5.3	5.0	2.5
	Time	380	330	190
Both (n=63)	Score	5.0	5.0	2.6
	Time	420	330	360

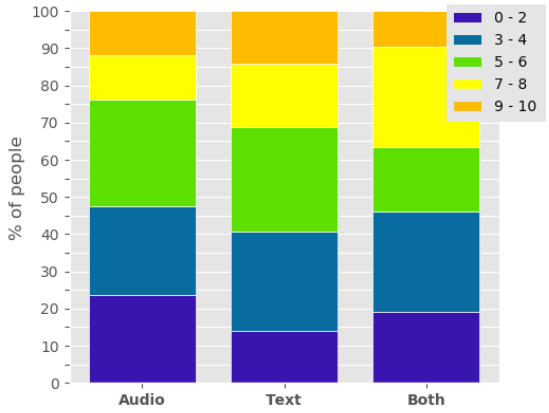

Figure 3: Distributions of post-test totals.

Table 2: Percentages of correct answers on each post-test question.

		Audio (n=59)	Text (n=64)	Both (n=63)
Transfer question 1		68%	76%	71%
Transfer question 2		54%	75%	59%
Transfer question 3		63%	54%	68%
Transfer question 4		36%	31%	30%
Transfer question 5		32%	37%	41%
Transfer question 6		40%	39%	40%
Transfer question 7		31%	34%	40%
Multiple-choice	3/3	32%	27%	33%
	2/3	14%	21%	6%
	1/3	36%	36%	41%
	0/3	18%	16%	20%

5.2 Cognitive-Load Scores

Table 3 shows the means, medians, and standard deviations from the self-assessment items that targeted different types of cognitive load. In that and the others tables, IL, EL, and GL refer to the respective means of the questionnaire items that target intrinsic load, extraneous load, and germane load.

None of the cognitive-load scores were normally distributed. There was a notable floor effect on the items targeting extraneous load, with means in the 1.6–2.0 range on a scale from one to ten.

A one-way ANOVA test across the modalities did not indicate significant differences in any of the cognitive-load components ($F = 0.1$, $p = 0.9$ for intrinsic load; $F = 0.9$, $p = 0.4$ for extraneous load; $F = 1.3$, $p = 0.3$ for germane load).

Table 4 shows the Pearson correlations between all three types of cognitive load and post-test scores, in aggregate for all three groups. As the table indicates, many of these measures correlated weakly or moderately. Table 5 expands on that analysis by showing, for each group separately, the correlations between the cognitive-load measures and post-test scores.

We checked for but did not find significant correlations between any scores (post-test or cognitive load) and time-on-task.

Table 3: Subjectively assessed cognitive-load scores (ranging from 1 to 10).

		Mean	Median	Stdev	N
Audio	IL	4.0	4.0	2.1	59
	EL	2.0	1.3	1.7	
	GL*	6.4	6.8	2.5	
Text	IL	4.0	3.3	2.2	64
	EL	1.6	1.0	1.1	
	GL*	6.6	6.0	2.1	
Both	IL	4.2	4.0	2.3	63
	EL	1.7	1.0	1.3	
	GL*	7.1	7.8	2.7	

*GL-targeting items positively worded; 10 means higher perceived learning.

Table 4: Correlations between cognitive-load components and post-test scores, with the three groups aggregated.

	IL	EL	GL	Post-test
Intrinsic (IL)	1	0.5	−0.1†	−0.4
Extraneous (EL)		1	−0.5	−0.4
Germane (GL)			1	0.3
Post-test				1

†$p = 0.09$; all the other correlations are $p < 0.001$

Table 5: Correlations between the items targeting different cognitive-load components, by modality.

	Post-test, Audio	Post-test, Text	Post-test, Both
IL	−0.3, p=0.009	−0.4, p<0.001	−0.4, p=0.001
EL	−0.4, p=0.001	−0.2, p=0.05	−0.5, p<0.001
GL	0.5, p<0.001	0.04, p=0.7	0.5, p<0.001

6 DISCUSSION

6.1 We Did Not Observe a Modality Effect

Following Morrison [66], we explored a prediction (H1) derived from research on the modality effect: audio explanations will be superior to textual explanations with regard to post-test performance (H1a) and will result in less extraneous cognitive load (H1b). We did not find support for these hypotheses. The difference in scores between the Audio and Text groups was not statistically significant and the effect size was minimal. In other words, we, like Morrison, did not find evidence for the modality effect.

Again following Morrison [66], we explored the related prediction (H2) that the Both group will be outperformed on the post-test by the other two groups (H2a) and will experience higher extraneous load (H2b). Contrary to these predictions, however, the Both group scored descriptively higher than the Audio group on the post-test and experienced descriptively less cognitive load than the Audio group; neither of these differences is statistically significant and the effect sizes were minimal. In other words, we, like Morrison, did not find evidence for the verbal redundancy effect either.

Although our study is similar in many respects to Morrison's [66], the self-reported cognitive loads cannot be directly compared between the two studies due to a number of factors, such as a different learner populations, different topics and video material, and a minor difference in the rating scales.

6.2 Interpretations, Limitations, and Prospects

In this section, we comment on possible reasons for not having found a modality effect and outline opportunities for future work.

6.2.1 Design Considerations. Given how multiple design factors are known to impact on the modality effect and how little is known about the modality effect in the programming domain, many aspects of our materials may have influenced our results decisively.

One of the leading explanations for Morrison's [66] results was the intrinsic complexity of the content. We reduced complexity by focusing our videos on far fewer concepts and showing multiple shorter programs that introduced the concepts gradually. Given the beginner-level nature of our videos and the fairly low intrinsic-load scores reported by the participants, we expect that excessive complexity is not behind the lack of a modality effect in our study.

Another possible explanation is insufficient complexity: the modality effect cannot be expected on trivial content or materials with low element interactivity. Although our examples are absolutely trivial for experts or even intermediate learners, we do not believe insufficient complexity to be a likely explanation, since programming concepts are highly interconnected [60, 79], many beginners have trouble tracing even simple code [58, 88], there are many known misconceptions about variables and assignment [76, 93], and we observed a range of post-test scores among our participants. The floor effect on extraneous-load scores does merit consideration, and we would be happy to think that we have produced a programming tutorial with next to no extraneous load; the mix of post-test scores rather curtails our enthusiasm, however.

Excessive length of the explanatory segments of audio would explain why the Audio group did no better than Text. However, were that the case, the theory suggests that the Both group's performance should have particularly suffered, which did not happen.

Morrison [66] reported that natural-language issues may have played a part in her study. In our case, the narrator was not a native speaker but the vast majority of participants were. In the feedback we received, none of the participants complained about the clarity of speech, and the low extraneous-load scores suggest that language problems were negligible.

6.2.2 Validity of Assessments. One threat to validity comes from the way we assessed learning. Our post-test was created ad hoc for this study and may not measure programming knowledge well. We had no pre-test. (In hindsight, we should have included a pre-test. We did not because of our initial, wishful hope that it would be easy to recruit many true beginners via MTurk [37].)

The cognitive-load questionnaire that we used is more established, but its construct validity is open to question [53, 62, 105]. Learners may interpret the questions in various ways and may not be able to differentiate between intrinsic and extraneous load. Moreover, a post-hoc questionnaire cannot capture the moment-to-moment shifting of working-memory load during a preceding learning episode [42, 106].

6.2.3 Limitations of Crowdsourcing. As described under Methods and detailed further by Hellas et al. [37], we took a number of precautions to mitigate the concerns that arise from crowdsourcing educational data from the internet. Those precautions notwithstanding, our use of MTurk must be considered a caveat when interpreting our findings. For instance, there is uncertainty over how many of our paid but possibly disinterested participants honestly filled in the background survey (including the question on prior programming experience), attentively watched the video, and thoughtfully completed the CL questionnaire and the post-test. Classroom studies are not immune to such concerns either, of course.

In any case, crowdsourcing does compromise the ecological validity of our study, as our learners were not formally or even informally invested in learning to program before or after participation.[2] Ideally, modality studies in CER would include ones with authentic materials studied in an authentic context by authentic learners.

6.2.4 Motivation and Active Learning. Cognitive load theory tends to assume motivated learners and to view motivation as external to the theory [42, 53, 96, 105]. Load effects such as the modality effect are expected to occur when learners are sufficiently engaged with the materials; that may not have been the case in our study.

Plausible reasons for (possible) low motivation among our participants include the lack of a surrounding educational context, the passive viewing format of instruction with no active tracing practice, the decontextualized program fragments, the lack of incentives for successful learning, and the absence of a visible presenter and an actual programming environment on the video.

Unfortunately, we did not survey the participants directly about their motivation for the task and engagement with the materials. Future studies might improve on that.

Another avenue for future work is to explore the modality effect in more active programming pedagogies. For instance, the learning-from-examples and cognitive-load communities are increasingly interested in *desirable difficulties* and *productive failure* [18, 31, 53, 57]. CER might take a leaf out of their book. In concrete terms, that could mean, say, having learners attempt to make sense of a program *before* having it explained to them (cf. [83, 102]).

6.2.5 The Nature of Program Code. We must also look critically at the fundamental assumptions underlying modality research in CER, including our own. The modality effect is essentially about *pictures* that are accompanied by verbal explanations, which CER has interpreted as *programs* accompanied by verbal explanations [22, 66]. As noted by Morrison [66], however, it is not necessarily the case that code is "like a picture." Moreover, the answer to whether programs are more "like a picture" or "like language" is likely to be complex. Intuitively, Python code for instance has both text-like and diagram-like qualities. We know that visual layout affects code-reading efficiency [34], that expert programmers visually imagine program structures while designing [72], that the order in which programmers visually examine code depends complexly on expertise and the program [15, 70, 71], and that brain areas associated with the phonological loop are activated during program comprehension [87]. But we are still far from deeply understanding, for example, how people might employ the visuospatial sketchpad and the phonological loop as they study program code—and how the answer differs for beginners, intermediate learners, and experts.

A related theme that is attracting attention is the similarity and dissimilarity of programming notations (and other formalisms) to natural languages [2, 3, 10, 24, 28, 29, 48, 49, 74, 75, 87]. Research on this intriguing and complex topic has only begun.

All that being said, even if Python code is cognitively processed more "like language" than "like a picture," the theory suggests a verbal redundancy effect for learners who receive both spoken and written explanations of code [43]. However, our Both group did not do substantively worse than the other groups.

Based on research to date, it is impossible to say whether and how the modality effect applies to programming education at large. It is important to note that we, like Morrison [66], showed learners text-based programs; replications with block-based or visual programs would be interesting. Furthermore, there are other, more obviously pictorial representations in programming education that could be investigated for the modality effect; these include program animations, flowcharts, and UML diagrams, to name just a few.

6.2.6 Individual Differences. Learners' individual characteristics, such as differences in working-memory capacity and spatial skills, influence cognitive load and may impact on which forms of example-based learning are effective [5, 63, 84]. In this experiment, we did not explore these factors. It is possible that the modality effect occurred for some subset of our participants with certain individual characteristics, without us discovering it.

7 CONCLUSION

The modality effect is known to be a nuanced and complicated phenomenon that is sensitive to many moderating factors. In the programming domain, it remains elusive. In the future, we hope to see further replications with different study designs and demonstrably motivated cohorts of learners, comparisons of textual and auditory explanations of block-based and visual programs, modality studies of educational program visualizations, studies featuring advanced learners and content, and basic research on how people use working memory as they read code.

[2] Some of the feedback messages that we received make us hope that at least a few participants have since continued with programming.

ACKNOWLEDGMENTS

We thank Briana Morrison for her insightful advice and for kindly sharing materials from her study. We thank Rodrigo Duran for helpful discussions.

REFERENCES

[1] Eryn J. Adams, Anh T. Nguyen, and Nelson Cowan. 2018. Theories of working memory: Differences in definition, degree of modularity, role of attention, and purpose. *Language, Speech, and Hearing Services in Schools* 49, 3, 340–355.

[2] Marie Amalric and Stanislas Dehaene. 2016. Origins of the brain networks for advanced mathematics in expert mathematicians. *Proceedings of the National Academy of Sciences of the United States of America* 113, 18, 4909–4917.

[3] Ian Arawjo. 2020. To write code: The cultural fabrication of programming notation and practice. In *ACM CHI Conference on Human Factors in Computing Systems*.

[4] Viviane C. O. Aureliano, Patricia C. de A.R. Tedesco, and Michael E. Caspersen. 2016. Learning programming through stepwise self-explanations. In *Iberian Conference on Information Systems and Technologies (CISTI '16)*.

[5] Katherine Ann Austin. 2009. Multimedia learning: Cognitive individual diffences and display design techniques predict transfer learning with multimedia learning modules. *Computers & Education* 53, 4, 1339–1354.

[6] Mohammad Reza Azadmanesh and Matthias Hauswirth. 2017. Concept-driven generation of intuitive explanations of program execution for a visual tutor. In *IEEE Working Conference on Software Visualization*. IEEE, 64–73.

[7] Alan Baddeley. 1992. Working memory. *Science*.

[8] Christina Areizaga Barbieri and Julie L. Booth. 2020. Mistakes on display: Incorrect examples refine equation solving and algebraic feature knowledge. *Applied Cognitive Psychology*, 862–878.

[9] Maik Beege, Sascha Schneider, Steve Nebel, and Günter Daniel Rey. 2020. Does the effect of enthusiasm in a pedagogical agent's voice depend on mental load in the learner's working memory? *Computers in Human Behavior* 112.

[10] Marina Umaschi Bers. 2019. Coding as another language: a pedagogical approach for teaching computer science in early childhood. *Journal of Computers in Education*.

[11] Nicolas Biard, Salomé Cojean, and Eric Jamet. 2018. Effects of segmentation and pacing on procedural learning by video. *Computers in Human Behavior* 89, 411–417.

[12] Lori Breslow, David E. Pritchard, Jennifer DeBoer, Glenda S. Stump, Andrew D. Ho, and Daniel T. Seaton. 2013. Studying learning in the worldwide classroom: Research into edX's first MOOC. *Research and Practice in Assessment*.

[13] Steve Buchheit, Derek W. Dalton, Troy J. Pollard, and Shane R. Stinson. 2018. Crowdsourcing intelligent research participants: A student versus MTurk comparison. *Behavioral Research in Accounting* 31, 2, 93–106.

[14] Sabine Buchholz and Javier Latorre. 2011. Crowdsourcing preference tests, and how to detect cheating. In *Twelfth Annual Conference of the International Speech Communication Association (INTERSPEECH '11)*.

[15] Teresa Busjahn, Roman Bednarik, Andrew Begel, Martha Crosby, James H. Paterson, Carsten Schulte, Bonita Sharif, and Sascha Tamm. 2015. Eye movements in code reading: Relaxing the linear order. In *IEEE International Conference on Program Comprehension*. 255–265.

[16] Martin C. Carlisle. 2010. Using YouTube to enhance student class preparation in an introductory Java course. *SIGCSE'10*, 470–474.

[17] Charles H. Chen and Philip J. Guo. 2019. Improv: Teaching programming at scale via live coding. In *Learning at Scale (L@S '19)*. ACM.

[18] Ouhao Chen and Slava Kalyuga. 2019. Exploring factors influencing the effectiveness of explicit instruction first and problem-solving first approaches. *European Journal of Psychology of Education*.

[19] Nelson Cowan. 1998. *Attention and Memory: An Integrated Framework*. Oxford University Press.

[20] Suzanne L. Dazo, Nicholas R. Stepanek, Robert Fulkerson, and Brian Dorn. 2016. An empirical analysis of video viewing behaviors in flipped CS1 courses. In *Annual Conference on Innovation and Technology in Computer Science Education*. 106–111.

[21] Björn B. de Koning, Vincent Hoogerheide, and Jean Michel Boucheix. 2018. Developments and trends in learning with instructional video. *Computers in Human Behavior* 89, 395–398.

[22] Barbara Ericson, Steven Moore, Briana B. Morrison, and Mark Guzdial. 2015. Usability and usage of interactive features in an online ebook for CS teachers. In *WiPSCE '15*. ACM.

[23] Barbara J. Ericson, Lauren E. Margulieux, and Jochen Rick. 2017. Solving Parsons problems versus fixing and writing code. In *Proceedings of the 17th Koli Calling Conference on Computing Education Research (Koli Calling '17)*. ACM, 20–29.

[24] Evelina Fedorenko, Anna Ivanova, Riva Dhamala, and Marina Umaschi Bers. 2019. The language of programming: A cognitive perspective. *Trends in Cognitive Sciences*.

[25] Pedro Guillermo Feijóo-García and Christina Gardner-McCune. 2020. Using student-created instructional videos in CS upper-level courses: A successful strategy in a functional programming course. In *12th International Conference on Computer Supported Education*.

[26] James B. Fenwick Jr., Barry L. Kurtz, and Philip Meznar. 2013. Developing a highly interactive ebook for CS instruction. In *The 44th ACM Technical Symposium on Computer Science Education (SIGCSE '13)*. ACM, 135–140.

[27] Logan Fiorella and Richard E. Mayer. 2018. What works and doesn't work with instructional video. *Computers in Human Behavior* 89, 465–470.

[28] 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 *39th International Conference on Software Engineering*. IEEE/ACM.

[29] Stephen R. Foster and Lindsay D. Handley. 2020. *Don't Teach Coding (Before You Read This Book)*. John Wiley & Sons.

[30] Paul Ginns. 2005. Meta-analysis of the modality effect. *Learning and Instruction* 15, 4, 313–331.

[31] Inga Glogger-Frey, Corinna Fleischer, Lisa Grüny, Julian Kappich, and Alexander Renkl. 2015. Inventing a solution and studying a worked solution prepare differently for learning from direct instruction. *Learning and Instruction* 39, 72–87.

[32] Mitchell Gordon and Philip J. Guo. 2015. Codepourri: Creating visual coding tutorials using a volunteer crowd of learners. In *IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC '15)*. IEEE.

[33] Lassi Haaranen. 2017. Programming as a performance – Live-streaming and its implications for computer science education. In *ITiCSE '17*. ACM, 353–357.

[34] Michael Hansen, Robert L. Goldstone, and Andrew Lumsdaine. 2013. What makes code hard to understand? *arXiv e-prints*.

[35] Qiang Hao, David H. Smith IV, Naitra Iriumi, Michail Tsikerdekis, and Amy J. Ko. 2019. A systematic investigation of replications in computing education research. *ACM Transactions on Computing Education* 4, 42.

[36] Kyle J. Harms, Jason Chen, and Caitlin Kelleher. 2016. Distractors in Parsons problems decrease learning efficiency for young novice programmers. In *The 12th International Computing Education Research Conference*. ACM, 241–250.

[37] Arto Hellas, Albina Zavgorodniaia, and Juha Sorva. 2020. Crowdsourcing in Computing Education Research: Case Amazon MTurk. In *Proceedings of the 20th Koli Calling International Conference on Computing Education Research (Koli Calling '20)*. ACM.

[38] Panagiotis G Ipeirotis, Foster Provost, and Jing Wang. 2010. Quality management on Amazon Mechanical Turk. In *Proceedings of the ACM SIGKDD workshop on human computation*. 64–67.

[39] Allison J. Jaeger, Joanna A. Marzano, and Thomas F. Shipley. 2020. When seeing what's wrong makes you right: The effect of erroneous examples on 3D diagram learning. *Applied Cognitive Psychology* 34, 844–861.

[40] Dayu Jiang and Slava Kalyuga. 2020. Confirmatory factor analysis of cognitive load ratings supports a two-factor model. 16, 3, 216–225.

[41] Dan R. Johnson and Lauren A. Borden. 2012. Participants at your fingertips. *Teaching of Psychology* 39, 4, 245–251.

[42] Slava Kalyuga. 2011. Cognitive load theory: How many types of load does it really need? *Educational Psychology Review* 23, 1–19.

[43] Slava Kalyuga. 2011. Instructional benefits of spoken words: A review of cognitive load factors. *Educational Research Review* 7, 145–159.

[44] Slava Kalyuga, Paul Chandler, and John Sweller. 1999. Managing split-attention and redundancy in multimedia instruction. *Applied Cogn. Psychology*, 351–371.

[45] Slava Kalyuga and Anne-Marie Singh. 2016. Rethinking the boundaries of cognitive load theory in complex learning. *Educational Psychology Review* 28, 831–852.

[46] Kandarp Khandwala and Philip J. Guo. 2018. Codemotion: Expanding the design space of learner interactions with computer programming tutorial videos. In *The Fifth Annual ACM Conference on Learning at Scale (L@S 2018)*. ACM.

[47] Andreas Korbach, Roland Brünken, and Babette Park. 2018. Differentiating different types of cognitive load: A comparison of different measures. *Educational Psychology Review* 30, 503–529.

[48] Ryan Krueger, Tyler Santander, Westley Weimer, and Kevin Leach. 2020. Neurological divide: An fMRI study of prose and code writing. In *42nd International Conference on Software Engineering*. ACM.

[49] David Landy and Robert L. Goldstone. 2007. Formal notations are diagrams: Evidence from a production task. *Memory & Cognition* 35, 8, 2033–2040.

[50] Wayne Leahy and John Sweller. 2016. Cognitive load theory and the effects of transient information on the modality effect. *Instructional Science* 44, 1, 107–123.

[51] Michael J. Lee and Amy J. Ko. 2015. Comparing the Effectiveness of Online Learning Approaches on CS1 Learning Outcomes. In *The Eleventh annual International Conference on International Computing Education Research*.

[52] J. Leek, R. D. Peng, and B. Caffo. 2020. Why automated videos? https://www.coursera.org/lecture/data-scientists-tools/why-automated-videos-enUSz

[53] Jimmie Leppink. 2020. Revisiting cognitive load theory: Second thoughts and unaddressed questions. *Scientia Medica* 30, 1–8.

[54] Jimmie Leppink, Fred Paas, Cees P. M. Van der Vleuten, Tamara Van Gog, and Jeroen J. G. Van Merriënboer. 2013. Development of an instrument for measuring different types of cognitive load. *Behavior research methods* 45, 1058–72.

[55] Jimmie Leppink, Fred Paas, Tamara van Gog, Cees P. M. van der Vleuten, and Jeroen J. G. van Merriënboer. 2014. Effects of pairs of problems and examples on task performance and different types of cognitive load. *Learning and Instruction* 30, 32–42.

[56] Tze Wei Liew, Su-Mae Tan, Teck Ming Tan, and Si Na Kew. 2020. Does speaker's voice enthusiasm affect social cue, cognitive load and transfer in multimedia learning? *Information and Learning Sciences* 121, 3/4.

[57] Vicki Likourezos and Slava Kalyuga. 2017. Instruction-first and problem-solving-first approaches: alternative pathways to learning complex tasks. *Instructional Science* 45, 2, 195–219.

[58] Raymond Lister. 2016. Toward a developmental epistemology of computer programming. In *Proceedings of the 11th Workshop in Primary and Secondary Computing Education (WiPSCE '16)*. ACM, 5–16.

[59] Lauren E. Margulieux and Richard Catrambone. 2017. Using learners' self-explanations of subgoals to guide initial problem solving in App Inventor. In *The 2017 ACM Conference on International Computing Education Research*. 21–29.

[60] Lauren E. Margulieux, Richard Catrambone, and Laura M. Schaeffer. 2018. Varying effects of subgoal labeled expository text in programming, chemistry, and statistics. *Instructional Science* 46, 707–722.

[61] Samiha Marwan, Joseph Jay Williams, and Thomas Price. 2019. An Evaluation of the Impact of Automated Programming Hints on Performance and Learning. In *The 2019 ACM Conference on International Computing Education Research*.

[62] Myrto F. Mavilidi and Lijia Zhong. 2019. Exploring the development and research focus of cognitive load theory, as described by its founders: Interviewing John Sweller, Fred Paas, and Jeroen van Merriënboer. *Educational Psychology Review* 31, 499–508.

[63] Richard E. Mayer (Ed.). 2014. *The Cambridge Handbook of Multimedia Learning* (2nd ed.). Cambridge University Press.

[64] Richard E. Mayer. 2020. *Multimedia Learning* (3rd ed.). Cambridge University Press.

[65] Briana B. Morrison. 2016. *Replicating Experiments from Educational Psychology to Develop Insights into Computing Education: Cognitive load as a Significant Problem in Learning Programming*. Doctoral dissertation. Georgia Institute of Technology.

[66] Briana B. Morrison. 2017. Dual modality code explanations for novices: Unexpected results. In *The 2017 ACM Conference on International Computing Education Research (ICER '17)*. ACM, 226–235.

[67] Briana B. Morrison, Brian Dorn, and Mark Guzdial. 2014. Measuring cognitive load in introductory CS: Adaptation of an instrument. In *The Tenth Annual Conference on International Computing Education Research (ICER '14)*. 131–138.

[68] Seyed Yaghoub Mousavi, Renae Low, and John Sweller. 1995. Reducing cognitive load by mixing auditory and visual presentation modes. *Journal of Educational Psychology* 87, 2, 319–334.

[69] Gabriele Paolacci and Jesse Chandler. 2014. Inside the Turk. *Current Directions in Psychological Science* 23, 3, 184–188.

[70] Patrick Peachock, Nicholas Iovino, and Bonita Sharif. 2017. Investigating Eye Movements in Natural Language and C++ Source Code - A Replication Experiment. In *11th International Conference on Augmented Cognition*. 206–218.

[71] Norman Peitek, Janet Siegmund, and Sven Apel. 2020. What drives the reading order of programmers? An eye tracking study. In *ICPC*.

[72] Marian Petre and Alan F. Blackwell. 1999. Mental imagery in program design and visual programming. *Int. J. of Human Computer Studies* 51, 1, 7–30.

[73] Elizabeth Poché, Nishant Jha, Grant Williams, Jazmine Staten, Miles Vesper, and Anas Mahmoud. 2017. Analyzing user comments on YouTube coding tutorial videos. In *25th International Conference on Program Comprehension (ICPC '17)*. IEEE, 196–206.

[74] Scott R. Portnoff. 2018. The introductory computer programming course is first and foremost a LANGUAGE course. *ACM Inroads* 9, 2, 34–52.

[75] 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. *Nature Scientific Reports* 10, 1, 3817.

[76] Yizhou Qian and James Lehman. 2017. Students' misconceptions and other difficulties in introductory programming: A literature review. *ACM Transactions on Computing Education* 18, 1, 1–24.

[77] Günter Daniel Rey, Maik Beege, Steve Nebel, Maria Wirzberger, Tobias H. Schmitt, and Sascha Schneider. 2019. A meta-analysis of the segmenting effect. *Educational Psychology Review* 31, 389–419.

[78] Juliane Richter, Katharina Scheiter, and Alexander Eitel. 2016. Signaling text-picture relations in multimedia learning: A comprehensive meta-analysis. *Educational Research Review* 17, 19–36.

[79] Anthony Robins. 2010. Learning edge momentum: A new account of outcomes in CS1. *Computer Science Education* 20, 1, 37–71.

[80] Beth A. Rogowsky, Barbara M. Calhoun, and Paula Tallal. 2016. Does modality matter? The effects of reading, listening, and dual modality on comprehension. *SAGE Open* 6, 3.

[81] Ralf Rummer, Judith Schweppe, Anne Fürstenberg, Tina Seufert, and Roland Brünken. 2008. Working memory interference during processing text and pictures: Implications for the explanation of the modality effect. *Applied Cognitive Psychology* 22, 1–13.

[82] Sascha Schneider, Maik Beege, Steve Nebel, and Günter Daniel Rey. 2018. A meta-analysis of how signaling affects learning with media. *Educational Research Review* 23, 1–24.

[83] Anne Schüler and Maria Gabriela Mayer. 2020. Illustrations before text reduce visuospatial working memory load during text processing. *Discourse Processes*.

[84] Matthias Schwaighofer, Markus Bühner, and Frank Fischer. 2016. Executive functions as moderators of the worked example effect: When shifting is more important than working memory capacity. *Journal of Educational Psychology* 108, 7, 982–1000.

[85] Stoo Sepp, Steven J. Howard, Sharon Tindall-Ford, Shirley Agostinho, and Fred Paas. 2019. Cognitive load theory and human movement: Towards an integrated model of working memory. *Educational Psychology Review* 31, 293–317.

[86] Jason H. Sharp and Leah A. Schultz. 2013. An exploratory study of the use of video as an instructional tool in an introductory C# programming course. *information Systems Education Journal*.

[87] 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 *International Conference on Software Engineering (ICSE '14)*. 378–389.

[88] Simon. 2011. Assignment and sequence: Why some students can't recognize a simple swap. In *The 11th Koli Calling International Conference on Computing Education Research*, Ari Korhonen and Robert McCartney (Eds.). ACM, 16–22.

[89] Beth Simon, Sue Fitzgerald, Renée McCauley, Susan Haller, John Hamer, Brian Hanks, Michael T. Helmick, Jan Erik Moström, Judy Sheard, and Lynda Thomas. 2007. Debugging assistance for novices: A video repository. *ITiCSE Working Group Reports*, 137–151.

[90] Teemu Sirkiä and Juha Sorva. 2015. Tailoring animations of example programs. In *The 15th Koli Calling Conference on Computing Education Research*. 147–151.

[91] Glenn Smith and Colin Fidge. 2008. On the efficacy of prerecorded lectures for teaching introductory programming. In *10th Australasian Computing Education Conference (ACE '08)*. 129–136.

[92] Adalbert Gerald Soosai Raj, Pan Gu, Eda Zhang, Arokia Xavier Annie R., Jim Williams, Richard Halverson, and Jignesh M. Patel. 2020. Live-coding vs static code examples: Which is better with respect to student learning and cognitive load?. In *The 22nd Australasian Computing Education Conference*. 152–159.

[93] Juha Sorva. 2018. Misconceptions and the beginner programmer. In *Computer Science Education: Perspectives on Teaching and Learning in School*, Sue Sentance, Erik Barendsen, and Carsten Schulte (Eds.). Bloomsbury, 171–187.

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

[95] Klaus D. Stiller, Annika Freitag, Peter Zinnbauer, and Christian Freitag. 2009. How pacing of multimedia instructions can influence modality effects: A case of superiority of visual texts. *Australasian Journal of Educ. Technology* 25, 2.

[96] John Sweller. 2010. Element interactivity and intrinsic, extraneous, and germane cognitive load. *Educational Psychology Review* 22, 123–138.

[97] John Sweller, Jeroen J. G. van Merriënboer, and Fred Paas. 1998. Cognitive architecture and instructional design. *Educ. Psychology Review* 10, 251–296.

[98] John Sweller, Jeroen J. G. van Merriënboer, and Fred Paas. 2019. Cognitive architecture and instructional design: 20 years later. *Educational Psychology Review*.

[99] Laura Toma and Jan Vahrenhold. 2018. Self-efficacy, cognitive load, and emotional reactions in collaborative algorithms labs — A case study. In *The 2018 ACM Conference on International Computing Education Research (ICER '18)*. ACM.

[100] Camilo Vieira, Alejandra J. Magana, Michael L. Falk, and R. Edwin Garcia. 2017. Writing in-code comments to self-explain in computational science and engineering education. *ACM Transactions on Computing Education* 17, 4, 1–21.

[101] Tamar Vilner, Ela Zur, and Ronit Sagi. 2012. Integrating video components in CS1. In *The 43rd ACM Technical Symposium on Computer Science Education (SIGCSE '12)*. ACM, 123–128.

[102] Peng Wang, Roman Bednarik, and Andrés Moreno. 2012. During automatic program animation, explanations after animations have greater impact than before animations. In *Proceedings of the 12th Koli Calling International Conference on Computing Education Research (Koli Calling '12)*. ACM, 100–108.

[103] Jonathan H. Williams. 1998. Guidelines for the use of multimedia in instruction. In *The Human Factors and Ergonomics Society Annual Meeting*. 1447–1451.

[104] Zhen Xu, Albert D. Ritzhaupt, Karthikeyan Umapathy, Yang Ning, and Chin Chung Tsai. 2020. Exploring college students' conceptions of learning computer science: A draw-a-picture technique study. *Comput. Sci. Educ.*

[105] Albina Zavgorodniaia, Rodrigo Duran, Arto Hellas, Otto Seppälä, and Juha Sorva. 2020. Measuring the cognitive load of learning to program: A replication study. In *United Kingdom & Ireland Computing Education Research conference (UKICER '20)*. ACM, 3–9.

[106] Robert Z. Zheng (Ed.). 2017. *Cognitive Load Measurement and Application*. Routledge.

[107] Rui Zhi. 2019. *Design and Evaluation of Instructional Supports for Novice Programming Environments*. Doctoral dissertation. North Carolina State University.