

Visual Programming as a Tool for Developing Knowledge in STEM Subjects : A Literature Review

Karin Stolpe and Jonas Hallström

Book Chapter

Cite this chapter as:

Stolpe, K., Hallström, J. Visual Programming as a Tool for Developing Knowledge in STEM Subjects : A Literature Review, In Hallström, J., J., M. (eds), Programming and Computational Thinking in Technology Education: Swedish and International Perspectives, : Brill Academic Publishers; 2023, pp. 130-169. ISBN: 978-90-04-68791-2

DOI: https://doi.org/10.1163/9789004687912_007

International Technology Education Studies, , No. 20

Copyright: Brill Academic Publishers

<https://brill.com/>

The self-archived postprint version of this book chapter is available at Linköping University Institutional Repository (DiVA):

<https://urn.kb.se/resolve?urn=urn:nbn:se:liu:diva-198608>

Visual Programming as a Tool for Developing Knowledge in STEM Subjects

A Literature Review

Karin Stolpe & Jonas Hallström
Dept. of Behavioural Sciences and Learning,
Linköping University, Sweden

Abstract

There is currently a trend toward introducing computational thinking in schools, and one popular tool to carry this out is through visual programming. A literature review focusing on visual programming and its effects on learning is still lacking, however; especially in STEM (science, technology, engineering, and mathematics) subjects. The aim of this literature review is to investigate and synthesise the findings of research studies on what students potentially learn in STEM subjects from visual programming. Database searches resulted in 25 studies that were analysed qualitatively. Results showed that by engaging in visual programming students do learn to code, but several basic programming concepts are still challenging. Visual programming for learning in STEM subjects is limited concerning science education, whereas it seems as if visual programming could work as a methodical glue for other STEM subjects. By introducing visual programming, our findings indicate that mathematics, technological, and engineering knowledge and skills, as well as basic programming knowledge and skills, are practised and reinforced. However, teachers need to be aware of which competences are in play in the specific teaching and learning modules, in order to support students' learning.

Keywords

Visual programming – STEM subjects – Learning – Literature review

Introduction

Computing has a strong connection to STEM (science, technology, engineering, and mathematics) because it has been conceived of variously as a science, a branch of mathematics, and/or a branch of technology and engineering (e.g. De Mol & Primiero, 2014; Hallström, 2022). The current trend of introducing computational thinking (CT) and programming into school curricula worldwide (Denning & Tedre, 2019a; Wing, 2006) is paralleled by an increasing focus on STEM education; either as individual subjects or as interdisciplinary cooperation between STEM subjects (Hallström & Schönborn, 2019; Tang & Williams, 2019). Visual programming is, for example, used in STEM education to promote subject-specific skills, although little is known about the effects of visual programming on the development of such skills, particularly in technology and engineering.

Visual programming languages are descendants of Seymour Papert's LOGO, which was introduced in the late 1960s with the ambition of promoting CT to children: developing students' cognitive and generic skills such as metacognition, problem solving capability, and creativity (Papert, 1980, 1996). However, research about the effects of learning to program with LOGO on students' cognitive skills or skills in other subjects—the so-called "transfer hypothesis"—was inconclusive (Björklund, 2016; Denning & Tedre, 2019b; Pea, 1987; Scherer et al., 2018; Voogt et al., 2015; Xu et al., 2019). Even though many of today's visual programming languages largely stem from LOGO, there are many differences concerning not only hardware and software but also the technological and educational contexts in which these languages are being implemented.

Aim and Research Question

The aim of this literature review is to investigate and synthesise the findings of research studies on what students potentially learn in STEM subjects from visual programming, in primary and lower secondary education from 2011 throughout 2021. The research question to be answered is: What do students potentially learn in terms of subject specific knowledge and skills in STEM subjects when working with visual programming, according to the reviewed studies?

Background and Earlier Research

The term "computational thinking" (CT) was originally coined by Papert (see e.g. Papert, 1996) to denote the metacognitive, procedural thinking

said to result from programming. Jeannette Wing recently reintroduced the concept and claimed that “[c]omputational thinking is the *thought process* involved in formulating a problem and expressing its solution(s) in such a way that a computer—human or machine—can effectively carry it out” (Wing, 2017, n.p.). Denning and Tedre (2019a) show, however, that computational thinking is a complex set of mental skills and practices that humans have developed for hundreds of years, and that it involves both an engineering design (software and hardware) and a scientific explanation. For the purposes of this chapter, suffice it to say that computational thinking is of empirical interest in the sense that most of the studied works on visual programming also claim to be about CT.

A major tool of education in computational thinking has therefore become visual programming. In schools, programming is often connected to or merged with one or more STEM subjects (e.g. Tucker-Raymond et al., 2019). In some European countries, programming was either seen as part of STEM in the curriculum, or there were plans to integrate it more fully with STEM subjects; in Malta and Belgium, for example (European Schoolnet, 2015). Following a long curriculum debate on computational thinking, Sweden updated the syllabi in technology and mathematics education for compulsory schools to include programming components in 2018 (Digitaliseringskommissionen, 2016; Skolverket, 2017). The new curriculum of 2022 still includes programming—for example, visual programming—in mathematics education and technology education (Skolverket, 2022).

This review will involve an inductive exploration of the character of the findings of research studies on visual programming as a tool for learning in STEM subjects. We define visual programming software as various kinds of graphical or visual environments in which coding is carried out by means of manipulating blocks, symbols, or pictures (e.g., Scratch), but also similar programming software for robotics, such as Lego Mindstorms.

There are some previous reviews focusing on programming and its relation to learning in STEM subjects. Benitti et al. reviewed the educational potential of programming robots in schools. The authors were able to show some gains in student learning in STEM subjects such as mathematics, physics, and programming, although there were also several studies indicating no improvement in learning (Benitti, 2012). Sullivan and Heffernan conducted a similar review focusing on how robotic construction kits function as computational manipulatives in K–12 STEM education. The results, although limited, indicated that robots may help students develop scientific literacy, improve their conceptual

understanding of concepts in physics related to forces and motion, and help in modelling biological systems (Sullivan & Heffernan, 2016).

Furthermore, Scherer et al. (2018) concluded that there was a positive transfer of programming skills to situations that required creative thinking, mathematical skills, and metacognition, as well as those requiring spatial skills and reasoning. Literacy and school achievement in subjects other than mathematics, on the other hand, benefited the least from learning to program.

Although the above results are illuminating, there has not yet been a literature review focusing on *visual* programming and its effects on subject specific knowledge and skills in STEM subjects. Earlier studies have explored older software and/or only partially focused on visual programming (cf. Scherer et al., 2018).

Research Methodology

Data Collection: Literature Searches and Selection

The procedures for performing a literature review employed in this study were inspired by systematic reviews and partly based upon Kitchenham (2004), and Grant and Booth (2009), but some adjustments and refinements have been made to adapt the procedures to the sample

The literature searches were primarily conducted in the electronic database Educational Resources Information Centre (ERIC). ERIC was selected because it covers the most prominent journals in education research. In addition, a complementary search was conducted in Google Scholar, during the initial search stage. Google Scholar was selected because of its very broad scope, in order to find relevant articles that were not, for some reason, indexed in ERIC. Thus, the literature searches in both databases covered the period from 2011 to 2016, but for the 2017–2021 period only ERIC was used because of the abundance of search hits.

The searches in ERIC were carried out in three rounds: a primary and a secondary round, the former being the most extensive and thorough and the latter being complementary, for the years 2011–2016, as well as a third round for the years 2017–2021. In the primary round, the following search terms were used in a full-text search: “Computational thinking” AND programming OR “programming languages” OR “visual programming” AND “elementary education” / “primary education” / “secondary education”. This search resulted in 60 hits, after delimiting it to only peer-reviewed articles in journals with a good reputation; that is, included in Web of Science. The primary search in ERIC was also complemented with

a search in Google Scholar, using the following search terms: “Computational thinking” AND programming OR “programming languages” OR “visual programming” AND “elementary education”. The number of hits was large (141) because it was not possible to automatically apply the criterion of only peer-reviewed journal articles. The sorting of the 141 hits to find only peer-reviewed articles indexed in Web of Science was therefore performed manually, after which 52 articles remained, 27 of which did not appear in the ERIC search. The final outcome of the primary search round was therefore 60 plus 27 articles, a total of 87 peer-reviewed articles.

In the second round, keywords were used as search terms, this time in a full-text search. The terms used in the secondary ERIC search were as follows: “programming” OR “programming languages” AND “elementary education” OR different grades (e.g., DE “Grade 1” OR DE “Grade 2”, etc., up to “Grade 9”). This search resulted in 32 articles, after delimiting the search to peer-reviewed articles in Web of Science journals in the second round. Due to the (so far) open nature of the searches, they identified a total of 119 individual peer-reviewed articles, so an additional search in Google Scholar was deemed unnecessary. For the years 2017–2021 in the third round, we only searched ERIC with the search terms ((DE "Programming" OR DE "Coding" OR DE "Computer Literacy") AND (DE "STEM Education")) AND ((DE "Secondary Education") OR (DE "Primary Education") OR (DE "Elementary Education")) which resulted in 55 articles.

After the three search rounds, further delimitations using several inclusion criteria were applied manually to the 174 articles (cf. Kitchenham, 2004). First, only articles on general education—elementary/primary or secondary education but not areas like special needs education—and only grades 1 through 9, were included. Secondly, only articles that included students’ visual programming were included, so articles only dealing with other programming software or teachers’ development of programming skills were excluded. Third, only articles focusing on STEM subjects (science, technology, engineering, mathematics) or visual programming seen as part of STEM education were included, so studies relating to all other school subjects were excluded. Fourth, the articles included in the final selection were only those that discussed completed empirical studies, so review articles, pilot studies or semi-completed studies, conference papers, and discussion/position papers were excluded.

The inclusion criteria drastically reduced the number of articles. Thus, based on the above inclusion criteria, our review included a total number of 25 unique scientific studies selected for analysis. Studies using qualitative, quantitative, and mixed-methods designs were represented among these 25 articles. For a step-by-step illustration of the whole selection process, see Figure 1.

[Insert Figure 1 about here]

Figure 1. The process of selecting the 25 articles for the literature review

Data Analysis

In this study, we considered it important to synthesise the actual findings of the included studies, to be able to answer our research questions based on data that was as “raw” as possible. Therefore, we were inspired by a method of data analysis called *interpretive synthesis* (Evans, 2002). The first step consists of tabulation; that is, presentation of the main characteristics and major findings of each included study in a table (see Table 1). Based on this table, we also formed a narrative summary of the major findings, as presented in the Findings section (Evans, 2002).

The second step of data analysis occurred in conjunction with writing the aforementioned narrative. For this purpose, we utilised another qualitative and interpretive method in line with Braun and Clarke’s (2006) description of thematic analysis in six phases. According to this method, we treated the findings of the 25 previous studies just as we would any kind of qualitative data or empirical material. The initial two phases were: (1) to familiarise ourselves with the data, which included repeated reading of the tabulation of the entire dataset (based on the original 25 research articles); and (2) to generate initial codes, which meant that all text sections of each article relating to the corresponding research questions were labelled with a descriptive code. This process was inductive. The next phase (3) was to generate themes, which meant sorting the codes into a hierarchical order. The subsequent phase, (4) reviewing themes, involved revising and refining themes to minimise any overlap between them. The next phase (5) consisted of defining and, finally, naming themes, to be presented as headings. The final phase (6) consisted of compiling exemplary data to exemplify the themes/headings, the results of which are presented in the Findings section (Braun & Clarke, 2006). In the Discussion, our thematic analysis and interpretive synthesis are summarised based on the findings, resulting in conclusions in response to the research question.

Findings

Results of the data analysis are presented in this section. This study specifically focused on visual programming software, and the most common software programs used in the included studies are Scratch and ScratchJr (11 out of 25), and different LEGO software (4 out of 25). The remaining studies utilised either lesser-known visual software, or software created specifically for these studies. The findings are presented under three headings, which constitute the main themes. The themes all relate to the research question. These themes are: 1) mathematical knowledge and skills, 2) technological knowledge and skills, and 3) programming knowledge and skills.

[Table 1 near here.]

Mathematical knowledge and skills

Depending on the specific programming tasks, opportunities for learning mathematical calculation and arithmetic may arise. In one study, primary students aged 8 to 9 years used TurtleArt, which is a tool for teaching geometry. Through observations, the researchers identified that students shared mathematical ideas about geometry, such as how to calculate the degrees of an angle (Boyd Harlow & Emerson Leak, 2014).

In a study that involved sixth graders working with programming robots, students worked in groups and discussed the objectives of the robot's design, construction, and programming. Students were also able to articulate their mathematical strategies (Castledine & Chalmers, 2011). Furthermore, these students used a variety of calculation methods, including estimation, addition, subtraction, and division.

Even though neither of these studies presented evidence of students' development of mathematical skills, they both showed that students discussed and shared mathematical ideas that may lead to the development of their mathematical content knowledge (Boyd Harlow & Emerson Leak, 2014; Castledine & Chalmers, 2011). This seems to be evident even if the task is not explicitly based in mathematics.

Similarly, Kalelioğlu (2015) did not test skills development in mathematics *per se*, but students were asked what skills they thought they had improved through the programming intervention. Several stated that they felt they had improved their mathematical and geometrical knowledge, although the study did not show evidence of the nature of such knowledge, and this was only what the students themselves expressed.

However, the students had worked on Code.org, which contains, for example, tasks for calculating angles, so calculation was indeed part of the programming curriculum.

In a study of 84 students aged 12–14 years, Gülmez and Özdener (2015) identified a correlation between students' overall academic success and their success in algorithm development. Their results also showed that there was a significant positive correlation between average programming post-test results and achievement in mathematics.

Dickes et al. (2020) claim that curricular integration of computing in K–12 STEM contexts is a challenging task for both teachers and students because it involves the introduction and adoption of new literacies (e.g., programming) alongside disciplinary ideas that students already find challenging to understand, such as mathematics. The classroom teacher's emphasis on mathematising and measurement as key learning activities helped to meaningfully integrate programming as an integral component of STEM work in the classroom in a manner that also helped them connect computational thinking and modelling with regular mathematics curricular needs (Dickes et al., 2020).

Kim et al. (2021) wrote that students in their study discovered the concepts of special angle pairs in geometry—namely, complementary and supplementary angles—as they learned to navigate the immediate feedback from the robot Sphero SPRK+ into a trial-and-error mathematics problem-solving process. Students' experiences in the coding activities revealed that engaging in reflective play could be shaped into meaningful teachable moments where students could participate in learning through pedagogical methods using educational robotics. These activities had transferability implications that might afford STEM learning access and opportunities for students to develop not only mathematical reasoning skills, but also problem solving and critical thinking skills operable in a coding environment.

However, Miller (2019) discusses that the teachers are not always aware of the mathematics in coding lessons. He concludes that there is a need to acknowledge mathematics, as it is not always apparent for the teachers. Miller (2019) shows that year 2 primary students improved their skills in recognising and deducing patterns when they had participated in a coding intervention. Each of the six lessons in the intervention focused on teaching mathematical concepts (e.g., drawing a square or moving a robot through a particular path) using Scratch and three coding robots. The in-depth analysis of the classroom activities shows that the students demonstrated higher levels of mathematical thinking in three different areas: "(i) converting their knowledge of quarter turns to work with metric measures

of 90° turns, (ii) orientation and perspective taking, and, (iii) deducing a repeating pattern to provide a generalised code for making a square” (Miller, 2019, p. 923).

In summary, the studies identified that students shared mathematical ideas regarding geometry (Boyd Harlow & Emerson Leak, 2014) and articulated their mathematical strategies (Castledine & Chalmers, 2011). A correlation between students’ overall academic success and their success in algorithm development was identified by Gülmez and Özdener (2015), as was a significant positive correlation between average programming post-test results and mathematics achievement. There is thus some evidence that programming activities could inform the development of mathematical skills, especially if teachers are made aware of the mathematics content in coding lessons (Miller, 2019).

Technological knowledge and skills

Some studies identified that when students programmed robots, they were also able to understand the software program, the different parts of the robot, and the sensors as a technical system. For example, at the beginning of a programming course, a systems approach occurred only occasionally among 11- to 12-year-old students when working on designing and programming robots (Slangen et al., 2011). The students were required to compare a well-functioning robot with one that had three different kinds of bugs. One bug was a gear that was not attached properly, another was a incorrectly-plugged cable, and the third was a software bug. The researchers concluded:

This analytic approach seemed to work well, since all pairs detected and repaired the bugs. In this problem solving context, pupils successfully identified subsystems or parts and their proper function. (Slangen et al., 2011, p. 459)

Hence, this kind of debugging task fostered an approach in which the students practiced separating different sub-systems. In the same study, students were asked to program a robot from scratch to perform a pre-defined task. They had to analyse the task, make a plan for the work, identify which functions the robot needed, and then construct, program and test the robot. Going through all these steps required students to identify the whole system as well as its subsystems and how different parts were adapted to each other. Even though Slangen et al. (2011) stated that the systems thinking the students developed was largely implicit, they also

concluded that there was a development in how the students talked about construction as a whole:

The ability to 'see' properly is no longer attributed to the robot as such, but is related to the position of the sensor in the whole construction. (Slangen et al., 2011, p. 460)

Slangen et al. (2011) claimed that working with light sensors and how they function helped students to reason in terms of input/process/output. From there, the students were better prepared to reason about how the program should be written.

A study in which sixth-grade students worked with LEGO Robotics revealed similar results. The students were required to both construct their own robots and program the robots to solve a task (Castledine & Chalmers, 2011). The students were active in their learning, as they discussed the robot design and construction as well as the software programming, and progressively, as their self-efficacy increased, they began to link the programming to the various sensors and to the robot's performance.

Students working with robotics use both domain specific problem-solving strategies and domain general strategies (Sullivan & Lin, 2012). The students in this example (11–12 years old) were working on a problem where they were instructed to construct robots by adding, for example, sensors and wheels, so that they could program the robots to follow an angled line on a paper.

Tan et al. (2020) writes that their findings proved that a STEAM (science, technology, engineering, arts, mathematics) integrated approach achieved via Scratch could narrow the gap between male and female technology learning. The findings indicated that the intervention had positive effects on male and female students' achievement in learning concepts of electricity. This research also provided a new method and an alternate connective framework for learning concepts of electricity via art, and showed that both males and females were able to understand the topic of electricity, which reduced gender biases.

The findings of Hébert and Jenson (2020) demonstrated that students in two distinct learning environments showed measurable increases in their knowledge of coding and circuitry after making an e-textile wearable. Students' self-reported areas of challenge and where they required the most assistance in making wearables also calls attention to gaps in knowledge around sewing and coding, while student responses that they were not likely to take up making outside of the classroom speak to the need to more

explicitly incorporate making into the curriculum as a means of supporting twenty-first century competency development and learning in STEM.

Similarly, Nugent et al. (2019) used a quasi-experimental, pre–post design with two groups (treatment and control) to measure the impact of a wearable technology intervention on students’ knowledge of circuitry, programming, and engineering design in making a wearable e-textile product. Results indicate that wearable technology integration of engineering, computing, and aesthetics promises to be a good interdisciplinary context for supporting students’ STEM learning and attitudes at the upper elementary level. The wearable technology instruction was found to promote learning of both males and females and under-represented and majority students when compared with a control group. But while self-efficacy of both male and female students was significantly higher than that of the control groups, results showed that males’ confidence exceeded that of females in the areas of programming and circuitry.

In summary, depending on the assignment, it seems as though working with visual programming in the context of robotics can enhance students’ technological systems skills. Programming robots may foster a systems approach, in which the software, the sensors, and the robot have to be seen as components of a larger technological system. Furthermore, through coding in Scratch and wearable e-textiles, understanding of electricity and circuitry could be enhanced both for boys and girls; although it seems that male students gain the most in confidence.

Programming knowledge and skills

Most of the studies included in this literature review showed that students developed their skills in programming through various activities. One of the studies reported inconclusive results, where differences between the participating schools varied largely (Moreno-León et al., 2016). Researchers found that some teachers found it harder to implement the programming activities than others. A few studies reported on differences between different teaching and learning approaches, and the following section will communicate these findings. This sub-theme focuses on how students learn to program and how they learn programming concepts. The results also report on which programming blocks students use in their programming activities, and how well they perform in the implementation.

ScratchJr is a programming environment inspired by Scratch and adapted for younger students, aged 5–7 years. In a study, Portelance et al. (2016) investigated which blocks children in kindergarten, first grade, and

second grade used in their projects after being taught programming for twelve lessons. The blocks in ScratchJr are organised into different categories, depending on their function. The results show that the motion block, with “move right”, was the most frequently used block across all ages. The trigger block, “start on green flag”, was the second most commonly used block and the least commonly chosen block was the control flow block “stop”. Some differences between the age groups were reported. However, the main conclusion was that children were able to use the blocks to create their own projects.

However, Sung et al. (2017) showed that the learning environment is crucial for young learners to develop their understanding of programming concepts. The researchers introduced ScratchJr to kindergarten and first-grade students. However, they concluded that it is most important to expose the students to appropriate learning activities that support computational perspectives and practices before exposing them to digital tools. Moreover, studies indicate that problem-solving-based learning; both individual (Su et al., 2014), and collaborative (Emara et al., 2021), improves students’ programming skills.

Older students were able to make more complex programs when they used graphical programming tools (Louca et al., 2011). However, students who initially learned programming in a text-based programming environment seemed to be able to transfer their knowledge to visual and text-based programming problems (Okita, 2014). Students who initially learned programming in a graphic environment had to work harder to apply their skills in text-based environments, even when they had been trained in text-based programming.

Mouza et al. (2016) investigated the effects of a nine-week after-school program in Scratch for middle-school students. Data was collected using a pre- and post-test design, in which students were asked to match a Scratch script to a computing concept such as loop, conditional, variable, or parallelism. They were also asked to determine the result of executing a script. The findings indicated that there was, overall, significant improvement in students’ knowledge of computing concepts after the course. However, no significant improvement was found in two out of the three questions dealing with the concept of loop. This was explained by the fact that students were not familiar with the language used in one of the questions. When analysing students’ Scratch projects, Mouza et al. (2016) found that the most common block associated with a computing concept was the loop; loops were found in almost 75% of projects. Conditionals were the second most common concept (58%). In more than 85% of the cases where loops and conditionals were used, the code also worked as

intended. Moreover, most of the students showed evidence of appropriate code organisation and documentation. The results indicate that the students both understood and were able to implement computational concepts such as loops, conditionals, data, and parallelism.

Kalelioğlu and Gülbahar (2014) showed that students did not improve their problem-solving skills after a five-week program using Scratch. The researchers suggest that the students added new blocks randomly using a trial-and-error approach, or because the teacher told them to do so.

Moreover, the results of Kwon et al. (2012) showed that the students found it easier to both understand and use programming activities that involve an actual robot. The control group used Scratch and solved problems by moving a virtual robot on the screen.

In their study, Sullivan and Bers (2016b) investigated children's programming knowledge after attending an eight-week robotics course. The children were of different ages, ranging from pre-kindergarten to second grade. The children in kindergarten, and the first and second graders all performed equally well on all the tasks. However, the children demonstrated a variety of mistakes as they were solving the tasks. Some mistakes were syntactical in character; which is to say that the children had made logical programming errors that rendered the programs dysfunctional for a real robot. Other mistakes were story-related, which means that the sequence did not match the intended actions of the robot. Overall, however, the children performed better on sequencing tasks than repeat loop or conditional statement tasks.

In another study conducted by the same researchers, similar results were found for the same programming curriculum (Sullivan & Bers, 2016a). In this study, the authors also reported on the absence of any statistical gender differences when it came to the simplest tasks. However, in the more advanced programming tasks, such as using repeated loops with numbers or sensors, boys performed significantly better than girls.

In a quasi-experimental study, Sáez-López et al. (2016) investigated the effect of an intervention using design-based research. The study was conducted with sixth graders in five different schools and the sample consisted of 107 students. The results showed a significant improvement from pre- to post-test in students' ability to understand programming concepts, after an active pedagogical intervention using a visual programming language. The test assessed students' understanding of different computational concepts, and the use of different commands and visual blocks. Important concepts were sequences, loops, conditional

statements, parallel execution, coordination, event handling, and keyboard input (Sáez-López et al., 2016).

In summary, the reviewed studies show that students learn to operationalise computing concepts with visual software through different interventions (e.g. Sáez-López et al., 2016). However, students, at least when it comes to younger students, found it easier to build sequences than to use more complex functions such as loops and conditionals, although the latter were also commonly employed (Sullivan & Bers, 2016b). Sung et al. (2017) also pointed out the importance of the learning environment as a precondition for students in learning programming. Moreover, several studies showed improvements from interventions, but programming concepts such as sequences, loops, conditional statements, parallel execution, etc. were still challenging for students to learn, especially conditionals (Mouza et al., 2016; Sáez-López et al., 2016; Sullivan & Bers, 2016b). Boys performed better at the more advanced programming tasks, although this was only reported by one study (Sullivan & Bers, 2016a).

Discussion

The research question in this study is: What do students potentially learn in terms of subject-specific knowledge and skills in STEM subjects when working with visual programming, according to the reviewed studies? Three main themes emerged from the review of the included studies: mathematical knowledge and skills, technological knowledge and skills, and programming knowledge and skills.

In line with Scherer et al. (2018), we could also see improvement on STEM-related mathematical skills. When it comes to the more subject-specific competencies, this review references several studies that suggest various ways in which mathematical competencies and self-efficacy might be enhanced by visual programming. However, in regard to chemistry, physics, and biology education, none of the included studies point to any benefits of visual programming on learning; even though STEM was one of our inclusion criteria. This contrasts with an earlier review on the effects of robotics on STEM learning. Sullivan and Heffernan's review focused on how robotic construction kits function as computational manipulatives. Their findings, although limited, indicated that robots may help students develop their scientific literacy, improve conceptual understanding of physics concepts related to forces and motion, and help in modelling biological systems (Sullivan & Heffernan, 2016).

There were a couple of studies (Castledine & Chalmers, 2011; Slangen et al., 2011) performed in a technology education context. One of these indicated a development in students' systems thinking in terms of seeing how a system is made up of components and subsystems that relate to each other; the students could also analyse the system in terms of model input–process–output (Slangen, et al., 2011). Visual programming thus helped these students acquire advanced systems thinking skills for their age (cf. Hallström & Klasander, 2017).

Earlier reviews in STEM learning and programming have focused on specific skills, rather than on subject specific improvement. By breaking down computational thinking into several narrower and more well-defined skillsets, it may be possible to determine the effects of programming on CT. Our results show some similarities with Scherer et al. (2018), in that we saw an effect of learning to program on programming skills, although the far-transfer effects we saw were less clear, especially for low achievers. We could see some positive effects on problem-solving strategies and capabilities, although overall the results were inconclusive, and in terms of abstract and logical thinking there was no observable effect. The latter could be compared to what Scherer et al. (2018) labelled “reasoning”, which had a less than moderate effect.

The findings of this review regarding students' learning of programming skills are similar to a British study by Straw et al. (2017). They evaluated so-called “Code Clubs” in the UK, and more precisely what impact attending such a club might have on children's computational thinking, programming skills, and attitudes toward computers and coding more generally. They concluded that attending Code Club for a year did not impact upon children's computational thinking—determined by having the children take a test about simple programming (e.g., giving proper instructions), problem solving and logical thinking—beyond changes that would have happened anyway; that is, compared to a control group. Mixed results like these were also seen in studies during the 1980s in the wake of LOGO. In fact, it seems that inconclusive results persist, regardless of the definition of CT, as long as CT is defined in rather general terms (e.g. Voogt et al., 2015).

However, if we focus on more narrowly-defined programming knowledge and skills, there were some reported effects in the reviewed studies. Students can learn to operationalise computing concepts with visual software through different interventions; especially simpler sequential operations as opposed to conditionals which were more difficult to learn, especially for girls (Sullivan & Bers, 2016a).

Conclusion

The introduction of computational thinking through visual programming in schools all over the world is a very significant development that will affect school systems for years to come. Because of this, it is imperative that this introduction is based on solid research, but so far there have been few studies and few clear-cut results. One reason for the inconclusiveness of prior research may be the fuzziness and multiple definitions of the concept of computational thinking, but the inconclusiveness seems to remain, regardless of definition. However, in this literature review, we focus on skills and knowledge in STEM subjects. We conclude this chapter by summarising our main findings:

Visual programming for learning in STEM subjects is limited to mathematical, technological, and programming knowledge and skills. This means that studies focusing on the scientific subjects such as biology, physics, and chemistry have not been found in this literature review.

Engaging in visual programming can nonetheless likely help develop skills in all the aforementioned subjects. For mathematics, it seems important that teachers are aware that programming activities include mathematic components (Dickes et al., 2020; Miller, 2019). Even if it could be challenging for students and teachers to introduce new programming literacies into a mathematical context (Dickes et al., 2020), several studies indicate that it does lead to more conversations about mathematics among students (Boyd Harlow & Emerson Leak, 2014).

Moreover, working with robotics and to programming robots seems to reinforce important technological knowledge and skills; such as, for example, technological systems thinking (Slangen et al., 2011), and circuitry, programming, and engineering design, especially in the context of the relatively new area of e-textiles (Hébert & Jenson, 2020; Nugent et al., 2019).

To conclude, it seems as if visual programming could work as a methodical glue for STEM subjects. By introducing visual programming in elementary school, our results indicate that mathematics, technological and engineering knowledge and skills, and basic programming knowledge and skills, are practised and reinforced. However, teachers need to be aware of which competences are at play in specific teaching and learning modules, in order to best support students' learning.

Limitations

There are some potential limitations to our selection of research articles in our review. First, the search algorithms of ERIC and Google Scholar are not entirely transparent, so the search results would probably have been slightly different had we made small changes in the Boolean search strings, or in the keywords. As a matter of fact, we once received slightly different results with the same search string on a certain day, compared to the day before. Because of this slight discrepancy, we also validated the selected studies at a research seminar with experts from STEM, ICT, and programming education. The final 25 studies were deemed a reasonable outcome of our searches and the application of the inclusion criteria. Secondly, there are other selection methods that we did not utilise; for example, snowball selection from reading lists of references in previous studies. The selected previous studies for this review might have been different had we used one or other of these methods, but we believe that the difference would have been small, given the previously-mentioned external validation (e.g. Robson, 2002). Thirdly, there is also a limitation related to the results of the included studies; namely, that not all of them are entirely transparent in regard to research design and methodology (see Table 1). It is thus sometimes difficult to compare them due, for example, to low transparency regarding what kinds of questions were asked, or how certain interventions were carried out, in more detail.

References

- Benitti, F. B. V. (2012). Exploring the educational potential of robotics in schools: A systematic review. *Computers & Education*, 58(3), 978-988.
<https://doi.org/https://doi.org/10.1016/j.compedu.2011.10.006>
- Björklund, L. (2016, June 15, 2016). *Datalogiskt tänkande, ett förhoppningsbaserat, generellt sätt att lösa problem?* CETIS' Summer Seminar, Rockelstad Castle, Sweden.
- Boyd Harlow, D., & Emerson Leak, A. (2014). Mapping students' ideas to understand learning in a collaborative programming environment. *Computer Science Education*, 24(2-3), 229-247.
- Braun, V., & Clarke, V. (2006). Using thematic analysis in psychology. *Qualitative Research in Psychology*, 3(2), 77-101.
- Castledine, A.-R., & Chalmers, C. (2011). LEGO Robotics: An authentic problem solving tool? *Design and Technology Education: An International Journal*, 16(3), 19-27.

- De Mol, L., & Primiero, G. (2014). Facing computing as technique: towards a history and philosophy of computing. *Philosophy & Technology*, 27(3), 321-326.
- Denning, P. J., & Tedre, M. (2019a). *Computational thinking*. MIT Press.
- Denning, P. J., & Tedre, M. (2019b). Thinking Like a Computer. *American Scientist*, 107(3), 187-189.
- Dickes, A. C., Farris, A. V., & Sengupta, P. (2020). Sociomathematical Norms for Integrating Coding and Modeling with Elementary Science: A Dialogical Approach. *Journal of Science Education and Technology*, 29(1), 35-52. <https://doi.org/10.1007/s10956-019-09795-7>
- Digitaliseringskommissionen. (2016). *Digitaliseringens effekter på individ och samhälle – fyra temarapporter. Delbetänkande av Digitaliseringskommissionen*.
- Emara, M., Hutchins, N. M., Grover, S., Snyder, C., & Biswas, G. (2021). Examining Student Regulation of Collaborative, Computational, Problem-Solving Processes in Open-Ended Learning Environments. *Journal of Learning Analytics*, 8(1), 49-74.
- European Schoolnet. (2015). *Computing our future, Computer programming and coding: Priorities, school curricula and initiatives across Europe*. E. Schoolnet.
- Evans, D. (2002). Systematic reviews of interpretive research: Interpretive data synthesis of processed data. *Australian Journal of Advanced Nursing*, 20(2), 22-26.
- Grant, M. J., & Booth, A. (2009). A typology of reviews: an analysis of 14 review types and associated methodologies. *Health Information & Libraries Journal*, 26(2), 91-108.
- Gülmez, I., & Özdener, N. (2015). Academic achievement in computer programming instruction and effects of the use of visualization tools; at the elementary school level. *British Journal of Education, Society & Behavioural Science*, 11(1), 1-18.
- Hallström, J. (2022). The Dual Nature(s) of Technology: Towards a Unified Conception of Digital and Analogue Technology. PATT 39: PATT on the Edge of Technology, Innovation and Education, St. John's, Newfoundland and Labrador, Canada.
- Hallström, J., & Klasander, C. (2017). Visible parts, invisible whole: Swedish technology student teachers' conceptions about technological systems. *International Journal of Technology and Design Education*, 27(3), 387-405. <https://doi.org/10.1007/s10798-016-9356-1>
- Hallström, J., & Schönborn, K. J. (2019). Models and modelling for authentic STEM education: reinforcing the argument. *International Journal of STEM Education*, 6(1), 1-10.

- Hébert, C., & Jenson, J. (2020). Making in schools: Student learning through an e-textiles curriculum. *Discourse: Studies in the Cultural Politics of Education*, 41(5), 740-761.
- Kalelioğlu, F. (2015). A new way of teaching programming skills to K-12 students: Code.org. *Computers in Human Behavior*, 52, 200-210.
- Kalelioğlu, F., & Gülbahar, Y. (2014). The effects of teaching programming via Scratch on problem solving skills: A discussion from learners' perspective. *Informatics in Education*, 13(1), 33-50.
- Kim, Y. R., Park, M. S., & Tjoe, H. (2021). Discovering concepts of geometry through robotics coding activities. *International Journal of Education in Mathematics, Science, and Technology (IJEMST)*, 9(3), 406-425.
- Kitchenham, B. (2004). *Procedures for Performing Systematic Reviews* (Keele University Technical Report TR/SE-0401).
- Kwon, D.-Y., Kim, H.-S., Shim, J.-K., & Lee, W.-G. (2012). Algorithmic Brics: A tangible robot programming tool for elementary school students. *IEEE Transaction on Education*, 55(4), 474-479.
- Louca, L. T., Zacharia, Z. C., Michael, M., & Constantinou, C. P. (2011). Objects, entities, behaviors, and interactions: a typology of student-constructed computer-based models of physical phenomena. *Journal of Educational Computing Research*, 44(2), 173-201.
- Miller, J. (2019). STEM education in the primary years to support mathematical thinking: using coding to identify mathematical structures and patterns. *ZDM*, 51(6), 915-927. <https://doi.org/10.1007/s11858-019-01096-y>
- Moreno-León, J., Robles, G., & Román-González, M. (2016). Code to learn: Where does it belong in the K-12 curriculum? *Journal of Information Technology Education: Research*, 15, 283-303.
- Mouza, C., Marzocchi, A., Pan, Y.-C., & Pollock, L. (2016). Development, Implementation, and Outcomes of an Equitable Computer Science After-School Program: Findings From Middle-School Students. *Journal of Research on Technology in Education*, 48(2), 84-104.
- Nugent, G., Barker, B., Lester, H., Grandgenett, N., & Valentine, D. (2019). Wearable Textiles to Support Student STEM Learning and Attitudes. *Journal of Science Education and Technology*, 28(5), 470-479. <https://doi.org/10.1007/s10956-019-09779-7>
- Okita, S. Y. (2014). The relative merits of transparency: Investigating situations that support the use of robotics in developing student learning adaptability across virtual and physical computing platforms. *British Journal of Educational Technology*, 45(5), 844-862.

- Papert, S. (1980). *Mindstorms: Children, computers, and powerful ideas*. Basic Books, Inc..
- Papert, S. (1996). An Exploration in the Space of Mathematics Educations. *International Journal of Computers for Mathematical Learning*, 1(1), 95-123.
- Pea, R. (1987). The aims of software criticism: Reply to Professor Papert. *Educational Researcher*, 16(5), 4-8.
- Portelance, D. J., Strawhacker, A. L., & Bers, M. U. (2016). Constructing the ScratchJr programming language in the early childhood classroom. *International Journal of Technology and Design Education*, 26(4), 489-504. <https://doi.org/10.1007/s10798-015-9325-0>
- Robson, C. (2002). *Real World Research: A Resource for Social Scientists and Practitioner-Researchers* (2nd ed.). Blackwell.
- 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.
- Scherer, R., Siddiq, F., & Sánchez Viveros, B. (2018). The Cognitive Benefits of Learning Computer Programming: A Meta-Analysis of Transfer Effects. *Journal of Educational Psychology*, 1-29. <https://doi.org/http://dx.doi.org/10.1037/edu0000314>
- Skolverket. (2017). *Förordning om ändring i förordningen (SKOLFS 2010:37) om läroplan för grundskolan, förskoleklassen och fritidshemmet*. Stockholm: Skolverket
- Skolverket. (2022). *Läroplan för grundskolan, förskoleklassen och fritidshemmet. Reviderad 2022*.
- Slangen, L., van Keulen, H., & Gravemeijer, K. (2011). What pupils can learn from working with robotic direct manipulation environments. *International Journal of Technology and Design Education*, 21(4), 449-469.
- Straw, S., Bamford, S., & Styles, B. (2017). *Randomised Controlled Trial and Process Evaluation of Code Clubs*. National Foundation for Educational Research and the Raspberry Pi Foundation.
- Su, A. Y. S., Yang, S. J. H., Hwang, W.-Y., Huang, C. S. J., & Tern, M.-Y. (2014). Investigating the role of computer-supported annotation in problem-solving-based teaching: An empirical study of a Scratch programming pedagogy. *British Journal of Educational Technology*, 45(4), 647-665.
- Sullivan, A., & Bers, M. U. (2016a). Girls, Boys, and Bots: Gender Differences in Young Children's Performance on Robotics and Programming Tasks. *Journal of Information Technology Education: Innovations in Practice*, 15, 145-165.

- Sullivan, A., & Bers, M. U. (2016b). Robotics in the early childhood classroom: learning outcomes from an 8-week robotics curriculum in pre-kindergarten through second grade. *International Journal of Technology and Design Education*, 26(1), 3-20. <https://doi.org/10.1007/s10798-015-9304-5>
- Sullivan, F. R., & Heffernan, J. (2016). Robotic Construction Kits as Computational Manipulatives for Learning in the STEM Disciplines. *Journal of Research on Technology in Education*, 48(2), 105-128.
- Sullivan, F. R., & Lin, X. (2012). The ideal science student: Exploring the relationship of students' perceptions to their problem solving activity in a robotics context. *Journal of Interactive Learning Research*, 23(3), 273-308.
- Sung, W., Ahn, J., & Black, J. B. (2017). Introducing Computational Thinking to Young Learners: Practicing Computational Perspectives Through Embodiment in Mathematics Education. *Technology, Knowledge and Learning*, 22(3), 443-463. <https://doi.org/10.1007/s10758-017-9328-x>
- Tan, W.-L., Samsudin, M. A., Ismail, M. E., & Ahmad, N. J. (2020). Gender differences in students' achievements in learning concepts of electricity via STEAM integrated approach utilizing scratch. *Problems of Education in the 21st Century*, 78(3), 423.
- Tang, K. S., & Williams, P. J. (2019). STEM literacy or literacies? Examining the empirical basis of these constructs. *Review of Education*, 7(3), 675-697.
- Tucker-Raymond, E., Puttick, G., Cassidy, M., Hartevelde, C., & Troiano, G. M. (2019). "I Broke Your Game!": critique among middle schoolers designing computer games about climate change. *International Journal of STEM Education*, 6(1), 1-16.
- Voogt, J., Fisser, P., Good, J., Mishra, P., & Yadav, A. (2015). Computational thinking in compulsory education: Towards an agenda for research and practice. *Education and Information Technologies*, 20, 715-728. <https://doi.org/10.1007/s10639-015-9412-6>
- Wing, J. M. (2006). Computational Thinking. *Communications of the ACM*, 49(3), 33-35.
- Wing, J. M. (2017, 8 September 2017). *Computational Thinking* Trippel Helix Conference on Computational Thinking and Digital Competences in Primary and Secondary Education, Stockholm.
- Xu, Z., Ritzhaupt, A. D., Tian, F., & Umapathy, K. (2019). Block-based versus text-based programming environments on novice student learning outcomes: a meta-analysis study. *Computer Science*

Education, 29(2-3), 177-204.
<https://doi.org/10.1080/08993408.2019.1565233>

Table 1. Summary of the twenty-five included articles in terms of aim/topic, sample, programming language/software, study design, data collection, and major findings

Article	Aim/topic	Sample	Programming language/software	Study design and data collection	Major findings
Boyd Harlow and Emerson Leak (2014)	How can collective idea and skill development be mapped and what classroom interactions lead to skill development and innovation?	20 third-grade students, 8–9 years old, and their teacher	TurtleArt—tool for teaching geometry	Qualitative. Data was collected over one school year. Participant observers, video recordings, two video cameras. Observable ideas (memes) were used as the tracer unit in the analysis.	Programming activities made students share ideas with each other concerning programming and mathematics. Ideas on how to program something more effectively were most often prompted by questions from the teacher. Ideas that led to using commands to obtain new outputs were prompted by questions or comments from students.
Castledine and Chalmers (2011)	Are LEGO robotics an effective problem-solving tool? Investigate problem-solving strategies students are engaged in.	23 grade 6 students	LEGO Mindstorms software program	Qualitative. Two weeks of daily one-hour lessons. Two problems for the students to solve—The Race and The Maze. Observation of student problem-solving discussions, and collection of student questionnaire and software programs.	Results indicate that student efficacy seems to play an important role when it comes to their willingness to seek their own solutions in programming issues. As efficacy increased, the students searched more actively for solutions to their problems. However, students were not able to discuss the actual problem-solving skills they used during the activities.
Dickes, Voss Farris, and Sengupta (2020)	How classroom teachers' actions and interactions with an agent-based programming tool impacted how computing was adopted and integrated into existing classroom practice. RQs: 1. How did the teacher's emphasis on mathematizing and measurement support the development of sociomathematical norms around model 'goodness', and how were those norms	15 third grade students, of which eight were female and seven male. The classroom teacher, Emma, was a first-year teacher in the school district.	ViMAP; an agent-based, visual programming language that uses the NetLogo modelling platform as its simulation engine	A design-based, microgenetic study in which an agent-based programming and computational modelling platform—ViMAP—was integrated with existing elementary science and math curricula through lessons co-designed and taught by the classroom teacher across a period of seven months. A dialogical re-positioning of coding is utilised, where disciplinarily grounded meanings of code emerge through the construction of computational utterances—i.e., computer models and complementary conversations and physical models that serve as mathematical and scientific explanations—through the use of socio-mathematical norms.	One finding that emerged is that curricular integration of computing in K–12 STEM contexts is a complex and challenging endeavour for both teachers and students that involves the introduction and adoption of new literacies (e.g., programming) alongside disciplinary ideas that students already find challenging to understand. The classroom teacher's emphasis on mathematizing and measurement is suggested to be key in learning activities that help to meaningfully integrate programming as an integral component of STEM work in the classroom in a manner that also helped her connect two new literacies—computational thinking and modelling—with her regular mathematics and science curricular needs. The

	<p>adopted by the students? 2. Did these norms shape in any way the development of students' computational models and computational thinking?</p>				<p>classroom teacher, in partnership with researchers, found and created opportunities to integrate agent-based programming with her regular science and mathematics curricula by iteratively developing sociomathematical norms.</p>
<p>Emara, Hutchins, Grover, Snyder, and Biswas (2021)</p>	<p>RQ1: what is the nature of collaborative problem solving (CPS) regulation activated by students when they work in groups on three different types of physics modelling tasks? RQ2: How do students' self- and shared regulatory activities correlate with the performance? RQ3: How do action patterns and problem-solving strategies derived using natural language processing (NLP) emerge across collaborative computational modelling tasks?</p>	<p>14 9th grade students, mean age 15 years, attending a selective program designed to immerse high school students in advanced academic experiences in a university setting.</p>	<p>NetsBlox which is an extension of the Snap! With custom domain-specific blocks that help learners to focus on physics concepts.</p>	<p>Students' visual and verbal behaviour were recorded using a screen-capture software that recorded activities on the screen, video, and audio. Qualitative data analysis—multimodal analytics that combined student discourse and activity data from the computer-based model building environment.</p>	<p>Student CPS strategies evolved over time; e.g., use of more systematic model-building action sequences. Students use more persuasive and explanatory words such as "because" when they engage in more open-ended challenge tasks. However, the study does not contribute on students' learning of CT or STEM in detail.</p>

Gülmez and Özdener (2015)	The purpose was to compare narrative tools with flowchart model tools, in terms of student success and motivation. Determine the effect of academic success on success in developing algorithms.	84 students who had taken programming education in elementary school. Participants ranging in age from 12 to 14. The students were novice programmers.	Scratch (Exp group 2) and Fcpro tool (Exp group 1). Fcpro is a flow chart model tool ensuring the creation of an algorithm by bringing the codes through the drag-and-drop method.	Quantitative. One of the groups (Exp group 1) used the flowchart model tool and the other (Exp group 2) used the narrative tool. The student groups were exposed to the tools for eight weeks. A pre-test was conducted to determine whether the groups were equal. At the end of applications, two post tests were used to compare the effects of the two visualisation tools on algorithm development success. A motivation test was conducted before and after the intervention.	There was no significant difference in students using flowchart model tools compared with students using narrative tools (Scratch) in terms of their success on the use of variables. There was a significant difference in favour of students using narrative tools on the use of both conditions and loop expressions. There was no difference in terms of the two groups' motivation. There was a positive significant correlation between the post-test averages and academic achievements in English course, and there was a positive significant correlation at a medium level in Turkish, maths and information technologies courses. Achievements in information technologies, maths, Turkish, and English, have a positive significant correlation at a medium level with the algorithm development achievement.
Hébert and Jenson (2020)	Examining student learning through a making project, focused on the construction of e-textiles—in this case, wearable hats—run as part of a series of after school workshops and as an in-class unit in a school in Ontario, Canada. We asked: through participation in this making unit, what can we report that students learned about coding and circuitry? What elements of making did students struggle with and	18 students from grades 6, 7, and 8 took part in the after-school workshops, 10 who identified as male and 8 as female. For the in-class portion of the project, participants consisted of 18 grade 6 students, 10 who identified as male, 8 as female.	LilyPad Arduino	This study examined student learning through a making project and the construction of e-textiles—wearable hats—in a unit delivered in both a series of after-school workshops and as in-class lessons in a school in Ontario, Canada. The workshops took place after school, over 12 sessions, each two hours in length. Groups were divided by gender. There were 10 in-school sessions. A pre-test on programming and electronics was taken by students prior to the making intervention, and a post-test was conducted afterwards. Students were also interviewed at the end of the project about their past experience making wearables, coding, and sewing, and what they learned through the process of participating in the unit. Altogether, 15 students were interviewed.	Findings demonstrated that students in two distinct learning environments showed measurable increases in their knowledge of coding and circuitry after making an e-textile wearable. Students' self-reported areas of challenge and where they required the most assistance in making wearables also calls attention to gaps in knowledge around sewing and coding, while student responses that they were not likely to take up making outside of the classroom speak to the need to more explicitly incorporate making into the curriculum as a means of supporting twenty-first century competency development and learning in STEM.

	find the most accessible, and what were their key take-aways from participating in the project?				
Kalelioğlu (2015)	Investigates, at a macro level, the effect of teaching code.org site on reflective thinking skills towards problem solving, and whether there is a gender difference.	32 primary students in grade 4 (10 years old) attending a computer course at a private school in Turkey. 17 females and 15 males.	code.org	Qualitative/ quantitative. The course was taught for one hour per week over a period of five weeks. Reflective Thinking Skill Scale Towards Problem Solving: questioning, reasoning, and evaluation. Focus group structured interviews were performed after completion of the five weeks.	Generally, there were no differences between pre- and post-test results concerning students' reflective thinking skills towards problem solving. However, there was a slight increase in female students' such skills. Students performed higher (tried to complete more levels) than in their normal classroom activities, and females performed slightly better than males. The students enjoyed programming with code.org and found it to be easy, and many said that they wanted to learn more about programming.
Kalelioğlu and Gülbahar (2014)	This study tried to explore the effectiveness of using Scratch in the process of teaching children programming in terms of contribution to problem solving skills.	49 primary school students (5th grade) who were attending a computer course at a private school.	Scratch	Qualitative/ quantitative. One hour per week over a period of five weeks, to write the programs at school (Hello World, Parrot, Aquarium programs and Maze project). Problem Solving Inventory—self-perception levels about their problem-solving skills. 24 items with three factors: 12 items for self-confidence in their problem-solving ability, 7 items for self-control, and 5 items for avoidance.	There was no difference between the pre- and post-test results regarding students' problem-solving skills. To solve their problems, the students asked the teacher or a peer. Students solved problems e.g., by adding new items to the programs or by doing what the teacher told them to do. The complexity of long blocks and the absence of special effects were mentioned as non-favourite aspects of the applications. All students stated that they liked programming, that they wanted to improve themselves in programming and that they wanted to create larger, more difficult 3D games.
Kim, Park, and Tjoe (2021)	The study aims, through a set of three robotics-coding activities and by building on students' prior geometric knowledge of measures of single angles (specifically, acute, right, obtuse, and straight angles), to introduce	The study involved 24 elementary school students (four 4th graders and 20 5th graders, nine males and 15	Sphero SPRK+, Sphero Edu app	The 24 students who enrolled in this two-week STEM summer school program participated in the current study voluntarily. Students' participation was part of the two-week, three-hours-per-day STEM summer school program that was led by two mathematics education faculty (the first and second authors) and four preservice elementary teachers, and was	Students in the present study discovered the concepts of special angle pairs in geometry—namely, complementary, and supplementary angles—as they learned to navigate the immediate feedback from the robot Sphero SPRK+ into a trial-and-error mathematics problem-solving process. Students' experiences in these three coding activities revealed, to a certain extent, that engaging in reflective play

	<p>elementary school students to the concepts of special angle pairs in geometry (namely, complementary and supplementary angles). to what extent would robotic coding activities interact with mathematical problem solving and critical thinking skills in the process of the development of new mathematical concepts in measures of complementary and supplementary angles at the elementary school level?</p>	<p>females, ages 9 to 10).</p>		<p>geared towards the Hispanic community in a southern state of the U.S. Students in the current study had little to no prior exposure to any computer programming activities. Specifically, they were not familiar with Sphero SPRK+ or the Sphero Edu app prior to the current study. A pre-test assessment was administered to students. Included in the pre-test assessment were elementary items in geometry involving measures of single angles (e.g., acute, right, obtuse, and straight angles) and those of special angle pairs (e.g., complementary and supplementary angles). It was expected that the 4th and 5th graders in the current study would not have been aware of the terms and concepts of complementary and supplementary angles, as those topics were part of the 7th grade common core state standards of mathematics. An assessment similar to the pre-test assessment was administered to students as the post-test assessment. Following the pre-test assessment and preceding the post-test assessment, students participated in three robotics-coding activities: driving, boomerang, and bowling.</p>	<p>could be shaped into meaningful teachable moments where students could participate in a doing with learning pedagogical method using educational robotics. These activities had transferability implications that might afford STEM learning access and opportunities for students to develop not only mathematical reasoning skills, but also problem solving and critical thinking skills operable to a coding environment.</p>
<p>Kwon, Kim, Shim, and Lee (2012)</p>	<p>Develop a programming tool through which students are able to experience procedural programming intuitively. It proposes a tool, Algorithmic Bricks</p>	<p>1st-grade elementary school students. 10 participants in an experimental group (A-Bricks), 14 in a control group (Scratch).</p>	<p>Algorithmics Bricks (A-bricks) and Scratch.</p>	<p>Quantitative. Five weeks of 10 sessions, each designed to last about 60 minutes. The experimental group (A-brick) used an actual robot to solve problems whereas the control group using Scratch solved problems by moving a virtual robot on the screen. Data 1) Pre survey of student logical thinking ability; 2) Post-test of convenience, understanding, and enjoyment; 3) Usability count of error occurrence; 4) Robot programming task.</p>	<p>The Scratch group asked more for help than the A-Bricks group and the A-bricks group rated understandability and usability higher than the Scratch group. Both groups enjoyed the programming activity. The Scratch group's problems appeared at level 3, while in the A-Bricks group's problems appeared at level 7.</p>

Louca, Zacharia, Michael, and Constantinou (2011)	The development of a framework for analysing and evaluating student-constructed models of physical phenomena through the use of computer-based programming.	Two classes of sixth-graders, 20 in each	Microworlds Logo (MW) - textual programming tool (TPT) and Stagecast Creator (SC) - graphical programming tool (GPT)	Qualitative/quantitative. The students of each class met once a week for 80 minutes, for a total of 7 months and always worked in pairs. In class 1, students used SC programming tool for developing computer-based models of the phenomenon studied, whereas in class 2, students used MW programming.	On a weekly basis, all the models that the students from each class (SC and MW) developed were collected, in total 220 models all versions included. In this study, only initial and final versions, which accounted for 80 model versions per condition. The SC group (graphical programming tool) were able to make programs that were more complex.
Miller (2019)	How mathematical patterns and structures can be promoted through engagement with coding.	132 students aged 7-8 years old; N =40 students in an experimental group; the rest constitutes the control group.	Scratch, coding robots (not defined)	Intervention-design. Pre- and post-tests, analysis of video data from the lessons.	The program focusing on coding had significant effect on students' capability to identify and generalise patterns and structures.
Moreno-León, Robles, and Román-González (2016)	Whether the use of programming with Scratch as an educational tool accelerates the learning curve, in terms of academic performance.	Three different schools, two 6th grade classes and one 2nd grade. In total three teachers and 129 students.	Scratch	Qualitative/quantitative. The participating teachers received a four-week training course in how to use Scratch as an educational tool. They then divided their classes into a control group and an experimental group, in a quasi-experimental design. Teachers designed a pre-test. The control group worked the unit without programming, while the experimental group included programming activities in Scratch. After at least four weeks of classes, a post test was taken.	The results from the three schools varied a lot. In two of the schools (grade 6), there were differences between pre- and post-test. The effect sizes varied between 1.19 and 0.06, the latter can be considered as a null effect. The computational thinking aspects varied between the different schools. However, students enjoyed working with Scratch, even though some of the schools' students found it harder than others.

Mouza, Marzocchi, Pan, and Pollock (2016)	How can equitable CS pedagogical practices be applied in after-school programs for middle school students? Does participation impact middle school learning of CS concepts, computational practices and attitudes?	Four CS undergraduates in their junior or senior year, partnered with a middle school teacher, to implement an after-school program for 52 students, 30 boys and 22 girls.	Scratch	Qualitative/quantitative. 9-week after-school CS program. 9 sessions, 90 minutes per session. Two groups - Group A, which had no prior experience of Scratch, and group B which included several students who had some prior experience with Scratch through introductory lessons in a similar program. Data a) observations of after-school program meetings; b) CS undergraduates' entries on their field experience; c) pre/post-tests of middle school students' learning of CS concepts, and attitudes and motivation; d) Scratch programs developed by students.	Statistically significant improvement from the pre- to the post test in six out of 10 questions: Handling an event, loop (forever), conditionals (if), conditional test, data (modifying variable) and parallelism (broadcast). A majority used sprites that exist in the platform. 71% of the projects were interactive, e.g., question/answer or the sprite is controlled to perform a certain task. 29% were non-interactive, e.g., the project could be described as a story with no information needed from the user. Most students showed evidence of appropriate code organisation and documentation. Positive changes in students' attitudes towards programming; but the changes were not statistically significant. Boys felt more confident with their computer skills than the girls did.
Nugent, Barker, Lester, Grandgenett, and Valentine, (2019)	What is the impact of the WearTec intervention on students' (a) knowledge of circuitry, programming, and engineering design and (b) self-efficacy in making an e-textile product? Are there differential effects for minorities and women?	1426 upper elementary students (808 treatment and 618 control) from 29 schools within a Midwestern state.	LilyPad Arduino	Quasi-experimental design with WearTec intervention in formal/informal learning environments on e-textiles, sewing, circuitry, microcontrollers and programming. Identical pre and post instruments were administered to students in both the treatment and control groups at baseline and following completion of the WearTec activities. The instruments included an assessment of wearable technologies self-efficacy and knowledge (circuitry, engineering design, and computer programming).	Results indicate that wearable technology's integration of engineering, computing, and aesthetics promises to be an excellent interdisciplinary context to support students' STEM learning and attitudes at the upper elementary level. The wearable technology instruction was found to promote learning of both males and females and under-represented and majority students when compared with a control group. But while self-efficacy of both male and female students was significantly higher than the control groups, results showed that males' confidence exceeded that of females in the areas of programming and circuitry.
Okita (2014)	Does initial knowledge building matter to the output carried out by an external source (i.e., robot or another human)?; (2) Does initial learning in specific environments	41 students: 26 in fifth grade and 15 in sixth grade between the ages of 9 to 11. No prior	LEGO Mindstorms NXT 2.0 and NXT-G software and ROBOTC. Physical platform:	Quantitative. Students in the high-transparency group learned to control robots using visual programming. Low-transparency group learned to control robots using syntactic programming. Learning phase 1: days 1–5, learning phase 2: days 6–10. 1 hour session for 10	Students who initially learned programming in a low-transparency environment (text-based) seemed to be able to transfer their knowledge to visual and text-based programming problems. Students who initially learned programming in a high-transparency environment had problems applying their knowledge to text-based

	continue to influence performance, after being exposed to alternative environments?; (3) Do benefits extend across different computing platforms?	experience or knowledge of robotics.	picture cards from "You Robot" and Ladybug Robot.	consecutive days, during which students learned to program robot movements using abstract concepts of speed, distance, and direction. Mid test at the end of day 5, and post-test at the end of day 10. The mid test included problems that measured performance on familiar materials and transfer. Post-test measured understanding of basic concepts after students experienced both high- and low-transparency learning environments and new problems.	problems. Even after being exposed to the low-transparency environment, the group that initially learned programming in a high-transparency environment, performed lower.
Portelance, Strawhacker, and Bers (2016)	Which ScratchJr programming blocks do young children choose to use after a tailored programming curriculum?	62 students: 21 students in kindergarten, 17 1st graders, 24 2nd graders	ScratchJr	Quantitative. 12 lessons to each class, twice a week for 6 weeks for a period of 1 h. Researchers taught each lesson. Textual representation of 977 projects written by the students. Researchers recorded open-ended field notes.	The students were dedicated to moving the character around the screen and hence the students used the Motion blocks most frequently.
Sáez-López, Román-González, and Vázquez-Cano (2016)	The main objective of the study is to analyse the benefits and possibilities of coding with a Visual Programming Language through projects and activities in primary education.	107 primary school students from grade 6 in five schools in Spain. 60.7% girls, 39.3% boys. Control group with 32 students.	Scratch	Quantitative. Application took place over two academic years in 20 one-hour sessions integrated in sciences and arts. Students created their own material based on the project goals. Visual Blocks Creative Computing Test (VBCCT) assessed computational concepts, and use of different commands. A questionnaire that analysed learning processes and students' attitudes, as well as observations.	The implemented program improved students' ability to understand programming concepts and logic. The students were enthusiastic, motivated, and happy to work with the approach. Understanding of computational concepts, an active approach through project-based learning, usefulness, motivation, and commitment underscore the importance of implementing visual programming through active methodologies in primary education.
Slangen, van Keulen, and Gravemeijer (2011)	What (key) concepts do pupils use to describe their knowledge, experiences and understanding of robotics? How does the pupils' conceptual understanding of robotics develop.	Six pairs of 11-12 years, last year of elementary. Three pairs were investigated during the whole trajectory.	LEGO Mindstorms NXT robots	Qualitative. Teaching experiment consisted of six video-taped 2-h lessons which were conducted during normal school hours but outside the classroom with pairs of pupils. The pupils, their mutual interactions, the construction and testing activities and the conversations with the researcher were focused. A webcam recorded the pupil's programming, while screen captures were made continuously with Camtasia 5.	At the beginning of the program, the students only used the 'move' blocks to create Reasoning-Acting programs. However, they developed their ability to see the systems as wholes as well as different subsystems along the experiment. Designing, constructing and programming a robot forced the students to relate the subsystems (e.g., the position of the sensor) to the whole construction. Most pupils quickly became familiar with the programming language.

					However, two or more synchronous actions appeared to be difficult, and they fell back to Reasoning-Acting loop thinking. Complex conditional reasoning (if... then...) were replaced with unconditional sequencing (first do this, then do that, do it for 10 seconds, etc.) of actions. The robot seemed to work as physical modelling which helped students to reason about how the robot should be programmed.
Su, Yang, Hwang, Huang, and Tern (2014)	Explore the effects of instructional tools (the ASP tool and the traditional tool) and programming instructional modes on Scratch programming performance.	135 sixth-graders (66 boys and 69 girls) mean age of 12 years old from four classes in Taiwan.	Scratch	Quasi-experimental: 6-week Scratch programming class. ASP-tool supported problem-solving instruction class, the ASP-tool-supported traditional instruction class, the traditional tool supported problem-solving instruction class, the traditional tool-supported traditional instruction class. Pre-test—prior knowledge test, and survey of prior computer backgrounds. Post-test—programming achievement test.	Students in the problem-solving-based teaching mode performed better than students in the traditional teaching mode did. In the traditional teaching mode, students' performance was not affected by the tool support. Students who received ASP tool support in conjunction with a problem-solving-based teaching approach performed significantly better than the other groups.
Sullivan and Bers (2016a)	Examination of young children's attitudes and ideas about technology and engineering products and performance on robotics and programming tasks.	18 K, 16 1st grade, 11 2nd grade	KIWI robotics	Quantitative. Participating children completed an eight-week robotics and programming curriculum in their classroom taught by research assistants from a university. Lessons were taught once a week and lasted ca. 1 hour. Pre- and post-test design.	Two tasks were carried out, "The Robot Parts Task" and the "Solve-Its tasks". In the simplest tasks, there was no statistically significant difference between grades, gender or the interaction between grades and gender. However, in the harder tasks, the boys' mean score was significantly higher than the girls' were.
Sullivan and Bers (2016b)	Evaluating a curriculum on robotics and its effects on basic robotics and programming skills	15 pre-K, 18 K, 16 1st grade, 11 2nd grade	KIWI (Kids Invent With Imagination) robotics	Quantitative. Over the course of 8 weeks, four classrooms completed an introductory robotics and programming curriculum taught by research assistants. Lessons were taught once a week and lasted ca. 1 h.	All students performed better for Sequencing tasks than for Repeat Loop and Conditional Statements Tasks. The mistakes they made were of different types; some were syntactical (logical programming errors), and others were story related (program made syntactical sense but did not match the sequence of the story). There was a statistically significant difference between different grades for the hard sequencing tasks, where older students performed better.

Sullivan and Lin (2012)	Examine the relationship of middle school students' perceptions of the ideal science student to their problem-solving activity and conceptual understanding in robotics.	Twenty-six 11- and 12-year-olds (22 boys) attending summer camp for academically advanced students.	LEGO Mindstorms robotics kit & Robolab software.	Videotaped activities from a summer camp for academically advanced 11- and 12-year-old students were analysed. The students were working on a problem in which they should complete individually the construction of a robot by adding sensors, wheels, and structural support. Then, they were asked to write a program that caused the robot to follow an angled black line on a paper track.	Five different domain specific problem-solving strategies used by the students were identified by the researchers, as well as four domain general strategies. The results also showed that students with a process-oriented view of being an ideal science student, for instance, being interested and motivated, were more likely to use domain specific strategies than students with a traits-based view of being an ideal science student; for example, being intelligent.
Sung, Ahn, and Black (2017)	The study investigates the effects of embodied activity and computational perspective-taking practice integrated in mathematics learning.	66 kindergarten and first-grade students enrolled in an after-school coding program	Scratch Jr.	Intervention with controlled randomised groups, pre- and post- and delayed tests to measure learning outcome.	One conclusion from the study is the importance of the activities the students perform, rather than relying solely on the use of programming software. Designing a learning environment that promotes explicit thinking processes is the key in deciding how to introduce programming. It is more powerful to prepare students in appropriate learning activities that support computational perspectives and practices and then expose them to digital tools.
Tan, Samsudin, Ismail, and Ahmad, 2020	The main aim of the presented research was to examine the interaction between student's gender and achievements in learning concepts of electricity via STEAM integrated approach utilising Scratch.	59 lower-secondary school students in Malaysia; 30 females and 29 males.	Scratch	Quasi-experimental design engaging male and female group were used in this research with the intention of studying the gender differences in learning concepts of electricity achievement. For both groups, the lessons on the electricity were taught using the STEAM integrated approach by using Scratch. A pre-test (Electricity Achievement Test, EAT) was carried out before the intervention to both groups. After the pre-test, they were given the intervention for two months from October to November. The intervention was carried out by using two lessons over one (1) week in three (3) months. Each lesson lasted for 90 minutes. Both groups underwent the	The findings proved that the STEAM integrated approach achievable via Scratch could narrow the gap between male and female in learning concepts of electricity. The findings of ANCOVA indicated that the intervention had similar positive effects on male and female students' achievement in learning concepts of electricity. This research also provided a new method and an alternate connective framework for learning concepts of electricity via art, and showed that both males and females were able to understand the topic of electricity, which reduced gender biases and disparity in the field of science.

				<p>same intervention, and the differences were the gender of the groups and taught by two different science teachers with a similar background in terms of qualification and years of teaching experience. Both groups of students were instructed with the STEAM integrated approach using Scratch. After the lessons, the students were required to design the games and animated stories based on the electricity topic by using Scratch. Also, a post-test was carried out to both groups after the intervention. The objective of the post-test was to identify the effect of the teaching approach in increasing students' learning concepts of electricity achievement.</p>	
--	--	--	--	--	--

ERIC + Google Scholar

ERIC

ERIC

Primary search: 60 + 27 peer-reviewed articles

+

Secondary search: 32 peer-reviewed articles

+

Third search: 55 peer-reviewed articles



Inclusion criteria applied to 174 articles



Application of criteria resulted in reduction by 149 articles → $\Sigma = 25$



Final selection of 25 qualitative and /or quantitative empirical studies about students' visual programming in one or more STEM subjects in grades 1 through 9 of general education, published in peer-reviewed journals 2011-2021.

Search terms: "Computational thinking" AND programming OR "programming languages" OR "visual programming" AND "elementary education"/"primary education"/"secondary education".

Search terms: "programming" OR "programming languages" AND "elementary education" OR different grades (e.g. DE "Grade 1" OR DE "Grade 2" etc.).

Search terms: "((DE "Programming" OR DE "Coding" OR DE "Computer Literacy") AND (DE "STEM Education")) AND ((DE "Secondary Education") OR (DE "Primary Education") OR (DE "Elementary Education"))

Only studies of grades 1 through 9 in general education; only students' visual programming; only studies of the STEM subjects; only finished empirical studies.