Computers in Human Behavior 109 (2020) 106349

Contents lists available at ScienceDirect

Computers in Human Behavior

journal homepage: http://www.elsevier.com/locate/comphumbeh

Full length article

A meta-analysis of teaching and learning computer programming: Effective instructional approaches and conditions

Ronny Scherer [a,*], Fazilat Siddiq [b], Bárbara Sánchez Viveros [c]

[a] Centre for Educational Measurement at the University of Oslo (CEMO), Faculty of Educational Sciences, University of Oslo, Norway
[b] Department of Education and Quality in Learning, University of South-Eastern Norway(USEN), Norway
[c] Faculty of Life Sciences, Humboldt-Universität zu Berlin, Germany

ARTICLE INFO

Keywords:
Computational thinking
Computer programming
Intervention studies
Multilevel meta-analysis
Scratch programming

ABSTRACT

This meta-analysis maps the evidence on the effectiveness of instructional approaches and conditions for learning computer programming under three study conditions: (a) Studies focusing on the effectiveness of programming interventions per se, (b) studies focusing on the effectiveness of *visualization* and *physicality*, and (c) studies focusing on the effectiveness of dominant *instructional approaches*. Utilizing the data from 139 interventions and 375 effect sizes, we found (a) a strong effect of learning computer programming per se (Hedges' $\overline{g} = 0.81$, 95% CI [0.42, 1.21]), (b) moderate to large effect sizes of visualization ($\overline{g} = 0.44$, 95% CI [0.29, 0.58]) and physicality interventions ($\overline{g} = 0.72$, 95% CI [0.23, 1.21]), and (c) moderate to large effect sizes for studies focusing on dominant instructional approaches ($\overline{g}$s = 0.49–1.02). Moderator analyses indicated that the effect sizes differed only marginally between the instructional approaches and conditions—however, collaboration in metacognition instruction, problem solving instruction outside of regular lessons, short-term interventions focusing on physicality, and interventions focusing on visualization through Scratch were especially effective. Our meta-analysis synthesizes the existing research evidence on the effectiveness of computer programming instruction and, ultimately, provides references with which the effects of future studies could be compared.

1. Introduction

Computer programming has regained considerable attention over the last decade, not only because of the rapid technological developments but also because it is claimed to foster other skills, including problem solving, logical thinking, and creativity (Liao & Bright, 1991; Scherer, 2016). Moreover, educational systems around the world are in the process of developing curricula that implement programming and so-called computational thinking—a concept that contextualizes computer programming and related skills as a form of problem solving (Shute, Sun, & Asbell-Clark, 2017)—either as a standalone subject or integrated in other subjects (European Commission, 2016; Yadav, Good, Voogt, & Fisser, 2017). Whereas the importance of computer programming has been widely recognized, the systematic evaluation of the effectiveness of instructional approaches and conditions fostering the acquisition of programming knowledge and skills has received little attention (Grover & Pea, 2013; Lye & Koh, 2014).

Besides, the existing body of literature abounds in diverse instructional approaches, focusing on the use of specific programming tools (Flórez et al., 2017), ways to facilitate the understanding of computational concepts and the acquisition of information processing along with metacognitive skills (Lye & Koh, 2014), the benefits of pair programming over individual programming (Umapathy & Ritzhaupt, 2017), and the setup of programming courses, including the effects of blended and project-based learning (Hsu, Chang, & Hung, 2018; Vihavainen, Airaksinen, & Watson, 2014)—just to name a few. These different foci have inevitably led to diverse findings concerning the effectiveness of certain instructional approaches and conditions. For instance, whereas Lou, Abrami, and d'Apollonia (2001) found weak effects of collaborative learning with technology, including computer programming, on individual and group performance (Cohen's $d = 0.15$–0.31), Umapathy and Ritzhaupt (2017) identified moderate to strong effects (Hedges' $\overline{g} = 0.41$–0.64). Moreover, whereas Yüksel and Yüksel (2015) obtained strong effects of teaching programming through problem solving ($g > 1.00$), Denny, Cukierman, and Bhaskar (2015) testified to only small effects ($g = 0.27$). The list of studies and diverse findings could be

* Corresponding author. Faculty of Educational Sciences, Centre for Educational Measurement at the University of Oslo (CEMO), Postbox 1161 Blindern, NO-0318, Oslo, Norway.
E-mail address: ronny.scherer@cemo.uio.no (R. Scherer).

https://doi.org/10.1016/j.chb.2020.106349
Received 9 October 2019; Received in revised form 20 January 2020; Accepted 15 March 2020
Available online 27 March 2020

extended easily—overall, these examples suggest that the effectiveness of programming instruction varies considerably across studies.

With more computer science educators interested in making programming accessible to young students, learning programming through game design, robotics, and with visual instead of text-based languages is expected to be more effective than other approaches (e.g., Batista, Connolly, & Angotti, 2016; Lee, Mauriello, Ahn, & Bederson, 2014; Lye & Koh, 2014). However, the existing body of research has not yet provided sufficient evidence supporting these expectations (Flórez et al., 2017; Scherer, 2016). So, what are effective approaches and conditions for teaching and learning computer programming? This meta-analysis is aimed at providing some answers to this question by synthesizing the evidence from experimental and quasi-experimental studies targeted at improving students' programming knowledge and skills. Specifically, using the framework for reviewing the effectiveness of educational technology proposed by Chen, Wang, Kirschner, and Tsai (2018), we distinguish between three categories of primary studies to examine three aspects of effectiveness and ultimately map the field of programming instruction (Fig. 1): (a) Studies that reported the effectiveness of learning computer programming per se (i.e., with control groups that did not engage in any programming activity), (b) Studies that reported the effectiveness of *visualization and physicality* during programming (e.g., visual programming languages such as Scratch, involvement of robotics), and (c) Studies that reported the effectiveness of dominant *instructional approaches* (e.g., programming instruction focusing on metacognition, game-based learning, collaboration, feedback). For these three categories, we estimate the overall intervention effect sizes on performance-based outcome variables—that is, measures of programming knowledge and skills—through multiple, separate meta-analyses and quantify the variation of effects within and across studies. Further moderator analyses are conducted to explain this variation by contextual variables. Overall, our research synthesis provides information about whether instructional approaches and conditions have fulfilled the expectations associated with their effectiveness for learning computer programming.

1.1. Anchoring computer programming in the concept of computational thinking

Computer programming is defined as the "process of developing and implementing various sets of instructions to enable a computer to perform a certain task, solve problems, and provide human interactivity" (Balanskat & Engelhardt, 2015, p. 7). Thus, in addition to having knowledge of programming languages, expertise in subjects related to the development of specialized algorithms and logic, and the ability to analyze, understand, and solve problems in an iterative process are required (Forsström & Kaufmann, 2018). The processes involved in programming are therefore largely similar to those involved in problem-solving, such as decomposing problems, applying algorithms, abstracting, and automatizing (Shute, Sun, & Asbell-Clarke, 2017; Yadav et al., 2017).

In their seminal review, Lye and Koh (2014) argued that computer programming "exposes students to computational thinking which involves problem-solving using computer science concepts like abstraction and decomposition." (p. 51). Ultimately, the authors concluded that fostering the skills involved in programming will also enhance the skills involved in computational thinking. Despite its criticism (Denning, 2017), the concept of computational thinking has found its way in existing computer science curricula, teacher education programs, and research agendas (Grover & Pea, 2013). Wing (2006) broadly defined computational thinking as a concept that "involves solving problems, designing systems, and understanding human behavior, by drawing on the concepts fundamental to computer science" (p. 33). Drawing on this definition and subsequent specifications of the very concepts that are 'fundamental to computer science", Shute et al. (2017) named the key processes involved in computational thinking—problem (re-)formulation, recursion, decomposition, abstraction, and systematic testing of solutions and procedures. In light of these processes, the authors argued that computational thinking can be considered a form of problem solving in technology-rich contexts.

Although the processes involved and the skills required in computer programming are those involved and required in computational thinking (Lye & Koh, 2014), the latter involves more than programming. In their influential framework, Brennan and Resnick (2012) outlined three key areas of computational thinking: Computational concepts (i.e., concepts used by programmers, such as sequences and loops), computational practices (i.e., problem-solving processes during programming, such as testing and debugging), and computational perspectives (i.e., students' understanding of themselves and their interaction with others and with technology, such as questioning technology as a means to solve real-life problems). Whereas computational concepts and practices play a critical role in programming, the latter—taking computational perspectives as a way to computational participation—represents a distinguishing feature of computational thinking (Kafai & Burke, 2013; Shute et al., 2017). Programming is considered a way of teaching and learning computational thinking—in other words, learning to program a computer can ultimately aid the acquisition of computational thinking skills (Flórez et al., 2017).

Given the limited focus of intervention studies on computational perspectives (Lye & Koh, 2014), the current series of meta-analysis

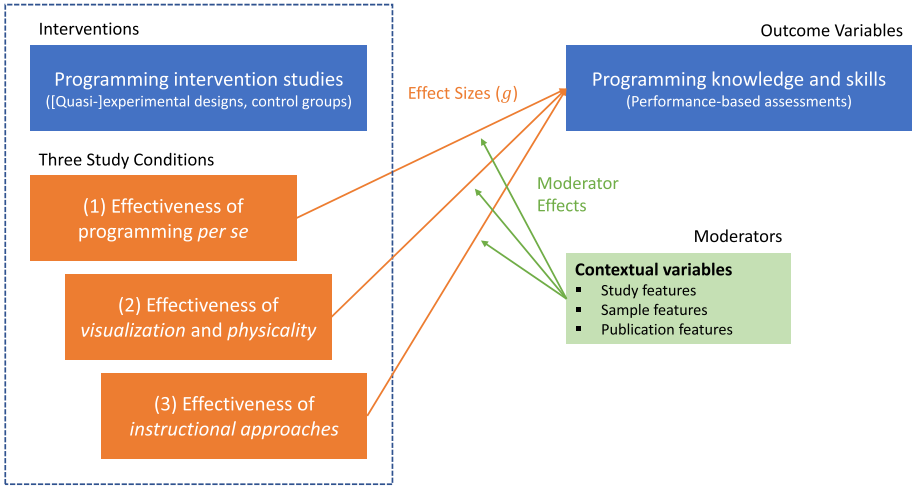


Fig. 1. Conceptual framework of the present meta-analysis.

focuses on the computational concepts and practices, labelled as programming knowledge and skills. Programming knowledge, in this respect, comprises the conceptual and procedural knowledge needed to solve problems computationally (i.e., syntactic, semantic, schematic, and strategic knowledge). Programming skills comprise the skills to create, modify, and evaluate computer code.

1.2. Approaches and conditions of computer programming instruction

In their recent article, Brown and Wilson (2018) reviewed the role of computer programming for computational biology and concluded that, in light of the extant literature on programming instruction, "competence at programming is not innate but is rather a learned skill that can be acquired and improved with practice" (p. 1). Based on this assumption that programming knowledge and skills can be taught effectively, several instructional approaches have been proposed and evaluated over the last decades—yet, with varying foci and degrees of success (Grover & Pea, 2013; Robins, Rountree, & Rountree, 2003).

In the early studies from the 1980s and 1990s, programming instruction with the Logo language was in the main focus. After a myriad of experimental and quasi-experimental studies had been conducted, the evidence base on the effectiveness of different instructional approaches was diverse. Whereas some studies found teacher-directed instruction to be more effective than discovery learning (Lee, 1991), others found the opposite effect (see Clements, 1995 for an overview). Palumbo (1990) consequently argued that key study design features, such as the type of programming language and the length of the intervention, should be considered to explain these varying effects. The context and tool dependence of effective programming instruction seems evident.

Reviewing the existing literature on K-12 computing education for the newer studies, Garneli, Giannakos, and Chorianopoulos (2015) highlighted several focus areas intervention studies have engaged in—these areas included examining the importance of programming tools, educational contexts, and instructional methods. The authors also emphasized the growing popularity of game design and robotics instruction, project-based interventions, and interventions that involve collaboration and the use of physical objects to determine the outcome of certain programming tasks. Garneli et al. (2015) concluded that implementing computing education in K-12 instruction can be "enjoyable and effective"—however, empirical evidence supporting these expectations is still scarce (Grover & Pea, 2013; Scherer, 2016). Lye and Koh (2014) consequently called for exploring more classroom interventions of computer programming to enrich the existing knowledge base of 'what works and what doesn't'. Reviewing the effectiveness of teaching introductory programming for course pass rates, Vihavainen et al. (2014) identified core intervention programs. These programs included collaboration and peer support, relatable content and contextualization, assessment procedures, course setup, and resourcing. The authors synthesized the effect sizes resulting from intervention studies that focused on at least one of these programs and found an overall positive effect suggesting that pass rates could be improved up to 40% compared to traditional lecture- and lab-based courses. At the same time, Vihavainen et al. (2014) acknowledged that these improvements vary across instructional approaches and that a combination of multiple approaches may be most effective for teaching programming. Flórez et al. (2017) concurred with this conclusion and further pointed out the importance of collaboration and peer support as well as the use of visualization tools to help students develop and explicate their mental models about programming concepts. Next to these trends in intervention studies to foster the teaching and learning of computer programming, several other programs exist, which focus, for instance, on the benefits of blended learning over face-to-face learning, the effectiveness of problem-solving instruction, feedback, and the fostering of metacognitive skills (for an overview and example studies, please refer to Table 2).

Overall, our review of the extant literature revealed that (a) diverse instructional approaches to fostering computer programming exist; (b) several intervention programs are effective in fostering programming knowledge and skills; (c) the effectiveness of intervention programs may vary across studies and instructional conditions (see also Li & Ma, 2010). In fact, existing studies indicated that the effectiveness of programming interventions depend on the context they are placed in. Kafai and Burke (2015), for example, noted the relevance of the intervention length that may range between some hours and several months and the integration of the intervention in short-term coding camps, extracurricular activities, or regular school lessons. Despite this diversity, some core programs seem to reoccur, such as the effectiveness of certain programming tools and collaboration (Hsu et al., 2018).

Visual programming tools. A considerable number of studies focused on the effectiveness of certain programming tools over alternative tools. For instance, Lee (1990), in an early meta-analysis, found that programming with the Logo software was significantly more effective than with the Basic software. Later on, Au (1992) confirmed this finding using problem-solving transfer tests as outcome measures. In the same study, the ways in which the Logo programming instruction was integrated (process-vs. content-oriented) moderated the overall effect size. Similarly, some evidence from the early studies exists that the superior effectiveness of the programming language Logo over Pascal and BASIC depended on the instructional approach (see also Lee, 1991). Liao and Bright (1991) summarized the primary programming studies and confirmed that some programming languages are more effective in fostering the transfer of programming skills than others—an observation that was also made for modern languages. Specifically, Costa and Miranda (2017) meta-analyzed intervention studies of the effectiveness of learning programming with the language Alice. The authors identified six eligible studies and found an overall, positive, and moderate effect on programming performance, $\overline{d} = 0.54$, 95% CI [0.34, 0.74]. Costa and Miranda concluded that Alice is an effective software to learn programming, yet they could not explain the variation of the intervention effect across studies. Furthermore, Moreno-León and Robles (2016) reviewed studies that used the visual programming language Scratch mainly in the contexts of game design and storytelling. The authors found support for the overall effectiveness of teaching with Scratch for improving students' attitudes toward programming and their programming performance; however, given the limited number of actual (quasi-) experimental studies, these effects could not be synthesized meta-analytically.

One of the main reasons for the hypothesized superiority of some programming languages over others lies in their visual nature which may make programming more accessible to students than text-based languages and thus more effective (Grover & Pea, 2013). In fact, some evidence suggests that additional visualizations, such as concept maps, may elevate these effects (Flórez et al., 2017). Especially with the development of the Scratch and Logo languages, computer science educators are hoping to teach students programming already in primary school and kindergarten. As early as the 1990s, customizing programming tools and languages for certain age groups of students, especially for younger students, is considered an integral part of developing programming instruction (Clements & Sarama, 1997).

Overall, what this brief review of the extant literature indicates is that programming interventions may be differentially effective across different programming languages, favoring visualization-based instruction and visual languages.

Collaboration. Another, substantial set of intervention studies focused on the effects of learning programming collaboratively, for instance, by pair programming. Lou et al. (2001) meta-analyzed the overall effects of learning with technology collaboratively in comparison to individual learning. The authors identified weak yet significant and positive effects on individual and group performance ($\overline{d} = 0.15$–0.31). Later, Umapathy and Ritzhaupt (2017) reviewed 28 effect sizes reported in 18 primary studies and found moderate to strong effects of pair

programming on performance in programming exams and assignments ($\bar{g} = 0.41–0.64$)—these effects varied significantly across studies. Brown and Wilson (2018) consequently encouraged lecturers of computer programming to consider collaboration a key element in their instruction. Peer support and collaborative problem solving seem to be especially effective in stimulating computational thinking as they allow students to resolve immediate inquiries more rapidly than working individually (Flórez et al., 2017). Although this evidence base largely supports the effectiveness of collaborative practices, some evidence suggests that the effects may between domains, gender, and the composition of the samples (Springer, Stanne, & Donovan, 1999).

Game design and physicality. Trying to make programming more accessible to younger students, researchers and computer science educators have contextualized programming instruction in the design of games and the use of robots (Lee et al., 2014). Behind this contextualization lies the expectation that both game design and robotics will not only facilitate the understanding of computational concepts more than alternative approaches but will engage students more effectively in collaboration (Batista et al., 2016). In Lee's (1990) early review, simulation- and game-based interventions were indeed most beneficial to higher-grade students' learning of computer programming. Concerning the interventions involving robotics (e.g., Lego Mindstorms®), Lito (2017) meta-analyzed the available effect sizes and found a strong positive and statistically significant effect size, $\bar{d} = 0.70$, 95% CI [0.28, 1.11], $k = 12$. One may argue that both designing games and programming robots are especially effective for teaching and learning programming, because they shift the focus from creating the code to the applications and the 'making' of creative products (Kafai & Burke, 2013). Moreover, the code students develop can be tested directly, and immanent feedback is accessible by observing, for instance, the movements of a programmed robot. Liu, Schunn, Flot, and Shoop (2013) supported the argument for involving physicality in programming interventions and provided some empirical evidence that physical programming environments may impact positively students' algorithmic thinking.

Creating games through programming may not only increase students' motivation to engage in programming and acquire the required technical skills but also create opportunities for collaborative learning experiences (Kafai & Burke, 2015). These approaches, however, still have to deliver on their promises by providing a sufficiently large body of evidence for their effectiveness (Flórez et al., 2017). The present meta-analysis examines some aspects of this evidence base.

1.3. Framework for the present meta-analysis

To synthesize the research evidence on the effectiveness of instructional approaches and conditions for the learning of computer programming, we drew from Chen et al. (2018) framework of three study conditions: (a) the effectiveness of technology interventions per se, (b) the effectiveness of features of the learning environments or tools, and (c) the effectiveness of instructional approaches. This framework was informed by Mayer's (2015) taxonomy of organizing the research evidence surrounding digital game-based learning and was also adopted in a recent meta-analysis by Tsai and Tsai (2018). In essence, it represents a way of categorizing primary studies into three conditions in order to shed light on the effectiveness of technology-based interventions from multiple perspectives rather than from a single perspective. Chen et al. (2018) consider this multi-perspective approach to be especially useful for organizing and mapping domains and study contexts with a variety of research foci and approaches. At the same time, this framework faces two challenges: First, given the different study conditions, separate meta-analyses must be performed to synthesize the evidence within each condition—this may, however, limit the number of studies available and ultimately reduce the power to detect small effect sizes. Second, the first condition (1) focuses on the effects of technology-based interventions per se. While these effects may not have specific and direct implications for instruction, they provide references against which the effects derived from conditions (2) and (3) could be compared.

Transferring this framework to the context of computer programming instruction, we distinguish between three study conditions: (1) Studies that reported the effects of *programming instruction* per se, which allowed us to compare the effects of programming instruction with instruction outside the programming domain; (2) Studies that reported the effects of *visualization and physicality;* (3) Studies that reported the effects of *instructional approaches*. Fig. 1 depicts these three conditions and the overall framework of this meta-analysis, and Table 1 clarifies the study designs underlying these conditions, which will be discussed in more detail in the method section of this paper.

1.4. The present meta-analysis

The present meta-analysis synthesized the evidence surrounding the effectiveness of instructional approaches and conditions for learning computer programming and tested some of the claims surrounding the effectiveness of certain instructional conditions. The main contribution of this study consequently lies in generating knowledge about what may or may not work well in computer programming instruction and whether new programming tools and ways of instruction can deliver on their promises. We synthesized the evidence within the three conditions (Fig. 1), addressing the following three research questions (RQs):

RQ1. To what extent are computer programming interventions effective in fostering students' programming knowledge and skills? (*Effectiveness of programming interventions* per se).

RQ2. (a) To what extent are interventions focusing on visualization

Table 1
Overview of the study conditions (a) to (c).

Study condition	Experimental group	Control group	Examples
(a) Effectiveness of programming interventions per se	Instruction with computer programming	Instruction without any computer programming	• Learning mathematics with Logo vs. learning mathematics without programming • Problem-solving instruction with programming vs. without programming
(b) Effectiveness of *visualization* or *physicality*	Instruction with visual (programming) tools or physical implementations of code (e.g., through robots)	Instruction without the visual (programming) tools or physical implementations of code (e.g., through robots)	• Programming instruction with Java vs. programming instruction with Scratch • Programming with a text-based language and visualizations vs. programming with only the text-based language • Programming instruction with vs. without robotics (e.g., Lego Mindstorms®)
(c) Effectiveness of *instructional approaches*	Programming instruction with specific instructional approaches	Programming with conventional instruction	• Pair programming vs. individual programming • Programming instruction with Logo and metacognitive reflections vs. programming instruction with Logo without metacognitive reflections

effective in fostering students' programming knowledge and skills? (b) To what extent are interventions focusing on physicality effective in fostering students' programming knowledge and skills? (*Effectiveness of visualization and physicality*).

RQ3. To what extent are the following instructional approaches to teaching computer programming effective in fostering students' programming knowledge and skills: (a) Blended learning, (b) Collaboration, (c) Feedback, (d) Game-based learning, (e) Metacognition, and (f) Problem solving? (*Effectiveness of instructional approaches*).

Besides synthesizing the effect sizes within these conditions, we also quantify their variation within and between studies and examine which study, sample, and publication features may explain this variation (*Moderator analyses*). To our best knowledge, this meta-analysis is the first to quantify the effectiveness of a broad range of intervention programs and to examine possible moderation effects after the publication of Lee's (1990) meta-analysis. Examining the intervention effects across the three conditions through separate meta-analyses provides information about the malleability of programming knowledge and skills from multiple perspectives and maps the field of programming instruction by providing some references against which researchers can evaluate their instructional interventions.

2. Method

We based this set of meta-analyses on a systematic review of the primary literature and followed certain steps to identify and extract the relevant information from the primary studies (Card, 2012). These steps included an extensive literature search, the screening of potential publications, and the extraction and coding of relevant information reported in eligible publications. Finally, we performed statistical analyses to synthesize the evidence surrounding the effectiveness of programming instruction.

2.1. Literature search

We extracted the literature relevant to the effectiveness of programming interventions from multiple sources (see Fig. 2): (a) Main databases in the field (ACM Digital Library, IEEE Xplore Digital Library, ERIC, PsycINFO, and Learn Tech Library) and supplementary databases (ProQuest Dissertations and Theses Database, Google Scholar,[1] and ResearchGate); (b) Academic journals (e.g., Computers & Education, Journal of Educational Computing Research); (c) Reference lists of previous meta-analyses and review articles (e.g., Liao & Bright, 1991; Grover & Pea, 2013; for a detailed reference list, please refer to Supplementary Material S2); (d) Vitae of scholars who have published studies or reviews in the field of computer science education with a focus on programming (e.g., Douglas Clements); and (e) Inquiries concerning unpublished studies via email. Our search included publications that were published between January 1, 1965 and January 31, 2017. We used the following search terms: (Programming OR coding OR code OR Scratch* OR Logo* OR Mindstorm* OR computing OR computational thinking) AND (teach* OR learn* OR educat* OR student* OR intervention OR training) AND Computer* AND (compar* OR control group* or experimental group* OR treatment). This set of search terms was comprised of four key elements: The first represented the context of computer programming and included some alternative terms used in the extant literature, such as coding or computational thinking. To capture studies that may not have used one of these terms in their titles, abstracts, or keyword lists, we further added the names of prominent computer programming languages, such as Logo and Scratch—this strategy was recommended by Scherer, Siddiq, and Sánchez Viveros (2019) in their recent meta-analysis. The former was especially important for identifying studies that were conducted in the 1980s and 1990s. The second set of terms defined the context and type of studies and was used to identify interventions that focused on fostering computer programming skills. The third search term defined the technology used to foster programming. Finally, the fourth set of search terms specified the design of the studies, that is, an experimental or quasi-experimental design that included a control and a treatment group. Overall, the four categories of search terms essentially defined the key constructs, the educational context, technology, and the design of the primary studies. These categories are considered essential in meta-analyses of technology-based interventions (e.g., Bernard, Borokhovski, Schmid, Tamim, & Abrami, 2014; Chauhan, 2017). In case Boolean search mechanisms were not available, we had to modify these groups of search terms. Supplementary Material S2 contains the full list of searches in the databases, including the details about necessary adaptations. The search for relevant literature yielded 5193 publications which were submitted to further screening (see Fig. 2).

[1] Given the limited options Google Scholar provides to conduct a systematic literature search based on standardized search terms (Atkinson & Cipriani, 2018) and the hard-to-manage number of search results (>1.7 million), we extracted only the first 100 entries (see Haddaway, Collins, Coughlin, & Kirk, 2015) and screened them.

Table 2
Instructional approaches to fostering programming skills.

Instructional approach	Example intervention(s)	Example reference(s)
Blended learning	Blended learning compared to face-to-face instruction of computer programming	Grover, Pea, and Cooper (2015); Olelewe and Agomuo (2016)
Collaboration	Teaching programming collaboratively vs. individually	Jehng and Chan (1998); Lai & Xin (2011)
Feedback	Continuous feedback on students' programming performance, feedback in structured teaching environments	Chao (1999); Johnson and Kane (1992)
Game-based learning	Game-based instruction of object-oriented programming, game development in Scratch	Cetin (2016); Rodríguez Corral, Civit Balcells, Morgado Estévez, Jiménez Moreno, and Ferreiro Ramos (2014)
Metacognition	Reflecting on problem-solving approaches, fostering metacognitive strategies	Lehrer, Lee, and Jeong (1999); Volet and Lund (1994)
Problem solving	Discovery learning vs. teacher-directed learning, teaching specific problem-solving methods and strategies	Suomala and Alajaaski (2002); Uysal (2014)
Others	Unidirectional vs. reciprocal teaching	Liu et al. (2013); Shadiev et al. (2014)

Note. Please find a more detailed description of the instructional approaches in the Supplementary Material S1.

2.2. Screening and eligibility criteria

After removing duplicates, we screened the titles and abstracts of 708 publications for (a) their relevance for examining the effectiveness of interventions of computer programming; (b) the presence of an intervention; (c) their quantitative nature; (d) English as their language of reporting (see Fig. 2). This initial screening resulted in 440 publications, which were further submitted to the screening of full texts.

One of the key criteria we applied to screen publications referred to the design of the primary studies—we only included studies which contained at least one control group and which followed either an experimental or a quasi-experimental design (i.e., posttest-only or pretest-posttest designs). Hence, we excluded pre-experimental designs which did not include any control groups. Besides, studies were excluded if (a) full texts or secondary sources containing sufficient information about the interventions were not available; (b) the results of the interventions were not reported sufficiently; (c) outcome measures

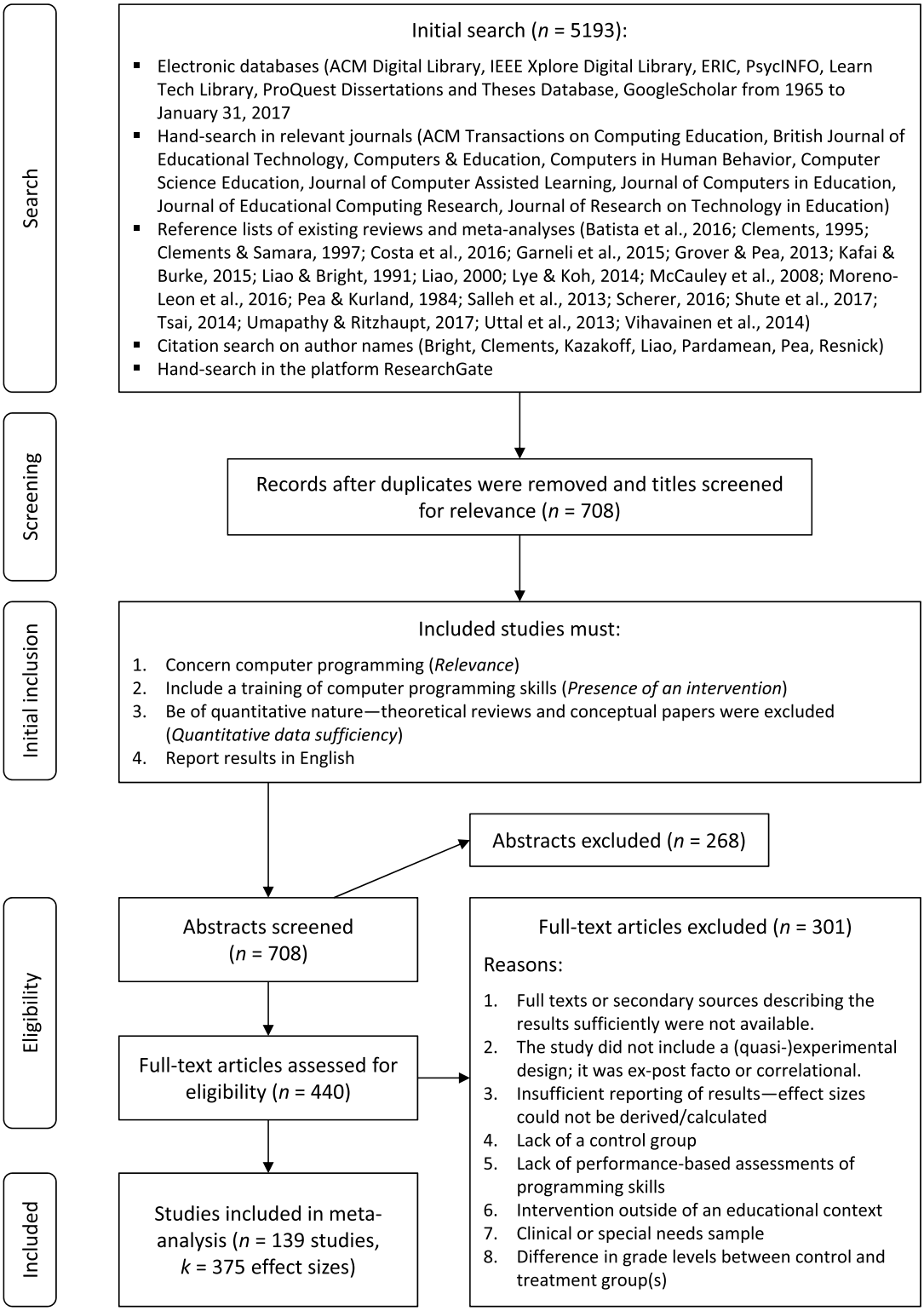


Fig. 2. Flow diagram describing the literature search and the selection of eligible training studies (adapted from the PRISMA Statement; Moher, Liberati, Tetzlaff, Altman, & The PRISMA Group, 2009).

were not based on performance assessments of programming skills; (d) interventions were conducted outside of educational contexts in which students received an instruction (e.g., studies in which students learned computer programming autodidactically without any teaching stimulus); (e) clinical or special needs samples were included; (f) control and treatment groups differed in their grade levels (see Fig. 2). We double-screened 20% of all eligible full texts to ensure the reliability of our inclusion/exclusion criteria. The resultant interrater agreement was high, weighted $\kappa = 0.97$. Any disagreement was resolved by discussing and reviewing specific cases. Overall, the screening of full texts yielded 139 eligible studies that provided 375 effect sizes. Supplementary Material S1 contains the full set of effect sizes; Supplementary Material S2

contains the corresponding reference list.

2.3. Effect size measures

Effect sizes were extracted directly from the primary studies or calculated based on the reported statistics. For pretest-posttest designs with a treatment group (T) and a control group (C), we calculated Hedges' *g* from the standardized mean difference *ES* as follows (Lipsey & Wilson, 2001):

$$ES = \frac{(\overline{X}_{T,Post} - \overline{X}_{T,Pre}) - (\overline{X}_{C,Post} - \overline{X}_{C,Pre})}{SD_{Pooled}}$$

$\overline{X}_{T,Pre}$ and $\overline{X}_{T,Post}$ represent the pretest and posttest mean scores of the treatment, and $\overline{X}_{C,Pre}$ and $\overline{X}_{C,Post}$ of the control group, respectively. SD_{Pooled} represents the pooled standard deviation of the pretest scores, which is calculated as follows (Morris, 2008; Schmidt & Hunter, 2014):

$$SD_{Pooled,Pre} = \sqrt{\frac{(N_T - 1)SD^2_{T,Pre} + (N_C - 1)SD^2_{C,Pre}}{N_T + N_C - 2}}$$

N_T and N_C represent the sample sizes of the treatment and control group, and $SD^2_{T,Pre}$ and $SD^2_{C,Pre}$ their pretest score variances. We then transformed the effect size *ES* into Hedges' g (with $df = N_T + N_C - 2$):

$$g = \left(1 - \frac{3}{4df - 1}\right) ES$$

The corresponding variance v_g and the standard error SE_g were then calculated as follows:

$$v_g = \left(1 - \frac{3}{4df - 1}\right)^2 \left(\frac{N_T + N_C}{N_T N_C} + \frac{ES^2}{2(N_T + N_C)}\right)$$

$$SE_g = \sqrt{v_g}$$

For posttest-only designs, we applied the same calculations, yet without the pretest scores and their standard deviations. In the cases where the authors of the primary studies reported only the results of statistical tests of mean differences (e.g., *t*- or *F*-tests), we used the reported statistics to calculate the effect size *ES* (for more details on these calculations, please refer to Lipsey & Wilson, 2001). We refrained from correcting the resulting effect sizes for the unreliability of the outcome measures for two reasons: (a) Most studies did not provide information on the reliability of the outcome measures; (b) The psychometric literature does not draw a clear picture about the effects unreliability corrections may have on the overall effect sizes and their variance components—in fact, the necessity to correct for unreliability has been discussed controversially (Cheung, 2015; Schmidt & Hunter, 2014).

2.4. Coding of studies

To identify the information that could be gained from the primary studies, examine possible moderation effects, and ultimately classify studies into three main conditions, we coded all study features as either categorical or continuous variables. These variables served at the level of effect sizes, studies, or both. This selection of variables was based on the findings from existing reviews, meta-analyses, and interventions which identified them as moderators or sources of differential effectiveness (e.g., Liao, 2000; Liao & Bright, 1991; Shute et al., 2017; Umapathy & Ritzhaupt, 2017). These variables further describe the contexts or conditions under which programming interventions may or may not succeed (Grover & Pea, 2013). To ensure the reliability of the coding, about 25% of the full texts were double-coded; the resulting agreement was 94%, and disagreements were resolved during a discussion session until consensus had been reached. Supplementary Material S1 contains all coded variables.

Classification of studies (Study conditions). Given the diversity of effects examined in the primary studies, we classified the studies according to the type of effects they allowed us to investigate. The resultant variable "Classification" was informed by the framework of intervention studies proposed by Chen et al. (2018). More concisely, primary studies were classified into one of the following three conditions (see also Table 1):

(1) *Studies that reported the effectiveness of programming instruction* per se *($m = 12$, $k = 14$):* These studies included at least one treatment group that was exposed to programming instruction and at least one control group that engaged in instruction other than programming. Examples of interventions are: Programming instruction with Lego Mindstorms® (experimental group) vs. no programming at all (control group; e.g., Milner, 1973; Nugent, Barker, Grandgenett, & Adamchuk, 2010); programming instruction to solve mathematical problems (experimental group) vs. instruction to solve mathematical problems without the involvement of programming (control group; e.g., Oprea, 1984; Psycharis & Kallia, 2017).

(2) *Studies that reported the effectiveness of visualization ($m = 20$, $k = 46$) or physicality ($m = 7$, $k = 27$):* These studies examined the effectiveness of visual programming tools or tools that involve physicality, that is, students can observe the result of their programming activities via the movements of physical objects. Examples of interventions are: Visualizing programming languages (experimental group) vs. representation of programming languages as only text (control group; e.g., Siozou, Tselios, & Komis, 2008); Programming instruction with visual programming language A (experimental group) vs. programming instruction with visual and text-based language B (control group; e.g., Cetin, 2016; Daly, 2013); Programming involving robotics (experimental group) vs. programming without robotics (e.g., Huang, Yang, & Cheng, 2013; Rodríguez Corral, Civit, Perez-Peña, & Molina, 2016).

(3) *Studies that reported the effectiveness of instructional approaches ($m = 88$, $k = 263$):* These studies examined the effects of instructional practices that did not involve the modification of the programming tools—control and treatment groups differed in their instruction, yet not the programming languages students used. Examples of interventions are: Pair programming (experimental group) vs. individual programming (control group; e.g., Altintas, Gunes, & Sayan, 2016); discovery learning or problem-solving instruction (experimental group) vs. teacher-directed instruction (control group; e.g., Carney, 2000; Yang, Hwang, Yang, & Hwang, 2015). During the systematic review, the existing body of instructional approaches was extracted and classified into the following categories: (a) Blended learning, (b) Collaboration, (c) Feedback, (d) Game-based learning, (e) Metacognition, (f) Problem solving, (g) Others. Although these approaches are well-aligned with the extant literature reviewing programming instruction (e.g., Hsu et al., 2018), this list is by no means exhaustive. In fact, other approaches may play an important role for computer science educators, such as storytelling, scaffolding, or critical computational literacy instruction (Hsu et al., 2018); however, the primary studies we extracted from the literature databases only allowed us to examine and synthesize the effectiveness of the beforementioned instructional approaches.

The detailed list of studies including their classification and a description of the study effects can be found in the Supplementary Material S1 (variable "Classification").

Outcome variables. As noted earlier, we referred to a broad conceptualization of programming skills in this meta-analysis, allowing both knowledge and skill domains as outcome variables. To further differentiate between different dimensions of programming skills—and therefore perhaps find evidence for or against the differential

effectiveness of programming interventions—we coded the outcome variables as either 'programming knowledge' or 'programming skills'. The former comprised procedural and conceptual knowledge; the latter comprised the skills to create, evaluate, and refine code as well as debugging and engaging in computational practices in general. Several measures of programming knowledge and skills were used—these measures comprised students' performance on knowledge tests (e.g., Logo Knowledge Test; see Lehrer, Lee, & Jong, 1999) or computational thinking tests (e.g., Jenkins, 2015), next to their course performance (e.g., measured by course grades or performance scores of programming assignments; Barak, Harward, Kocur, & Lerman, 2007) and exam scores (e.g., Shyr, 2010). We note that the computational thinking tests assessed mainly skills rather than knowledge; this skillset comprised the creation, modification, or application of computer code—these tests consequently fell into the category of skills tests. Besides, some authors used process and product data to describe and evaluate students' programming performance (e.g., by evaluating code, Liu et al., 2013; by evaluating indicators of programming difficulty, Mason & Cooper, 2013). We notice that all outcome measures of programming knowledge and skills were performance-based and did not include any self-report measures. Students' performance was indicated by test scores, grades, or scores that describe the quality of the programming code. Overall, the two outcome categories programming knowledge and skills may include overlapping competences; however, it was not possible to provide a greater level of granularity due to the limited reporting of the more specific sub-competences measured by the tests or exams.

Instructional approaches. Exploring the studies that reported the effectiveness of dominant instructional approaches (Flórez et al., 2017; Hsu et al., 2018), we found that the intervention programs focused on blended learning, the provision of feedback, learning programming through computer games, fostering metacognition, collaborative activities, problem solving instruction, and others. Table 2 gives an account of these instructional approaches and contains sample references; Supplementary Material S1 contains more detailed descriptions of these approaches for each study. We note that the category "Collaboration" contains primary studies that compared an intervention group in which students learned programming collaboratively with a control group in which students worked individually. This category also contained studies that examined the effectiveness of so-called "pair programming".

Programming tools. We coded the programming tools used in the interventions as 'visual' (e.g., Scratch, Alice), 'text-based' (e.g., C, Java), or a 'mixture' of both. Given the popularity of Lego Mindstorms®, Logo, and Scratch in recent years (Hsu et al., 2018), we further identified more specifically whether or not these three tools were used.

Study features. The design of the primary studies was coded as either a 'pretest-posttest control group design' or a 'posttest-only design'. Given that some studies contained multiple measures and samples, it was possible that multiple designs occurred within one study. For instance, if the authors of a study administered a programming skills test before and after the intervention and a programming knowledge test only after the intervention, the study contained both designs—that is, a pretest-posttest design for the former and a posttest design for the latter. Hence, the study design was primarily a variable at the level of effect sizes. Next to the study design, we also coded the randomization (i.e., 'randomized', 'not randomized') and matching (i.e., 'matched', 'not matched') of the experimental groups, the collaboration among students during the intervention (i.e., 'collaboration', 'no collaboration'), the study context (i.e., 'regular lessons', 'extracurricular activity'), and the standardization of the outcome measures (i.e., 'standardized', 'unstandardized'). Finally, the intervention length was coded as the time spent on the intervention in hours. The selection of these study features was based on the previous meta-analyses, including that conducted by Scherer et al. (2019) on the transfer effects of computer programming.

Sample features. Sample features comprised the educational level the intervention was targeted at (i.e., 'primary', 'secondary', or 'tertiary' education), the continent the study sample originated from (i.e., 'Asia', 'Europe', 'North America', or 'Others'; the latter included Australia and African countries and occurred seldomly), the average age of students in years, and the proportion of female students in the primary studies.

Publication features. We established publication status as another, possible moderating variable and based the definition of "grey literature" on Adams, Smart, and Huff's (2017) framework. In this framework, grey literature included dissertations, conference proceedings, working papers, book chapters, technical reports, and other references that have not been published in scholarly journals after peer-review (see also Schmucker et al., 2017). Publication status was thus coded as 'grey' or 'published'. Despite the efforts taken (e.g., contacting the authors via informal platforms, such as ResearchGate or the mailing lists of computer science education societies), unpublished studies could not be retrieved. Next to the status of publication, we kept track of the year of publication.

2.5. Statistical analyses

The meta-analytic data within the three study conditions have a nested structure, because many studies reported multiple effect sizes. This nesting of effect sizes in studies represents a violation of the independence assumption in classical meta-analysis (Borenstein, Hedges, Higgins, & Rothstein, 2009). As a consequence, we took an approach that directly accounted for the dependencies between effect sizes, namely three-level random-effects meta-analysis (Cheung, 2014). In three-level random-effects meta-analysis, the variation of effect sizes between studies (level 3, variance σ_3^2) and their variation within studies (level 2, variance σ_2^2) are quantified in addition to the sampling variability (level 1). For a given data set of primary studies exhibiting a nested structure, these variance components can be estimated and tested for their deviation from zero by means of model comparisons (i.e., comparing a model with freely estimated variances with a model constraining these variances to zero). Cheung (2015) suggested using the likelihood-ratio test to conduct such model comparisons (see Supplementary Material S3–S8). Nevertheless, as the testing of significant within- and between-study variances is against the boundary of zero, the confidence intervals of the variances may contain zero, and the likelihood-ratio tests may indicate only a marginal difference in model fit (Cheung, 2015). As a consequence, several authors argued that the decision for a baseline model with random effects should not only be based on the significance tests of variances and heterogeneity tests only, but relies mainly on the substantive assumptions on whether the effect sizes may or may not vary within or between studies (Cheung, 2015; Viechtbauer, 2005). Acknowledging the limitations of the statistical tests and considering that the meta-analytic data are hierarchical, we chose the three-level random-effects model as the baseline model.

For the three study conditions, we performed separate meta-analyses to obtain the aggregated effect sizes specific to these conditions (see also Chen et al., 2018). More specifically, to ensure that studies reporting the same type of effects within each condition are synthesized (and thus a validity argument for the overall effect sizes can be crafted), we performed one meta-analysis for study condition 1, two meta-analyses for study condition 2 (i.e., for primary studies focusing on visualization or physicality), and six meta-analyses for study condition 3 (i.e., one for each instructional approach).

To examine the extent to which study, sample, and publication features may explain variation within or between studies, we extended the meta-analytic baseline models to three-level mixed-effects models (Cheung, 2015). Categorical moderators with more than two categories were dummy-coded, and moderators without any variation across effect sizes or only one effect size within a category were not considered in these analyses. Continuous moderators were z-transformed or, in the case of proportions, arcsine-transformed.

We specified all models in the R package 'metafor' using restricted maximum likelihood estimation (Viechtbauer, 2017), and variance

explanations were obtained from the reduction of level-2 and level-3 variances (Cheng, Cheung, & Wang, 2018). Please find the corresponding R code and output in the Supplementary Material S3–S8.

2.6. Publication bias, influential effect sizes, and sensitivity analyses

To determine the degree of publication bias present in the meta-analytic data sets in each study condition, we conducted several analyses: First, we performed trim-and-fill analyses and examined the funnel plot of effect sizes to identify a possible asymmetry that might be due to publication bias (Duval & Tweedie, 2000). These analyses provided an overall intervention effect size in each condition adjusted for publication bias and the number of missing studies to achieve symmetry in the funnel plot. We further tested the asymmetry using Egger's linear regression test (Egger, Smith, Schneider, & Minder, 1997). Second, we estimated the fail-safe *N* based on Rosenberg's procedure (Borenstein et al., 2009). Third, we examined the *p*-curve underlying the all intervention effects on computer programming in the data set (Simonsohn, Nelson, & Simmons, 2014). If the *p*-curve is right-skewed, the primary studies have evidential value and there is no evidence for *p*-hacking. We used the '*P*-curve Online App' to obtain the *p*-curve (Simonsohn, Nelson, & Simmons, 2017).

Besides the analysis of publication bias, we identified influential effect sizes using Viechtbauer and Cheung's (2010) diagnostics using the R package 'metafor'. An effect size was considered influential if the leave-one-out diagnostics exceeded the common thresholds (for more details on these thresholds, please refer to Viechtbauer, 2017). Please find the corresponding diagnostic plots in the Supplementary Material S3, S5, and S7. If, indeed, influential effect sizes are detected, researchers have several options to handle them—either delete or keep them. Such decisions, however, are to be supplemented by a review of the study, sample, and publication features, which may or may not indicate poor study quality. In fact, if an effect size is identified as influential *and* the study quality is poor (e.g., no randomization, posttest-only design, small sample sizes, no information about the reliability of measures; Valentine, 2019), researchers may well exclude it from the meta-analytic data. In the present study, we followed this procedure (i.e., examining the features of the study that exhibits influential effect sizes). If effect sizes were indeed excluded, we also studied the effects of this exclusion on the meta-analytic model parameters. These effects are reported as part of the sensitivity analyses.

3. Results

3.1. Description of studies

The full sample comprised of 139 primary studies yielding 375 effect sizes from 26,864 students (control groups: N_C = 13,090, treatment groups: N_T = 13,774). Most studies followed a posttest-only design (74.1%), included active control groups (92.8%) that were not matched with the treatment group (77.7%), and that implemented the programming intervention as part of regular school lessons (83.5%). About half of the studies reported a randomization of the experimental groups (49.6%). The study samples mainly included college and university students (72.3%), whose age ranged between 7 and 27 years, and the average proportion of female students was 46.0% (SD = 14.7%, Mdn = 50.0%). Interventions lasted between one and 105 h (M = 21.4, SD = 18.5, Mdn = 20 h). Supplementary Material S1 contains the raw data underlying this description.

3.2. P-curve and influential effect sizes

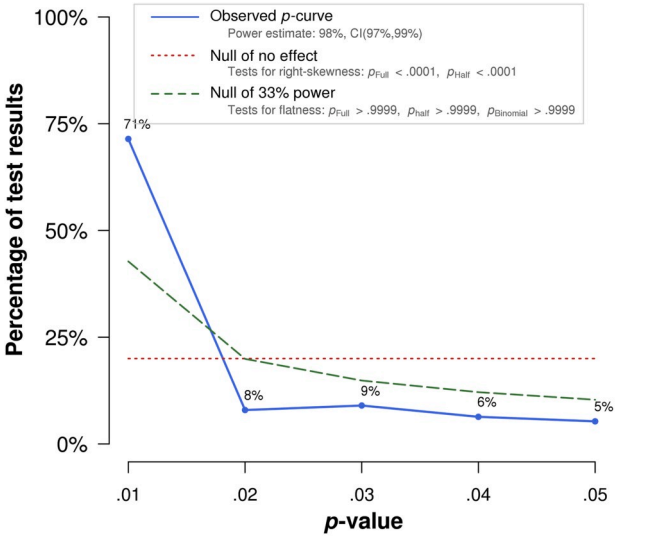
The *p*-curve was right-skewed and suggested that the pool of effect sizes extracted from the primary studies had evidential value (see Fig. 3). We identified one influential effect size in study condition 1, one in study condition 2 (physicality), and six influential effect sizes in study condition 3 (collaboration, feedback, metacognition, and others), each of which were flagged by student residuals, Cook's distance, and other leave-one-out deletion measures (see Supplementary Material S3, S5, and S7). These effect sizes were large and positive and ranged between g = 1.74 and g = 4.08. After reviewing the study, sample, and publication features underlying these effect sizes, we decided to remove three of them. Please find the detailed reasoning for this decision in the Supplementary Material S2.

3.3. Effectiveness of programming interventions per Se (RQ1)

Baseline model. To obtain an overall effect size describing the effectiveness of computer programming intervention per se, we established a baseline model that accounts for the variation of effect sizes within (level 2) and between studies (level 3). This three-level random-effects model resulted in an overall effect size of $\bar{g}$ = 0.814 (95% CI [0.420, 1.207]), a significant variance σ_3^2 (see Table 3), and provided evidence for the heterogeneity of effects (Q [12] = 59.8, $p < .001$; I_2^2 = 0.0%, I_3^2 = 93.7%). These indices suggest substantial variation of effect sizes between rather than within studies, given that only two studies provided multiple effect sizes. The profile plot showed a maximum at the estimate $\hat{\sigma}_3^2$ and a decrease on log-likelihood values when moving further away from it (see Supplementary Material S4). The between-study variance can therefore be identified. Although the within-study variance was small, and its 95% confidence interval contained zero, we still allowed for its estimation due to the issues associated with testing this variance against its boundary (Cheung, 2015).

Moderator analysis. Due to the small number of effect sizes and primary studies in this category, we were not able to conduct meaningful moderator analyses—moderator effects would have been underpowered, and variances and variance explanations may not have been reliably estimated, especially for subgroups of studies containing only one or two effect sizes (e.g., Jackson & Turner, 2017; Valentine, Pigott, & Rothstein, 2010). Nevertheless, we reported the effect size for all moderator categories in the Supplementary Material S4.

Sensitivity analysis and publication bias. To examine the effect of the influential effect size, we estimated the three-level random-effects model for the full sample of studies in this condition, that is, the sample of primary studies keeping the one influential case (see Supplementary Material S3). This model revealed a positive, statistically significant, and slightly larger intervention effect, $\bar{g}$ = 1.047 (95% CI [0.472, 1.622], z =



Note: The observed *p*-curve includes 189 statistically significant ($p < .05$) results, of which 160 are $p < .025$. There were 158 additional results entered but excluded from *p*-curve because they were $p > .05$.

Fig. 3. *P*-curve of the full sample of effect sizes.

Table 3
Results of the baseline models describing the overall intervention effects for study conditions 1 and 2.

	Study condition 1	Study condition 2	
	Effectiveness of Programming Instruction Per Se	Effectiveness of Visualization	Effectiveness of Physicality
Overall effect size			
$\bar{g}$	0.814	0.436	0.718
95% CI	[0.420, 1.207]	[0.289, 0.583]	[0.226, 1.210]
z-value	4.05	5.81	0.01
p-value	<.001	<.001	.004
m	11	20	7
k	13	46	27
Variance estimates			
Within-study variance			
σ_2^2	0.000	0.018	0.000
95% CI	[0.000, 0.543]	[0.000, 0.091]	[0.000, 0.027]
Between-study variance			
σ_3^2	0.359	0.062	0.392
95% CI	[0.000, 1.306]	[0.000, 0.204]	[0.119, 1.735]
Heterogeneity test			
Cochran's *Q*	59.78	93.48	65.31
df	12	45	26
p-value	<.001	<.001	<.001
Heterogeneity indices			
I_2^2	0.0%	13.8%	0.0%
I_3^2	86.1%	47.2%	84.9%
Publication bias			
Rosenberg's fail-safe *N*	277	955	155
Egger's linear regression test			
t-value	0.00	1.93	1.25
df	11	44	25
p-value	1.00	.06	.22

Note. $\bar{g}$ = Weighted average effect size Hedges' *g*, 95% *CI* = 95% Wald confidence interval, *m* = Number of studies, *k* = Number of effect sizes, *df* = degrees of freedom, I_2^2 = Heterogeneity index for level 2, I_3^2 = Heterogeneity index for level 3 (see Cheung, 2015). The analysis of publication bias was based on a two-level random-effects model.

3.6, $p < .001$). The within-study variance was $\sigma_2^2 = 0.000$ (95% CI [0.000, 0.674]), and the between-study variance was $\sigma_3^2 = 0.922$ (95% CI [0.130, 2.874]), indicating larger variation and uncertainty in the estimates. The removal of the influential case decreased the overall effect size; yet, the conclusion that a large, positive, and significant effect of programming instructions per se exists remained.

The trim-and-fill analyses indicated that no study was missing on the left side of the funnel plot ($SE = 2.124$), and Egger's linear regression test suggested that the no statistically significant funnel plot asymmetry was given (see Supplementary Material S3). In light of the small number of effect sizes, the fail-safe *N* was large (Table 3).

3.4. Effectiveness of visualization and physicality (RQ2)

Baseline models. For the sample of primary studies examining the effects of *visualization*, the three-level random-effects models resulted in an overall and significant effect size of $\bar{g} = 0.436$ (95% CI [0.289, 0.583]). The within-study variation was small, while the between-study variation was substantial (see Table 3). Moreover, significant heterogeneity of effect sizes was indicated, Q [45] = 93.5, $p < .001$. For the sample of primary studies examining the effects of *physicality*, the overall effect size was large, $\bar{g} = 0.718$ (95% CI [0.226, 1.210]). Similar to the visualization effects, the within-study variation was negligible but some between-study variation existed (see Table 3). However, the effect sizes were homogeneous, Q [25] = 28.3, $p = .295$.

3.4.1. Moderator analyses[2]

Visualization. While the study design and publication features did not exhibit significant moderation effects (see Supplementary Material S5), some sample features did. Specifically, primary studies involving Asian student samples showed higher effect sizes ($\bar{g} = 0.801$, 95% CI [0.567, 1.005]) than samples comprising students from other continents ($\bar{g}$s = 0.053–0.348)—the difference was statistically significant ($B = 0.748$, $SE = 0.206$, $p < .001$; $R_2^2 = 0.261$, $R_3^2 = 0.865$; $Q_M[3] = 20.7$, $p < .001$). Moreover, the proportion of female students in the primary studies was positively associated with the overall effect size ($B = 1.922$, $SE = 0.450$, $p < .001$, with arcsine transformation). Finally, the primary studies involving the visual programming Scratch showed larger effect sizes ($\bar{g} = 1.014$, 95% CI [0.562, 1.466]) than those involving other programming languages ($\bar{g} = 0.380$, 95% CI [0.248, 0.512])—these effects were statistically significant ($B = 0.634$, $SE = 0.240$, $p = .008$; $R_2^2 = 0.008$, $R_3^2 = 0.525$).

Physicality. The three-level mixed-effects models identified several study and sample features as significant moderators (see Supplementary Material S5). Similar to the studies focusing on visualization, studies comprising Asian samples showed larger effects ($\bar{g} = 1.574$, 95% CI [1.154, 1.995]) than those comprising other samples ($\bar{g}$s = 0.216–0.871). This moderation effect was statistically significant ($B = 1.358$, $SE = 0.221$, $p < .001$; $Q_M[2] = 46.0$, $p < .001$). Furthermore, the effectiveness of physicality as a means to programming instruction was significantly smaller for samples enrolled in secondary education ($\bar{g} = 0.238$, 95% CI [0.115, 0.360]) than for primary ($\bar{g} = 1.439$, 95% CI [1.083, 1.795]) or tertiary education ($\bar{g} = 1.472$, 95% CI [1.130, 1.815]; $B = -1.235$, $SE = 0.178$, $p < .001$; $Q_M[2] = 48.3$, $p < .001$). The average age of the student samples was positively associated with the effect sizes ($B = 0.251$, $SE = 0.124$, $p = .043$). Finally, short-term interventions were more effective than longer interventions, as the negative moderation effect of intervention length indicated ($B = -0.245$, $SE = 0.090$, $p = .007$).

Sensitivity analyses and publication bias. After excluding one influential effect size, the overall intervention effect of interventions focusing on physicality decreased, $\bar{g} = 0.478$ (95% CI [0.149, 0.808], $z = 2.8$, $p = .004$), and so did the within-study ($\sigma_2^2 = 0.000$, 95% CI [0.000, 0.027]) and between-study variances ($\sigma_3^2 = 0.129$, 95% CI [0.008, 0.840]). The moderation effects, however, could also be found in the reduced sample (see Supplementary Material S6).

For the visualization interventions, five primary effect sizes were missing to achieve symmetry in the funnel plot ($SE = 4.413$; see Supplementary Material S5), reducing the overall effect to $\bar{g} = 0.373$ (95% CI [0.264, 0.482], $z = 6.7$, $p < .001$). Egger's regression test, however, indicated that asymmetry was not significant, and the fail-safe *N* was large (see Table 3). For the physicality interventions, the trim-and-fill analyses indicated that no study was missing on the left side of the funnel plot ($SE = 0.1271$), and Egger's linear regression test suggested that the no statistically significant funnel plot asymmetry was given (see Supplementary Material S5), and the fail-safe *N* was large (Table 3). Overall, these results suggested that some degree of publication bias existed in the visualization condition.

3.5. Effectiveness of instructional approaches (RQ3)

3.5.1. Baseline models

Overall study sample. As noted earlier, we performed separate meta-analyses for each of the instructional approaches to ensure the comparability of effects reported in the primary studies within these approaches. Nevertheless, to set a reference of instructional effectiveness against which the resultant effect sizes for each approach could be evaluated, we specified and estimated a baseline model for the entire data in this study condition ($m = 88$, $k = 263$; see Supplementary Material S7). The resultant three-level random-effects model yielded an overall effect size of $\bar{g} = 0.598$ (95% CI [0.494, 0.702], $z = 11.29$, $p <$

Table 4
Results of the baseline models describing the overall intervention effects for study condition 3.

	Study condition 3					
	Effectiveness of instructional approaches					
	Blended learning	Collaboration	Feedback	Game-based learning	Metacognition	Problem solving
Overall effect size						
$\bar{g}$	1.023	0.560	0.493	0.821	0.658	0.518
95% CI	[0.291, 1.756]	[0.353, 0.767]	[0.207, 0.780]	[-0.126, 1.768]	[0.332, 0.983]	[0.378, 0.659]
z-value	2.74	5.30	3.38	1.70	3.95	7.25
p-value	.006	<.001	<.001	.089	<.001	<.001
m	3	24	10	2	14	27
k	4	50	78	3	49	63
Variance estimates						
Within-study variance						
σ_2^2	0.039	0.164	0.245	0.591	0.059	0.243
95% CI	[0.000, 2.563]	[0.083, 0.329]	[0.140, 0.407]	[0.005, 13.088]	[0.000, 0.245]	[0.149, 0.392]
Between-study variance						
σ_3^2	0.346	0.148	0.073	0.000	0.289	0.000
95% CI	[0.000, 7.183]	[0.026, 0.395]	[0.000, 0.684]	[0.000, >10.000]	[0.018, 0.969]	[0.000, 0.072]
Heterogeneity test						
Cochran's *Q*	21.98	361.89	277.82	9.98	129.17	280.10
df	3	49	77	2	48	61
p-value	<.001	<.001	<.001	.007	<.001	<.001
Heterogeneity indices						
I_2^2	8.9%	46.9%	58.1%	85.9%	12.4%	82.5%
I_3^2	78.6%	42.3%	17.3%	0.0%	61.1%	0.0%
Publication bias						
Rosenberg's fail-safe *N*	43	2649	1046	9	892	1996
Egger's linear regression test						
t-value	−2.02	1.74	0.34	0.27	1.68	2.52
df	2	48	76	1	47	60
p-value	.18	.09	.74	.83	.10	.01

Note. $\bar{g}$ = Weighted average effect size Hedges' *g*, 95% *CI* = 95% Wald confidence interval, *m* = Number of studies, *k* = Number of effect sizes, *df* = degrees of freedom, I_2^2 = Heterogeneity index for level 2, I_3^2 = Heterogeneity index for level 3 (see Cheung, 2015). The analysis of publication bias was based on a two-level random-effects model.

.001) and exhibited within-study and between-study variation (Q [262] = 1348.91, $p < .001$; $\sigma_2^2 = 0.276$, 95% CI [0.203, 0.368]; $\sigma_3^2 = 0.072$, 95% CI [0.012, 0.187]; $I_2^2 = 66.8\%$, $I_3^2 = 17.4\%$). Introducing the type of instructional approach as a moderator variable to this model indicated that the effect sizes did not differ significantly across approaches, $Q_M(6) = 2.81$, $p = .83$. Moreover, modeling the instructional approaches as another level of analysis (i.e., in a four-level random-effects model) showed that the between-approaches variance was negligible ($\sigma_4^2 = 0.000$, 95% CI [0.000, 0.038]), and the information criteria were reduced only marginally (three-level model: AIC = 536.8, BIC = 547.5; four-level model: AIC = 538.8, BIC = 553.0). Hence, there was no evidence supporting the statistically significant differences in effect sizes between instructional approaches. We notice that the overall effect size across the instructional approaches should not be further interpreted substantively, given the different nature of effects and experimental conditions across the primary studies.

Separate meta-analyses for the instructional approaches. Table 4 shows the results of the separate meta-analyses, that is, the parameters of the three-level random-effects models for each of the instructional approaches. Supplementary Material S7 and S8 contain the corresponding data input and analytic output files. Overall, the average effect sizes ranged between $\bar{g} = 0.493$ (feedback) and $\bar{g} = 1.023$ (blended learning) and thus exhibited moderate to large effects. Notably, the samples of primary studies and effect sizes varied in their sizes across approaches. Specifically, two approaches contained only three (blended learning) or four effect sizes (game-based learning); the resultant average effect sizes and their variance components should therefore be interpreted with caution. However, all other approaches contained between 40 and 78 effect sizes and provided moderate and statistically significant effect sizes (see Table 4). These effects varied mainly within studies (collaboration, feedback, and problem solving), and in only one case substantially between studies (metacognition). Next to the effects of these well-defined instructional approaches, seven studies provided 15 effect sizes indicating the effectiveness of other approaches (category "Other"). Synthesizing these effects resulted in a moderate overall effect size, $\bar{g} = 0.490$ (95% CI [0.028, 0.952]; see Supplementary Material S7 and S8). However, we neither interpreted nor extended the meta-analysis of this category by moderator variables due to the lack of comparability of effects within it.

Moderator analyses.[2] In the following, we will summarize the results of the moderator analyses for each instructional approach. These analyses, however, excluded the following categories: Blended learning, game-based learning, and others. For more details on the moderator analyses, we refer readers to the Supplementary Material S7 and S8.

Collaboration. Variation in the effect sizes for this instructional approach could be explained by the following moderators: (a) *Test type*: Primary studies administering standardized tests showed higher effects ($\bar{g} = 1.323$, 95% CI [0.954, 1.692]) than those administering non-standardized tests of programming knowledge or skills ($\bar{g} = 0.438$, 95% CI [0.274, 0.601])—this difference was statistically significant ($B = 0.885$, $SE = 0.206$, $p < .001$). (b) *Educational level:* Sample comprising students enrolled in primary education showed the smallest and insignificant effects with large uncertainty ($\bar{g} = -0.855$, 95% CI [-1.918, 0.207]), followed by university and college students ($\bar{g} = 0.496$, 95% CI [0.300, 0.692]). The highest effect sizes were indicated for the samples of students in secondary education ($\bar{g} = 1.507$, 95% CI [0.875, 2.139]). These differences were statistically significant ($Q_M[2] = 0.04$, $p = .007$) and explained mainly between-study variation ($R_2^2 = 0.066$, $R_3^2 = 0.461$). (c) *Type of outcome variable:* Primary studies focusing on

[2] In some instances, the variance explanations (within and between studies) could not be estimated reliably due to missing data in the moderator variables.

Table 5
Summary of the main findings.

Research Question (RQ)	Overall Effect Size ($\bar{g}$)	Significant Moderator Effects
RQ1. Effectiveness of programming instruction per se	$\bar{g} = 0.814$, 95% CI [0.420, 1.207]	• *Type of outcome variable*: Larger effects for tests assessing programming knowledge
RQ2a. Effectiveness of visualization	$\bar{g} = 0.436$, 95% CI [0.289, 0.583]	• *Programming tool*: Larger effects for primary studies using Scratch • *Continent*: Larger effects for Asian student samples • *Proportion of female students*: Larger effects for samples with more female students
RQ2b. Effectiveness of physicality	$\bar{g} = 0.718$, 95% CI [0.226, 1.210]	• *Programming tool*: Larger effects for studies involving Lego Mindstorms® • *Continent*: Order of effects, Asian samples > European samples > Samples from other continents • *Educational level:* Smaller effects for samples of secondary school students • *Average age*: Positive association between average age and intervention effects • *Intervention length*: Larger effects for shorter interventions
RQ3. Effectiveness of instructional approaches	$\bar{g} = 0.520$, 95% CI [0.437, 0.603]	–
a Blended learning	$\bar{g} = 1.023$, 95% CI [0.291, 1.756]	–
b Collaboration	$\bar{g} = 0.560$, 95% CI [0.351, 0.767]	• *Type of outcome variable*: Smaller effects on tests assessing programming knowledge • *Test type*: Larger effects for standardized tests • *Educational level*: Order of effects, Secondary > Tertiary education > Primary education • *Proportion of female students*: Smaller effects for samples with more female students
c Feedback	$\bar{g} = 0.494$, 95% CI [0.207, 0.780]	• *Randomization*: Smaller effects for studies with randomly assigned groups
d Game-based learning	$\bar{g} = 0.821$, 95% CI [-0.126, 1.768]	–
e Metacognition	$\bar{g} = 0.658$, 95% CI [0.332, 0.983]	• *Collaboration*: Larger effects for collaborative settings
f Problem solving	$\bar{g} = 0.518$, 95% CI [0.378, 0.659]	• *Study context*: Larger effects for programming instruction as part of extracurricular activities
g Others	$\bar{g} = 0.490$, 95% CI [0.028, 0.952]	–

programming knowledge showed a small and insignificant average effect size ($\bar{g} = -0.036$, 95% CI [-0.636, 0.563]), while those focusing on programming skills showed a moderate average effect size ($\bar{g} = 0.607$, 95% CI [0.405, 0.808])—this difference was statistically significant ($B = -0.643$, $SE = 0.311$, $p = .039$; $R_2^2 = 0.075$, $R_3^2 = 0.160$). (d) *Proportion of female students:* Finally, effect sizes derived from studies with more female students tended to be smaller ($B = -1.109$, $SE = 0.578$, $p = .055$, with arcsine transformation). All other study, sample, and publication features did not show moderating effects (see Supplementary Material S8).

Feedback. Among all possible moderating variables, only the study feature of randomization explained variation in the effect sizes for this category (see Supplementary Material S7). More specifically, primary studies performing randomization showed smaller average effect sizes ($\bar{g} = 0.324$, 95% CI [0.143, 0.506]) than those without randomization ($\bar{g} = 0.999$, 95% CI [0.480, 1.518]). This difference was statistically significant, $B = -0.675$, $SE = 0281$, $p = .016$.

Metacognition. The moderator analyses revealed that primary studies taking metacognitive instruction as an approach to teaching computer programming were more effective when conducted in collaborative settings ($\bar{g} = 1.774$, 95% CI [0.999, 2.549]; $B = 1.355$, $SE = 0.430$, $p = .002$) than in settings with individual work ($\bar{g} = 0.419$, 95% CI [0.090, 0.749]), explaining mainly between-study variance in effect sizes ($R_2^2 = 0.083$, $R_3^2 = 0.661$). Please find more details on the moderator analyses in the Supplementary Material S7.

Problem solving. The context in which the programming instruction focusing on problem solving was conducted moderated the average effect size in this category (see Supplementary Material S7). More specifically, primary studies conducted in extracurricular settings were more effective ($\bar{g} = 0.736$, 95% CI [0.479, 0.992]) than those conducted in regular lessons ($\bar{g} = 0.431$, 95% CI [0.270, 0.591]). This difference was statistically significant ($B = -0.305$, $SE = 0.154$, $p = .048$) and explained only within-study variation ($R_2^2 = 0.094$, $R_3^2 = 0.000$).

Summary. Overall, some of the study, sample, and publication features moderated the average effect sizes for the instructional approaches. However, these moderation effects were by no means systematic, and some of them must be interpreted with caution, given the relatively small number of effect sizes in some of the categories. Table 5 summarizes the effects for study condition 3 and all other conditions.

Sensitivity analyses and publication bias. For the instructional approaches with identified influential effect sizes, we performed sensitivity analyses. A detailed description of the specific results for each approach is provided in Supplementary Material S8. Overall, the exclusion of influential cases reduced the average effects and the corresponding variance components slightly; the moderation effects largely remained. The analyses of publication bias revealed some degree of bias in the problem-solving category (significant Egger's regression test), yet no further evidence for the other categories (see Table 4).

4. Discussion

4.1. Effectiveness of computer programming interventions in the three conditions

Effectiveness per se. Synthesizing the primary studies of the effectiveness of programming interventions per se, we found a large effect for the studies that compared programming instruction with instruction outside the programming domain ($\bar{g} = 0.814$). This effect size serves as a reference point for all other effect sizes—presumably, effect sizes in the other study conditions may be lower, mainly because there was no exposure to programming in the control groups under study condition (1) (see also Scherer et al., 2019). Similarly, Tsai and Tsai (2018), as they reviewed game-based interventions in the domain of language learning, found the largest effects when control groups did not engage in game-based education. The remaining two conditions, indeed, showed moderate effect sizes for the categories with at least five effect sizes ($\bar{g}$s = 0.494–0.718).

Although the effectiveness of programming interventions per se does not carry information directly relevant to instructional approaches and conditions, the finding that the effect size was positive already has several implications and contributions to the field of educational technology: First, it shows that programming knowledge and skills can be taught and acquired. Together with Brown and Wilson (2018), we argue that programmers are "not born but made"—in other words, we agree that the knowledge and skills involved in programming are not necessarily innate, thus contrasting some beliefs about the nature of

programming talent as being innate (Guzdial, 2014). Among other domains, learning computer programming is possible, and intervention programs are, on average, effective in fostering the acquisition of the knowledge and skills needed for the mastery of the art (for a similar discussion in other domains, see Macnamara, Hambrick, & Oswald, 2014). Although the finding that "programming knowledge and skills can be taught and acquired" is expected and may not be considered ground-breaking, it is still a relevant observation as corresponding effect size creates a reference point within the field. More specifically, researchers who may want to evaluate the magnitude of their programming interventions can use this reference. In this sense, the reporting of overall effect sizes contributes to the mapping of the effectiveness of programming intervention next to the effectiveness of other, technology-based interventions. For instance, Chauhan (2017), who meta-analyzed interventions targeted at learning with technology in elementary education, found an overall effect of $\bar{g} = 0.55$; Young (2017) found that learning mathematics with technology is effective for fostering mathematical knowledge and skills, $\bar{g} = 0.38$. Of course, although these meta-analyses contained primary studies focusing on the effectiveness of programming instruction, these effect sizes may only serve as rough references because these meta-analyses also included primary studies with outcome variables outside the programming domain. In their meta-analysis of the transfer effects of learning computer programming on skills other than programming, Scherer et al. (2019) identified a moderate overall effect size, $\bar{g} = 0.47$. The effect size we obtained from primary studies using programming knowledge and skills as outcome variables was substantially higher ($\bar{g} = 0.814$), especially because the outcome variables and the intervention content were aligned. Comparisons such as these validate the direction and size of the intervention effects we identified.

Second, the positive overall effects on programming knowledge and skills extend previous systematic reviews of programming interventions which mostly summarized the types of interventions, measures, and their characteristics qualitatively (e.g., Lye & Koh, 2014; Moreno-León & Robles, 2016; Shute et al., 2017)—of course, with some exceptions (e.g., Costa and Miranda, 2017; Umapathy & Ritzhaupt, 2017; Vihavainen et al., 2014). The main contribution of the present study therefore lies in the quantifying of effect sizes across a broad range of interventions and outcome variables. Despite the positive evidence for the effectiveness of programming interventions per se, we notice that (a) these effects vary within and between primary studies; (b) these effects do *not* suggest that learning computer programming enables students to transfer the acquired knowledge and skills to non-programming domains—our meta-analysis was only concerned with the *near* transfer to similar programming tasks.

Visualization and physicality. Concerning the effectiveness of programming tools under condition (2), our meta-analysis identified moderate effects of visualization ($\bar{g} = 0.436$) and large effects for physicality ($\bar{g} = 0.718$). The hopes associated with the better effectiveness of visual languages seem to be, at least to some extent, fulfilled (Flórez et al., 2017). Visual programming languages may reduce the cognitive load associated with the reading, understanding, and creating of code and may therefore be more accessible to students than purely text-based languages (Sengupta, Dickes, Voss Farris, Martin, & Wright, 2015). At the same time, some graphical elements in languages such as Scratch may be perceived as distracting and thus draw on students' inhibition capabilities (Çakiroğlu and Suiçmez, 2018). The use of visual representations of code may also aid the creating of a mental model about the coding sequencing and the functioning of the code (Di Lieto and Inguaggiato, 2017; Tsai, 2019). As we dug deeper in the meta-analytic data, we tested whether the programming language Scratch was especially effective in primary studies focusing on visualization. The moderator analyses showed that effect sizes were significantly higher for studies using Scratch as compared to the languages (e.g., Logo). This finding substantiates Morenó-Leon's and Robles' (2016) observations of the effectiveness associated with this language. The authors proposed several explanations for this effectiveness beyond the visual nature of the language, such as the flexibility of Scratch to accommodate different types of projects allowing for different interests and learning styles or the positive impact on attitudes toward a subject or programming which may lead to better learning outcomes. From our perspective, the empirical evidence backing these explanations still needs to be developed, and the mechanisms behind the effectiveness of visualization still need to be understood. We therefore encourage researchers to examine the cognitive underpinnings of programming with different types of languages and assess students' capabilities of shifting between different types of code representations.

Concerning the effectiveness of physicality (for instance, in primary studies using Lego Mindstorms®), we believe that this tendency may be due to the immediate feedback made available to students after programming (Grover & Pea, 2013; Lee et al., 2014). This feedback is manifested in specific movements or reactions the Lego robots show after putting the computer code to action. Observing the functioning of a computer code for real or virtual objects may also aid students' motivation for engaging in coding (Afari & Khine, 2017; Tsai, 2019). The large effects seem promising from an educator's perspective, because learning programming with such experiences may not only be effective in terms of fostering students' knowledge and skills but also their motivation and engagement (Hsu et al., 2018).

Instructional approaches. We found some differences between the instructional approaches to fostering computer programming identified in our meta-analytic sample—however, these differences were statistically insignificant. On the one hand, this finding might be perceived disappointing because educational researchers would want to find evidence of what "works best". On the other hand, this finding might also be perceived promising because it shows that there are multiple ways of fostering programming knowledge and skills (Brown & Wilson, 2018). Nevertheless, some differences surfaced: Blended learning approaches showed the largest intervention effects ($\bar{g} = 1.023$). This finding may be explained by the reasoning that learning management systems, which were used in the primary studies that focused on blended learning, can offer students executable code examples, programming tutorials, educational videos, platforms to share problem solution—in other words, a wealth of learning material which may benefit students' process of learning computer programming (Flórez et al., 2017). Next, the effect of game-based interventions was slightly larger than that of others (gaming: $\bar{g} = 0.821$). Most primary studies focusing on gaming provided students with virtual life experiences and opportunities to design games. These processes, as Kafai and Burke (2013) noted, shift the focus from computer code to applications and their making. This shift may help students engage better in computer programming and thus create more effective interventions (Batista et al., 2016; Grover & Pea, 2013). We notice that the number of effect sizes focusing on blended and game-based learning were small, and the evidence presented here must be substantiated further.

Primary studies facilitating metacognitive strategies to help students learning computer programming showed a large overall effect ($\bar{g} = 0.658$), comparable to the general effects identified for metacognitive strategy instruction ($\bar{g} = 0.50$–0.63; de Boer, Donker, Kostons, & van der Werf, 2018). This finding is not too surprising: Learning computer programming is considered to be challenging, especially for novices, as it requires not only executing or adapting computer code, but also skills related to the planning and organizing of code, the monitoring of the problem-solving progress and success, self-reflection, and trouble-shooting—just to name a few (Nurulain Mohd Rum & Zolkepli, 2018; Volet & Lund, 1994). These activities seem to be easier for students who have acquired metacognitive strategies and awareness (Bernard & Bachu, 2015). In this sense, teaching programming through metacognition seems effective, and the metacognitive skills acquired during instruction may ultimately impact students' problem-solving performance and success.

The effects of collaborative activities ($\bar{g} = 0.560$) were comparable to

those identified by Umapathy and Ritzhaupt (2017), thus supporting that this approach is effective in fostering programming knowledge and skills. Nevertheless, we could not confirm an overall superiority of collaborative learning over individual learning. Thinking of computer programming as a means to engage students in problem-solving processes, adding a collaborative component to these already demanding processes can create even more burdens on students, especially in the beginning of the collaborative process where roles, perspective, and knowledge are shared (OECD, 2017). Besides, not every programming task may be feasible to be administered in collaborative settings—Siddiq and Scherer (2017) argued that collaborative tasks in ICT-rich contexts should be designed carefully to facilitate meaningful student-student interactions. All in all, we argue that collaboration per se is not more effective than the individual learning of computer programming.

Finally, problem solving instruction ($\bar{g} = 0.518$) and feedback strategies ($\bar{g} = 0.494$) showed moderate effect sizes. Without discussing this observation in large detail, we notice that these two instructional approaches have generally been reported to aid the acquisition of knowledge and skills across several domains (e.g., Azevedo & Bernard, 1995; Dochy, Segers, van den Bossche, & Gijbels, 2003). Reviewing the evidence base in our study, we observed that most of the studies conducted in the 1980s and 1990s took a problem-solving approach to computer programming by providing students with specific steps and sequences of actions to solve coding problems. As noted earlier, computer programming, as a key element of computational thinking, engages students in problem solving—thus, problem solving instruction is an obvious choice for fostering programming knowledge and skills.

4.2. Contextual variables explaining variation in the intervention effects

Next to the overall effect sizes under conditions (1)–(3), we examined possible moderation effects by study, sample, and publication features, such as the type of outcome and contextual variables. The selection of a broad range of moderators extends existing meta-analyses on the effectiveness of specific programming interventions and tools and allows researchers to explore which instructional conditions may be more or less effective. Several findings of the moderator analyses are worth discussing.

Measurement of programming knowledge and skills. We examined whether the intervention effect sizes under conditions (1) to (3) differed between measures of programming knowledge and skills and found significant differences in two instances: We observed larger effects of studies assessing programming knowledge in the study condition 1 and smaller effects for studies taking a collaborative teaching approach to programming instruction in the study condition 3. The latter finding aligns with the tendency of collaborative instruction to be especially effective for skill acquisition in computer-based learning environments as compared to knowledge acquisition (Graesser et al., 2018). Student-student interactions may be aiding the development of skills but may depend on the knowledge students have acquired before engaging in a problem-solving process (Siddiq & Scherer, 2017). At the same time, knowledge sharing is key to collaboration—this sharing, however, does not ensure that all students who collaborate will acquire the shared knowledge to the same extent. The finding that no other moderating effects occurred was surprising, given the considerable diversity of these measures, as noted by Shute et al. (2017) in their review. At the same time, the intervention programs seemed to be equally effective in fostering both outcome variables—we consider this to be a promising result because it shows that both knowledge and skills in the programming domain could be fostered to a similar extent. Moreover, the inclusion of skills beyond knowledge measures in the primary studies resonates with Flórez et al.'s (2017) plea to focus not only on the pure knowledge of and about computer code but the very skills surrounding it. The challenge, however, remains to clearly define these skills, especially with respect to taking computational perspectives (Grover & Pea, 2013; Lye & Koh, 2014).

Notably, our meta-analysis did not reveal consistent differences in effects between standardized than for unstandardized tests. In fact, in only one category (study condition 3, collaboration), standardized assessments of programming knowledge tended to result in larger effects than assessments that were not standardized. To summarize, we neither found consistent evidence for the differential effectiveness of programming interventions between knowledge and skills not between standardized and non-standardized tests.

Sample, study, and publication features. Examining the moderation effects of study features, we would like to highlight one finding: First, there was no evidence for the differences in effects between pretest-posttest and posttest-only control group designs—in contrast to the general observations noted by Cheung and Slavin (2016). To some extent, this finding suggests that the method bias caused by different study designs may only be limited in our meta-analysis. Second, we observed a tendency of negative moderation effects by the intervention length. Longer interventions under condition (2) focusing on physicality tended to be less effective than shorter interventions. Although this result may be counter-intuitive, it is not unusual in the domain of educational technology. For instance, Chauhan (2017) found that interventions of elementary students' learning with technology that exceeded 6 months were less effective than short-term interventions. Sung, Chang, and Liu (2016) expected the duration of interventions that examined the effects of integrating mobile devices on students' learning to be positive—however, the authors did not find any significant moderation effect of intervention length in their meta-analysis. The negative relation we found in our meta-analysis may have several reasons, such as the high value of novelty (i.e., when learning programming with a new tool or language) or the higher engagement of students in the intervention tasks in the beginning of the interventions (e.g., Cheung & Slavin, 2013). Third, some differences between continents surfaced: for instance, the largest intervention effects for studies examining visualization or physicality were reported by studies involving Asian student samples. Although our meta-analysis cannot provide substantive explanations for these differences, we suspect that several factors may have contributed to this moderation effect. We believe that the countries' development of information and communication technology (ITU, 2017), the openness to these technologies and their advancements (OECD, 2013), and the curricular emphasis on skills related to ICT and computer programming (UNESCO Institute for Statistics, 2018) may be among the possible, explanatory factors.

Finally, we would like to highlight two more moderation effects: For the studies focusing on problem solving instruction, programming as an extracurricular activity was more effective than as a part of regular lessons. One of the reasons for this difference may lie in the presumably enhanced motivation of students enrolling in extracurricular activities, posing a selection effect on the effectiveness of the intervention (e.g., Durlak, Weissberg, & Pachan, 2010). Although the conclusion that extracurricular activities may generally be more effective in fostering programming knowledge and skills may be tempting, the primary studies did not allow us to estimate long-term effects—such effects could clarify the observed differences.

Studies focusing on metacognition instruction were more effective if they engaged students in collaboration rather than having students work on programming tasks individually. This observation is not unknown to studies in the field of computer programming, and evidence exists supporting it (Umapathy & Ritzhaupt, 2017). Bernard and Bachu (2015) argue that collaborative settings (e.g., pair programming) allow students to explain their thinking, reflect on their strategies, and correct each other if needed. The benefits of collaboration lie therefore in the promotion of metacognitive strategies and awareness.

In sum, only few study, sample, and publication features explained the variation of effects within and between studies and thus indicated only a marginal degree of differential effectiveness.

4.3. Limitations and future directions

The current meta-analysis has some limitations worth noting: First, we categorized the primary studies under condition (3) by their instructional approaches to computer programming using broad categories. Alternative, more fine-grained categorizations may provide more detailed information about the differential effectiveness of these approaches, although the available sample sizes in each category may limit the level of granularity. We therefore encourage researchers to explore different categorizations. Second, as most studies implemented a posttest-only design, adjustments for students' prior programming skills were largely missing. We believe that pretest-posttest intervention designs will draw a more accurate picture of the actual intervention effects (Shadish, Cook, & Campbell, 2002), and we encourage researchers who study the effectiveness of programming interventions to implement such rigorous research designs (e.g., Tsai, 2019). Third, the measures of programming skills and knowledge were diverse—this diversity may have contributed to the between-study variation in the intervention effects. The lack of uniform outcome measures in primary studies is a current challenge for effectiveness studies of programming interventions (Shute et al., 2017). Fourth, the success of programming interventions may depend on factors other than the ones examined through moderator analyses. For instance, Tsai (2019) showed that their visual programming intervention was more successful for students with low to moderate self-efficacy. Consequently, explaining the mechanisms behind the effectiveness of certain interventions requires considering such motivational and attitudinal factors.

4.4. Conclusions and implications

The present meta-analysis set out to examine the empirical evidence surrounding the effectiveness of computer programming instruction per se (study condition 1), the effectiveness of visualization and physicality (study condition 2), and the effectiveness of instructional approaches (study condition 3) for fostering students' programming knowledge and skills. Several findings surfaced that inform research in this area: First, the strong positive effect size of programming instruction per se was expected ($\bar{g} = 0.814$) and suggests the trainability of programming knowledge and skills. This report of an overall effect size provides a reference point against which future interventions and their effect sizes could be evaluated (see also Chen et al., 2018; Tsai & Tsai, 2018).

Second, the meta-analysis of studies focusing on visualization or physicality identified positive overall effect sizes (visualization: $\bar{g} = 0.436$, physicality: $\bar{g} = 0.718$) and supported some of the existing claims surrounding the effectiveness of modern programming languages such as Scratch. Surprisingly, visualization was differentially effective across student samples. Hence, we argue that the context of the instruction and the change in representation modes of programming languages should be considered carefully for the specific groups of students (see also Sáez-López, Román-González, & Vázquez-Cano, 2016). Similarly, programming tools involving physicality showed large effect sizes than non-physical ones; once again, although these tendencies may suggest that these languages and tools may have delivered on their promises, they need to be backed with further empirical evidence obtained from carefully designed experimental studies.

Third, apart from these conclusions that may primarily have a scientific merit as they generate knowledge about the overall effectiveness of interventions, the subsequent moderator analyses may have more practical implications: We did not find evidence for the superiority of specific interventions, such as collaborative activities, feedback-based instruction, or game-based learning in programming. Hence, the claims surrounding their superiority among alternative instructional approaches could not be fully substantiated for the present, meta-analytic samples. Educators may therefore choose among them without losing out on effectiveness yet with considering the suitability for the specific group of students they are teaching. At the same time, the evidence on the age specificity of some intervention effects supports the current attempts to design programming interventions with tools that are customized for the different age groups of students. The finding that instructional practices did not differ significantly in their effect sizes may be a relief to many researchers and educators: First, they may not have to restrict their teaching to a specific approach but provide students with different learning experiences and teaching practices. Second, they may adapt their instructional approaches to the specific circumstances and conditions of the learning environments and students without deteriorating the success of the programming instruction. At the same time, monitoring continuously what works best for learning programming should become an integral part of the research agenda in this field. Moreover, the fact that key study design features (e.g., randomization, standardized testing) moderated some of the effects emphasizes the importance of well-designed experimental studies in order to reduce the possible methodological bias in effect sizes (see also Mayer, 2015). We therefore encourage researchers to continue systematically investigating the effectiveness of various instructional approaches and conditions in well-designed experimental studies and to strive for developing interventions, tools, and features that make computer programming accessible to students at different stages of their education.

Acknowledgement

This research was supported by the FINNUT Young Research Talent Project Grant (NFR-254744 "ADAPT21") awarded to Ronny Scherer by The Research Council of Norway.

Appendix A. Supplementary data

Supplementary data to this article can be found online at https://doi.org/10.1016/j.chb.2020.106349.

References[3]

Adams, R. J., Smart, P., & Huff, A. S. (2017). Shades of grey: Guidelines for working with the grey literature in systematic reviews for management and organizational studies. *International Journal of Management Reviews, 19*(4), 432–454. https://doi.org/10.1111/ijmr.12102.

Afari, E., & Khine, M. S. (2017). Robotics as an educational tool: Impact of Lego Mindstorms. *International Journal of Information and Education Technology, 7*(6), 437–442. https://doi.org/10.18178/ijiet.2017.7.6.908.

* Altintas, T., Gunes, A., & Sayan, H. (2016). A peer-assisted learning experience in computer programming language learning and developing computer programming skills. *Innovations in Education & Teaching International, 53*(3), 329–337. https://doi.org/10.1080/14703297.2014.993418.

Atkinson, L. Z., & Cipriani, A. (2018). How to carry out a literature search for a systematic review: A practical guide. *BJPsych Advances, 24*, 74–82. https://doi.org/10.1192/bja.2017.3.

Au, W. L. (1992). *Logo programming: Instructional methods and problem solving (Doctoral dissertation)*. New Zealand: Massey University. Retrieved from http://hdl.handle.net/10179/4122.

Azevedo, R., & Bernard, R. M. (1995). A meta-analysis of the effects of feedback in computer-based instruction. *Journal of Educational Computing Research, 13*(2), 111–127. https://doi.org/10.2190/9LMD-3U28-3A0G-FTQT.

Balanskat, A., & Engelhardt, K. (2015). *October). Computing our future. Computer programming and coding: Priorities, school curricula and initiatives across europe. Brussels: European schoolnet.* Retrieved from https://www.researchgate.net/publication/284139559_Computing_our_future_Computer_programming_and_coding_-_Priorities_school_curricula_and_initiatives_across_Europe. (Accessed 20 January 2020).

* Barak, M., Harward, J., Kocur, G., & Lerman, S. (2007). Transforming an introductory programming course: From lectures to active learning via wireless laptops. *Journal of Science Education and Technology, 16*(4), 325–336. https://doi.org/10.1007/s10956-007-9055-5.

Batista, A. L. F., Connolly, T., Angotti, J. A., & P.. (2016, October). A framework for games-based construction learning: A text-based programming languages approach. Paper presented at the 10th European conference on games based learning, at paisley, UK. Retrieved from https://search.proquest.com/openview/8a1755d985

[3] References marked with an asterisk (*) indicate studies included in the meta-analysis.

b3545571f3101a9fe45b24/1.pdf?pq-origsite=gscholar&cbl=396495. (Accessed 30 July 2018).
Bernard, M., & Bachu, E. (2015). Enhancing the metacognitive skill of novice programmers through collaborative learning. In A. Peña-Ayala (Ed.), *Metacognition: Fundaments, applications, and trends* (pp. 277–298). Cham: Springer. https://doi.org/10.1007/978-3-319-11062-2_11.
Bernard, R. M., Borokhovski, E., Schmid, R. F., Tamim, R. M., & Abrami, P. C. (2014). A meta-analysis of blended learning and technology use in higher education: From the general to the applied. *Journal of Computing in Higher Education, 26*(1), 87–122. https://doi.org/10.1007/s12528-013-9077-3.
de Boer, H., Donker, A. S., Kostons, D. D. N. M., & van der Werf, G. P. C. (2018). Long-term effects of metacognitive strategy instruction on student academic performance: A meta-analysis. *Educational Research Review, 24*, 98–115. https://doi.org/10.1016/j.edurev.2018.03.002.
Borenstein, M., Hedges, L. V., Higgins, J. P., & Rothstein, H. R. (2009). *Introduction to meta-analysis*. Chichester, West Sussex: John Wiley & Sons, Ltd.
Brennan, K., & Resnick, M. (2012, April). New frameworks for studying and assessing the development of computational thinking. Paper presented at the Annual Meeting of the American Educational Research Association, Vancouver, Canada. Retrieved from https://web.media.mit.edu/~kbrennan/files/Brennan_Resnick_AERA2012_CT.pdf. (Accessed 31 July 2018).
Brown, N. C. C., & Wilson, G. (2018). Ten quick tips for teaching programming. *PLoS Computational Biology, 14*(4), 1–8. https://doi.org/10.1371/journal.pcbi.1006023.
Çakiroğlu, Ü., Suiçmez, S. S., et al. (2018). Exploring perceived cognitive load in learning programming via Scratch. *Research in Learning Technology*. https://doi.org/10.25304/rlt.v26.1888.
Card, N. A. (2012). *Applied meta-analysis for social science research*. New York, NY: The Guilford Press.
* Carney, R. W. (2000). An evaluation of a method for teaching the transfer of fundamental computer programming statements between QBASIC and four other computer programming languages, 9991563 Ed.D.. In *Wilmington college (Delaware), ann arbor*. ProQuest Dissertations & Theses A&I database.
* Cetin, I. (2016). Preservice teachers' introduction to computing: Exploring utilization of Scratch. *Journal of Educational Computing Research, 54*(7), 997–1021. https://doi.org/10.1177/0735633116642774.
* Chao, J. (1999). *Effects of structured teaching method on students' understanding of angle and rotation in Logo geometry*, 9923528 Ph.D. Ann Arbor: ProQuest Dissertations & Theses A&I database. Arizona State University.
Chauhan, S. (2017). A meta-analysis of the impact of technology on learning effectiveness of elementary students. *Computers & Education, 105*, 14–30. https://doi.org/10.1016/j.compedu.2016.11.005.
Cheng, C., Cheung, M. W.-L., & Wang, H.-y. (2018). Multinational comparison of internet gaming disorder and psychological problems versus well-being: Meta-analysis of 20 countries. *Computers in Human Behavior, 88*, 153–167. https://doi.org/10.1016/j.chb.2018.06.033.
Chen, J., Wang, M., Kischner, P., & Tsai, C.-C. (2018). The role of collaboration, computer use, learning environments, and supporting strategies in CSCL: A meta-analysis. *Review of Educational Research, 88*(6), 799–843. https://doi.org/10.3102/0034654318791584.
Cheung, M. W.-L. (2014). Modeling dependent effect sizes with three-level meta-analyses: A structural equation modeling approach. *Psychological Methods, 19*(2), 211–229. https://doi.org/10.1037/a0032968.
Cheung, M. W.-L. (2015). *Meta-analysis: A structural equation modeling approach*. Chichester, West Sussex: John Wiley & Sons, Ltd.
Cheung, A. C. K., & Slavin, R. E. (2013). The effectiveness of educational technology applications for enhancing mathematics achievement in K-12 classrooms: A meta-analysis. *Educational Research Review, 9*, 88–113. https://doi.org/10.1016/j.edurev.2013.01.001.
Cheung, A. C. K., & Slavin, R. E. (2016). How methodological features affect effect sizes in education. *Educational Researcher, 45*(5), 283–292. https://doi.org/10.3102/0013189X16656615.
Clements, D. H. (1995). Teaching creativity with computers. *Educational Psychology Review, 7*(2), 141–161. https://doi.org/10.1007/BF02212491.
Clements, D. H., & Sarama, J. (1997). Research on Logo. *Computers in the Schools, 14*(1–2), 9–46. https://doi.org/10.1300/J025v14n01_02.
Costa, J. M., & Miranda, G. L. (2017). Relation between Alice software and programming learning: A systematic review of the literature and meta-analysis. *British Journal of Educational Technology, 48*(6), 1464–1474. https://doi.org/10.1111/bjet.12496.
* Daly, T. (2013). *Influence of Alice 3: Reducing the hurdles to success in a CS1 programming course*. Dissertations/Theses - Doctoral Dissertations. University of North Texas. Available from: Ovid Technologies (3579195).
Denning, P. J. (2017). Remaining trouble spots with computational thinking. *Communications of the ACM, 60*(6), 33–39. https://doi.org/10.1145/2998438.
* Denny, P., Cukierman, D., & Bhaskar, J. (2015, November). Measuring the effect of inventing practice exercises on learning in an introductory programming course. Paper presented at the Koli Calling Conference on Computing Education Research, Koli, Finland https://doi.org/10.1145/2828959.2828967.
Di Lieto, M. C., Inguaggiato, E., et al. (2017). Educational robotics intervention on executive functions in preschool children: A pilot study. *Computers in Human Behavior, 71*, 16–23. https://doi.org/10.1016/j.chb.2017.01.018.
Dochy, F., Segers, M., van den Bossche, P., & Gijbels, D. (2003). Effects of problem-based learning: A meta-analysis. *Learning and Instruction, 13*, 533–568. https://doi.org/10.1016/S0959-4752(02)00025-7.
Durlak, J. A., Weissberg, R. P., & Pachan, M. (2010). A meta-analysis of after-school programs that seek to promote personal and social skills in children and adolescents. *American Journal of Community Psychology, 45*(3–4), 294–309. https://doi.org/10.1007/s10464-010-9300-6.
Duval, S., & Tweedie, R. (2000). Trim and fill: A simple funnel-plot–based method of testing and adjusting for publication bias in meta-analysis. *Biometrics, 56*(2), 455–463. https://doi.org/10.1111/j.0006-341X.2000.00455.x.
Egger, M., Smith, G. D., Schneider, M., & Minder, C. (1997). Bias in meta-analysis detected by a simple, graphical test. *BMJ, 315*(7109), 629–634. https://doi.org/10.1136/bmj.315.7109.629.
European Commission. (2016). Coding and computational thinking on the curriculum. Retrieved from https://ec.europa.eu/education/sites/education/files/2016-pla-coding-computational-thinking_en.pdf. (Accessed 30 July 2018).
Flórez, F. B., Casallas, R., Hernández, M., Reyes, A., Restrepo, S., & Danies, G. (2017). Changing a generation's way of thinking: Teaching computational thinking through programming. *Review of Educational Research, 87*(4), 834–860. https://doi.org/10.3102/0034654317710096.
Forsström, S. E., & Kaufmann, O. T. (2018). A literature review exploring the use of programming in mathematics education. *International Journal of Learning, Teaching and Educational Research, 17*(12), 18–32. https://doi.org/10.26803/ijlter.17.12.2.
Garneli, V., Giannakos, M. N., & Chorianopoulos, K. (2015, March). Computing education in K-12 schools: A review of the literature. IEEE global engineering education conference (EDUCON), Tallinn, Estonia. https://doi.org/10.1109/EDUCON.2015.7096023.
Graesser, A. C., Fiore, S. M., Greiff, S., Andrews-Todd, J., Foltz, P. W., & Hesse, F. W. (2018). Advancing the science of collaborative problem solving. *Psychological Science in the Public Interest, 19*(2), 59–92. https://doi.org/10.1177/1529100618808244.
Grover, S., & Pea, R. (2013). Computational thinking in K-12: A review of the state of the field. *Educational Researcher, 42*(1), 38–43. https://doi.org/10.3102/0013189x12463051.
* Grover, S., Pea, R., & Cooper, S. (2015). Designing for deeper learning in a blended computer science course for middle school students. *Computer Science Education, 25*(2), 199–237. https://doi.org/10.1080/08993408.2015.1033142.
Guzdial, M. (2014). Anyone can learn programming: Teaching > Genetics. Communications of the ACM [Blog post]. Retrieved from https://cacm.acm.org/blogs/blog-cacm/179347-anyone-can-learn-programming-teaching-genetics/fulltext. (Accessed 2 August 2018). October 14.
Haddaway, N. R., Collins, A. M., Coughlin, D., & Kirk, S. (2015). The role of Google Scholar in evidence reviews and its applicability to grey literature searching. *PloS One, 10*(9), e0138237. https://doi.org/10.1371/journal.pone.0138237.
Hsu, T.-C., Chang, S.-C., & Hung, Y.-T. (2018). How to learn and how to teach computational thinking: Suggestions based on a review of the literature. *Computers & Education, 126*, 296–310. https://doi.org/10.1016/j.compedu.2018.07.004.
* Huang, K. H., Yang, T. M., & Cheng, C. C. (2013). Engineering to see and move: Teaching computer programming with flowcharts vs. LEGO robots. *International Journal of Emerging Technologies in Learning (iJET), 8*(4), 23–26. https://doi.org/10.3991/ijet.v8i4.2943.
International Telecommunication Union (ITU). (2017). *Measuring the information society report* (Vol. 1). Geneva: International Telecommunication Union (ITU). Retrieved from: https://www.itu.int/en/ITU-D/Statistics/Documents/publications/misr2017/MISR2017_Volume1.pdf. (Accessed 15 November 2018).
Jackson, D., & Turner, R. (2017). Power analysis for random-effects meta-analysis. *Research Synthesis Methods, 8*(3), 290–302. https://doi.org/10.1002/jrsm.1240.
* Jehng, J.-C. J., & Chan, T.-W. (1998). Designing computer support for collaborative visual learning in the domain of computer programming. *Computers in Human Behavior, 14*(3), 429–448. https://doi.org/10.1016/S0747-5632(98)00015-6.
* Jenkins, C. (2015). Poem generator: A comparative quantitative evaluation of a microworlds-based learning approach for teaching English. *International Journal of Education and Development Using Information and Communication Technology, 11*(2), 153–167. Retrieved from http://ijedict.dec.uwi.edu/viewarticle.php?id=1972. (Accessed 2 August 2018).
* Johnson, J., & Kane, K. (1992). Developmental and task factors in Logo programming. *Journal of Educational Computing Research, 8*(2), 229–253. https://doi.org/10.2190/992T-JJDQ-TGX2-04P5.
Kafai, Y. B., & Burke, Q. (2013). Computer programming goes back to school. *Phi Delta Kappan, 95*(1), 61–65. https://doi.org/10.1177/003172171309500111.
Kafai, Y. B., & Burke, Q. (2015). Constructionist gaming: Understanding the benefits of making games for learning. *Educational Psychologist, 50*(4), 313–344. https://doi.org/10.1080/00461520.2015.1124022.
* Lai, H., & Xin, W. (2011, December). An experimental research of the pair programming in Java programming course. Paper presented at the International Conference on e-Education, Entertainment and e-Management (ICEEE), Bali, Indonesia https://doi.org/10.1109/ICeEEM.2011.6137800.
Lee, W.-C. (1990). The effectiveness of computer-assisted instruction and computer programming in elementary and secondary mathematics: A meta-analysis. Doctoral dissertation (ProQuest No. AAI9022709), university of Massachusetts, Amherst, MA. Retrieved from https://scholarworks.umass.edu/dissertations/AAI9022709/.
Lee, M. O. C. (1991). Guided instruction with Logo programming and the development of cognitive monitoring strategies among college students. Doctoral Dissertation (ProQuest No. 62897250), University of Massachusetts, Amherst, MA. Retrieved from https://search.proquest.com/docview/62897250?accountid=14699.
Lee, Y. Y., Mauriello, M. L., Ahn, J., & Bederson, B. B. (2014). CTArcade: Computational thinking with games in school age children. *International Journal of Child-Computer Interaction, 2*, 26–33. https://doi.org/10.1016/j.ijcci.2014.06.003.
* Lehrer, R., Lee, M., & Jeong, A. (1999). Reflective teaching of Logo. *The Journal of the Learning Sciences, 8*(2), 245–289. https://doi.org/10.1207/s15327809jls0802_3.

Liao, Y.-K. C., & Bright, G. W. (1991). Effects of computer programming on cognitive outcomes: A meta-analysis. *Journal of Educational Computing Research, 7*(3), 251–268. https://doi.org/10.2190/e53g-hh8k-ajrr-k69m.

Liao, Y.-k. C. (2000). A meta-analysis of computer programming on cognitive outcomes: An updated synthesis. In *Paper presented at the Proceedings of World Conference on Educational Multimedia*. Montreal, Canada: Hypermedia and Telecommunications.

Li, Q., & Ma, X. (2010). A meta-analysis of the effects of computer technology on school students' mathematics learning. *Educational Psychology Review, 22*, 215–243. https://doi.org/10.1007/s10648-010-9125-8.

Lipsey, M. W., & Wilson, D. (2001). *Practical meta-analysis*. Thousand Oaks, CA: Sage.

Lito, A. T. (2017). Robotics interventions for improving educational outcomes—a meta-analysis. University of Ioánnina, School of Education, Greece. Retrieved from http://olympias.lib.uoi.gr/jspui/bitstream/123456789/28575/1/%CE%9C.%CE%95.%20%CE%91%CE%98%CE%91%CE%9D%CE%91%CE%A3%CE%99%CE%9F%CE%A5%20%CE%9B%CE%97%CE%A4%CE%A9%202017.pdf.

* Liu, A. S., Schunn, C. D., Flot, J., & Shoop, R. (2013). The role of physicality in rich programming environments. *Computer Science Education, 23*(4), 315–331. https://doi.org/10.1080/08993408.2013.847165.

Lou, Y., Abrami, P. C., & d'Apollonia, S. (2001). Small group and individual learning with technology: A meta-analysis. *Review of Educational Research, 71*(3), 449–521. https://doi.org/10.3102/00346543071003449.

Lye, S. Y., & Koh, J. H. L. (2014). Review on teaching and learning of computational thinking through programming: What is next for K-12? *Computers in Human Behavior, 41*, 51–61. https://doi.org/10.1016/j.chb.2014.09.012.

Macnamara, B. N., Hambrick, D. Z., & Oswald, F. L. (2014). Deliberate practice and performance in music, games, sports, education, and professions: A meta-analysis. *Psychological Science, 25*(8), 1608–1618. https://doi.org/10.1177/0956797614535810.

* Mason, R., & Cooper, G. (2013). Mindstorms robots and the application of cognitive load theory in introdictory programming. *Computer Science Education, 23*(4), 296–314. https://doi.org/10.1080/08993408.2013.847152.

Mayer, R. E. (2015). On the need for research evidence to guide the design of computer games for learning. *Educational Psychologist, 50*(4), 349–353. https://doi.org/10.1080/00461520.2015.1133307.

* Milner, S. (1973). The effects of computer programming on performance in mathematics. In *Paper presented at the annual meeting of the American educational research association* (New Orleans, Louisiana).

Moher, D., Liberati, A., Tetzlaff, J., Altman, D. G., The Prisma Group, et al. (2009). Preferred reporting items for systematic reviews and meta-analyses: The PRISMA statement. *PLoS Medicine, 6*(7), 1–6. https://doi.org/10.1371/journal.pmed.1000097.

Moreno-León, J., & Robles, G. (2016). April). Code to learn with Scratch? A systematic literature review. IEEE global engineering education conference (EDUCON), Abu Dhabi, United Arab Emirates. https://doi.org/10.1109/EDUCON.2016.7474546.

Morris, S. B. (2008). Estimating effect sizes from pretest-posttest-control group designs. *Organizational Research Methods, 11*(2), 364–386. https://doi.org/10.1177/1094428106291059.

* Nugent, G., Barker, B., Grandgenett, N., & Adamchuk, V. I. (2010). Impact of robotics and geospatial technology interventions on youth STEM learning and attitudes. *Journal of Research on Technology in Education, 42*(4), 391–408. https://doi.org/10.1080/15391523.2010.10782557.

Nurulain Mohd Rum, S., & Zolkepli, M. (2018). Metacognitive strategies in teaching and learning computer programming. *International Journal of Engineering & Technology, 7*, 788–794. https://doi.org/10.14419/ijet.v7i4.38.27546.

OECD. (2013). PISA 2012 results: Ready to learn: Students' engagement, drive and self-beliefs (volume III). Paris. OECD Publishing https://doi.org/10.1787/9789264201170-en.

OECD. (2017). *PISA 2015 results: Collaborative problem solving* (Vol. V). Paris: OECD Publishing. https://doi.org/10.1787/9789264285521-en.

* Olelewe, C. J., & Agomuo, E. E. (2016). Effects of B-learning and F2F learning environments on students' achievement in QBASIC programming. *Computers & Education, 103*, 76–86. https://doi.org/10.1016/j.compedu.2016.09.012.

* Oprea, J. M. (1984). *The effects of computer programming on a student's mathematical generalization and understanding of variables*, 8504061 Ph.D. Ann Arbor: ProQuest Dissertations & Theses A&I database. The Ohio State University.

Palumbo, D. B. (1990). programming language/problem-solving research: A review of relevant issues. *Review of Educational Research, 60*(1), 65–89. https://doi.org/10.3102/00346543060001065.

* Psycharis, S., & Kallia, M. (2017). The effects of computer programming on high school students' reasoning skills and mathematical self-efficacy and problem solving. *Instructional Science, 45*(5), 583–602. https://doi.org/10.1007/s11251-017-9421-5.

Robins, A., Rountree, J., & Rountree, A. (2003). Learning and teaching programming: A review and discussion. *Computer Science Education, 13*(2), 137–172. https://doi.org/10.1076/csed.13.2.137.14200.

* Rodríguez Corral, J. M., Civit Balcells, A., Morgado Estévez, A., Jiménez Moreno, G., & Ferreiro Ramos, M. J. (2014). A game-based approach to the teaching of object-oriented programming languages. *Computers & Education, 73*, 83–92. https://doi.org/10.1016/j.compedu.2013.12.013.

* Rodríguez Corral, J. M., Civit, A., Perez-Peña, F., & Molina, D. (2016). Application of robot programming to the teaching of object-oriented computer languages. *International Journal of Engineering Education, 32*(4), 1823–1832.

Sáez-López, J.-M., Román-González, M., & Vázquez-Cano, E. (2016). Visual programming languages integrated across the curriculum in elementary school: A two year case study using "Scratch" in five schools. *Computers & Education, 97*, 129–141. https://doi.org/10.1016/j.compedu.2016.03.003.

Scherer, R. (2016). Learning from the past – the need for empirical evidence on the transfer effects of computer programming skills. *Frontiers in Psychology, 7*(1390). https://doi.org/10.3389/fpsyg.2016.01390.

Scherer, R., Siddiq, F., & Sánchez Viveros, B. (2019). The cognitive benefits of learning computer programming: A meta-analysis of transfer effects. *Journal of Educational Psychology, 111*(5), 764–792. https://doi.org/10.1037/edu0000314.

Schmidt, F. L., & Hunter, J. E. (2014). *Methods of meta-analysis: Correcting error and bias in research findings* (3 ed.). Thousand Oaks, CA: Sage.

Schmucker, C. M., Blümle, A., Schell, L. K., Schwarzer, G., Oeller, P.Cabrera, L., … (2017). Systematic review finds that study data not published in full text articles have unclear impact on meta-analyses results in medical research. on behalf of the, O. c *PloS One, 12*(4), 1–16. https://doi.org/10.1371/journal.pone.0176210.

Sengupta, P., Dickes, A., Voss Farris, A., Martin, D., & Wright, M. (2015). Programming in K-12 science classrooms. *Communications of the ACM, 58*(11), 34–35. https://doi.org/10.1145/2822517.

* Shadiev, R., Hwang, W.-Y., Yeh, S.-C., Yang, S. J. H., Wang, J.-L., Han, L., et al. (2014). Effects of unidirectional vs. Reciprocal teaching strategies on web-based computer programming learning. *Journal of Educational Computing Research, 50*(1), 67–95. https://doi.org/10.2190/EC.50.1.d.

Shadish, W. R., Cook, T. D., & Campbell, D. T. (2002). *Experimental and quasi-experimental designs for generalized causal inference*. Boston, MA: Houghton Mifflin.

Shute, V. J., Sun, C., & Asbell-Clarke, J. (2017). Demystifying computational thinking. *Educational Research Review, 22*, 142–158. https://doi.org/10.1016/j.edurev.2017.09.003.

* Shyr, W.-J. (2010). Multiprog virtual laboratory applied to PLC programming learning. *European Journal of Engineering Education, 35*(5), 573–583. https://doi.org/10.1080/03043797.2010.497550.

Siddiq, F., & Scherer, R. (2017). Revealing the processes of students' interaction with a novel collaborative problem solving task: An in-depth analysis of think-aloud protocols. *Computers in Human Behavior, 76*, 509–525. https://doi.org/10.1016/j.chb.2017.08.007.

Simonsohn, U., Nelson, L. D., & Simmons, J. P. (2014). P-curve: A key to the file-Drawer. *Journal of Experimental Psychology: General, 143*(2), 534–547. https://doi.org/10.1037/a0033242.

Simonsohn, U., Nelson, L. D., & Simmons, J. P. (2017). P-curve *online App version 4.06*. http://www.p-curve.com/app4/.

* Siozou, S., Tselios, N., & Komis, V. (2008). Effect of algorithms' multiple representations in the context of programming education. *Interactive Technology and Smart Education, 5*(4), 230–243.

Springer, L., Stanne, M. E., & Donovan, S. S. (1999). Effects of small-group learning on undergraduates in science, mathematics, engineering, and technology: A meta-analysis. *Review of Educational Research, 69*(1), 21–51. https://doi.org/10.3102/00346543069001021.

Sung, Y.-T., Chang, K.-E., & Liu, T.-C. (2016). The effects of integrating mobile devices with teaching and learning on students' learning performance: A meta-analysis and research synthesis. *Computers & Education, 94*, 252–275. https://doi.org/10.1016/j.compedu.2015.11.008.

* Suomala, J., & Alajaaski, J. (2002). Pupils' problem-solving processes in a complex computerized learning environment. *Journal of Educational Computing Research, 26*(2), 155–176. https://doi.org/10.2190/58XD-NMFK-DL5V-0B6N.

Tsai, C.-Y. (2019). Improving students' understanding of basic programming concepts through visual programming language: The role of self-efficacy. *Computers in Human Behavior, 95*, 224–232. https://doi.org/10.1016/j.chb.2018.11.038.

Tsai, Y.-L., & Tsai, C.-C. (2018). Digital game-based second-language vocabulary learning and conditions of research designs: A meta-analysis study. *Computers & Education, 125*, 345–357. https://doi.org/10.1016/j.compedu.2018.06.020.

Umapathy, K., & Ritzhaupt, A. D. (2017). A meta-analysis of pair-programming in computer programming courses: Implications for educational *practice. ACM Transactions on Computing Education, 17*(4), 1–13. https://doi.org/10.1145/2996201.

UNESCO Institute for Statistics. (2018). A global framework of reference on digital literacy skills for indicator 4.4.2. Information Paper No. 51 (UIS/2018/ICT/IP/51). Montréal, Canada, QC. UNESCO Institute for Statistics. Retrieved from: http://uis.unesco.org/sites/default/files/documents/ip51-global-framework-reference-digital-literacy-skills-2018-en.pdf. (Accessed 15 November 2018).

* Uysal, M. P. (2014). Improving first computer programming experiences: The case of adapting a web-supported and well-structured problem-solving method to a traditional course. *Contemporary Educational Technology, 5*(3), 198–217.

Valentine, J. C. (2019). Incorporating judgments about study quality into research syntheses. In H. Cooper, L. V. Hedges, & J. C. Valentine (Eds.), *The handbook of research syntheses and meta-analyses* (3rd ed., pp. 129–140). New York, NY: Russell Sage Foundation.

Valentine, J. C., Pigott, T. D., & Rothstein, H. R. (2010). How many studies do you need?: A primer on statistical power for meta-analysis. *Journal of Educational and Behavioral Statistics, 35*(2), 215–247. https://doi.org/10.3102/1076998609346961.

Viechtbauer, W. (2005). Bias and efficiency of meta-analytic variance estimators in the random-effects model. *Journal of Educational and Behavioral Statistics, 30*(3), 261–293. https://doi.org/10.3102/10769986030003261.

Viechtbauer, W. (2017). *Metafor: Meta-Analysis package for R. R package version 2.0-0*.

Viechtbauer, W., & Cheung, M. W.-L. (2010). Outlier and influence diagnostics for meta-analysis. *Research Synthesis Methods, 1*(2), 112–125. https://doi.org/10.1002/jrsm.11.

Vihavainen, A., Airaksinen, J., & Watson, J. (2014, August). A systematic review of approaches for teaching introductory programming and their influence on success. Proceedings of the Tenth Annual conference on international computing education research (ICER), glasgow, UK. https://doi.org/10.1145/2632320.2632349.

* Volet, S., & Lund, C. (1994). Metacognitive instruction in introductory computer programming: A better explanatory construct for performance than traditional factors. *Journal of Educational Computing Research, 10*(4), 297–328. https://doi.org/10.2190/9A08-Y2Q0-6AER-6KLQ.

Wing, J. M. (2006). Computational thinking. *Communications of the ACM, 49*(3), 33–35. https://doi.org/10.1145/1118178.1118215.

Yadav, A., Good, J., Voogt, J., & Fisser, P. (2017). Computational thinking as an emerging competence domain. In M. Mulder (Ed.), *Competence-based vocational and professional education: Bridging the worlds of work and education* (pp. 1051–1067). Cham: Springer International Publishing.

* Yang, T.-C., Hwang, G.-J., Yang, S. J. H., & Hwang, G.-H. (2015). A two-Tier test-based approach to improving students' computer-programming skills in a web-based learning environment. *Educational Technology & Society, 18*(1), 198–210.

Young, J. (2017). Technology-enhanced mathematics instruction: A second-order meta-analysis of 30 years of research. *Educational Research Review, 22*, 19–33. https://doi.org/10.1016/j.edurev.2017.07.001.

* Yüksel, H., & Yüksel, A. (2015). The effect of the computer assisted instruction on the academic achievement and retention of technical programme students' in vocational foreign language. *Procedia—Social and Behavioral Sciences, 174*, 2513–2518. https://doi.org/10.1016/j.sbspro.2015.01.924.