Does Computer Game Design and Programming Benefit Children? A Meta-Synthesis of Research

JILL DENNER and SHANNON CAMPE, Education, Training, Research (ETR)
LINDA WERNER, UC Santa Cruz

It is widely believed that there are educational benefits to making computer games, but there is no systematic review of research on this topic. This article describes a meta-synthesis of research on children designing and programming computer games that investigates the extent to which there is evidence of benefits for computer science learning and motivation. Over 400 articles were identified, and 68 articles met the inclusion criteria. A systematic analysis and synthesis across studies showed some evidence that computer game design and programming can lead to changes in programming knowledge, problem solving, and computer science attitudes and confidence. However, most of the evidence described engagement in computing-related practices and did not measure learning. The findings were mostly positive, although several studies noted more negative attitudes toward programming after making games. The results were similar across different pedagogical approaches, although social interaction may provide unique opportunities for computer science learning. The synthesis resulted in a list of design elements for studying computer game design and programming activities; these can be used to increase the availability of evidence about learning. The article concludes with the identification of gaps in the research and suggestions for additional research.

CCS Concepts: • **Social and professional topics → Computational thinking**; **K–12 education**;

Additional Key Words and Phrases: Computational thinking, computer science, game design and programming, K–12, meta-synthesis

ACM Reference format:
Jill Denner, Shannon Campe, and Linda Werner. 2019. Does Computer Game Design and Programming Benefit Children? A Meta-Synthesis of Research. *ACM Trans. Comput. Educ.* 19, 3, Article 19 (January 2019), 35 pages.
https://doi.org/10.1145/3277565

1 INTRODUCTION

In the last decade, the field of games and learning has exploded, with significant advances in leveraging students' interest in gaming for educational outcomes. Most of this work focuses on game play, but there has also been a dramatic increase in the number of tools and opportunities for children to learn to program computer games. Despite the growing excitement and intuitive logic that making a game results in learning, educators are increasingly challenged to justify the use of this

This work was supported by NSF (Award No. 1252276). The authors are grateful to help from Erica Marsh, David M Torres, and Julie Fasolas.
Authors' addresses: J. Denner and S. Campe, ETR, 100 Enterprise Drive, Suite G300, Scotts Valley, CA 95066; emails: {jill.denner, shannonc}@etr.org; L. Werner, CSE Dept, UC Santa Cruz, 1156 High Street, Santa Cruz, CA 95064; email: llwerner@ucsc.edu.

1946-6226/2019/01-ART19 $15.00
https://doi.org/10.1145/3277565

approach in their classrooms. This article presents a systematic review of research on teaching children to design and program their own computer games.

Computer game design and programming (CGD&P), when used for education, has its roots in research on novice programming. This work is based in constructionism, a theory of learning, teaching, and design where learning involves constructing relationships between old and new knowledge while building artifacts and interacting with others [37, 52]. In the late 1980s, there was a flurry of interest in teaching children to program, spurred on by the availability of tools like Logo [53, 72]. Despite the initial excitement about the educational benefits of computer programming, the research was not conclusive and both technical and training barriers prevented schools from adopting this approach. More recently, new programming tools have become widely available and there is renewed interest in and attention to using CGD&P for educational purposes [40].

Children designing and programming games is a subset of a growing movement to teach children to program (or code) computers. Yasmin Kafai's foundational book on CGD&P for learning, Minds in Play [38], described the process and learning benefits for children programming computer games; since then, she has written extensively about this approach for education. Despite the great promise of CGD&P for learning, it was not widely used over the next decade, due to limited access to child-friendly software and resistance by schools [39]. In the last decade, there has been a resurgence of efforts to teach children to program games because of the availability of tools usable by children. For example, in 2007 the MIT Media Lab released the Scratch programming language, which aimed to make programming accessible to everyone [60]. This 2D tool can be used to create animated stories, interactive narratives, multimedia art projects, and games, all of which can be shared and modified in an online community. Some of these tools aimed to foster children's programming skills, while others focused more on creating space for creativity, so they ranged in the extent to which they require programming, from not at all (e.g., Stencyl, Gamestar Mechanic) to some (e.g., Alice, Agent Sheets, Kodu Game Lab) [25, 79] to a lot (e.g., Unity, Greenfoot, GameMaker Language) [29]. There are now at least a hundred child-friendly programming languages available, and computer game programming is increasingly being used as an educational strategy to teach and motivate youth, as well as to attract children to computing fields. Gee & Tran [28] provide a description of some of the most popular tools used to make games and highlight some of the larger game-making projects.

CGD&P is widely believed to be an effective strategy for K–12 education, and common assumptions are that it can have a measurable impact on both knowledge and attitudes. For example, CGD&P has been used with the goal of helping students practice and learn core math, science, storytelling, and logic skills [12]. Other goals include engaging students from underrepresented groups in making games in order to increase their interest and confidence to program computers [16, 17]. In a recent review of almost 500 publications, Earp [20] concludes that the most common focus of game-making studies is on individual learning attributes (e.g., computational thinking and creativity), followed by curriculum (including a range of academic content), and technology practices (e.g., programming).

But what are the benefits of CGD&P? Previous research reviews conclude that the benefits of CGD&P for learning include engagement in computational practices, increased knowledge of computational concepts, deeper engagement in thinking and problem solving, more positive attitudes toward computers and learning, having a creative outlet, and building academic content knowledge and skills [10, 31, 76]. For example, Kafai and Burke [40] review some of the literature and state that "making games is a compelling context for learning computational concepts, practices, and broadening perspectives across ages, tools, and settings." However, none of these reviews have included a systematic synthesis, which takes a critical look at the research methods, participants,

pedagogy, limitations of the conclusions as stated by the original authors, or the extent to which assumptions about benefits are supported by research evidence.

In addition to the research reviews mentioned above, widely held beliefs about the benefits of CGD&P are also articulated in opinion pieces. Commentaries appear on sites like GeekWire, SiliconBeat, BrainPopEducators, and CommonSenseMedia. These typically claim that making a game brings scientific concepts to life, allows the student to become an active participant in their own learning, and challenges the programmer to make the game fun for the player. For example, online commentaries in blog posts and news articles reveal that CGD&P is believed to help children develop skills that prepare them for high-paying jobs that include computer programming, persistence with problem solving, video game design, teamwork, and creating models and software architecture. In addition, CGD&P is viewed as highly motivating because video games are very popular with most children; the organizers of game-making competitions claim that they are making academic subjects more fun, and that their participants learn by making games.

Despite the widespread enthusiasm about CGD&P, what remains unclear is the extent to which there is support for each of these widely held assumptions about the benefits of CGD&P, who benefits, and under what conditions. In addition, there is great variation in how CGD&P is taught including the duration, the setting, students' prior experience, and the pedagogy used. No existing literature reviews have taken these factors into consideration when summarizing results across studies.

This systematic synthesis is designed to identify the conditions under which there are benefits, who benefits, and the range of evidence provided. The focus is on research that examines outcomes related to learning and motivation as a result of children designing and programming computer games. The goal is to examine whether the excitement about the broad educational benefits of game programming has substantive research-based evidence, and to identify both research and education opportunities for the field of games and learning, including the conditions or pedagogies with which CGD&P is most likely to lead to learning. The following question is addressed: To what extent is there evidence to support widely held assumptions about the benefits of CGD&P?

2 META-SYNTHESIS APPROACH

The accumulation, organization, and synthesis of research-based knowledge is critical for advancing the field and contributing to educational practice [69]. A systematic analytic methodology goes beyond a literature review to compare and contrast findings. Methods for conducting research syntheses must meet or exceed the level of methodological rigor of the studies they are aiming to synthesize [14]. There are many complexities involved in synthesizing research, particularly when there is diversity across studies in their methods, disciplinary or theoretical frameworks, and epistemologies. However, the American Educational Research Association (AERA) recognizes that the benefits of pulling together these sources outweigh the challenges of including a range of methodological and ideological approaches, and frameworks are emerging for increasing the rigor of a methodologically inclusive synthesis [73].

Meta-synthesis is a systematic approach to integrating the results of studies that use a range of methods [73]. The science of conducting a rigorous synthesis of mixed methods research is still developing, and much of this research comes from the field of healthcare [57, 66] with the addition of some more recent articles on the application to education and social science [46, 58, 74]. Our approach uses recent advances in methods for research synthesis that start with pre-defined research questions and concepts. Our goal was to integrate findings, to "create taxonomies of the range of conceptual findings and provide the foundation for the development of conceptual descriptions of phenomena across studies" [65]. A meta-synthesis is an integrative method that is based within a philosophical framework that views knowledge as both socially and culturally constructed [66].

A meta-synthesis is also an interpretive integration of qualitative and quantitative research findings based on a systematic literature search and analysis process [65].

A preliminary review of the existing literature led us to choose a meta-synthesis approach. In particular, we concluded that an aggregate study of frequencies and relative weight of themes was not possible because of the range of outcomes, epistemological foundations, and measurement approaches, as well as the small sample sizes in the studies. Thus, the goal was to identify meta-themes and taxonomic groupings. Whittemore and Knafl [87] suggest that an integrative review "is the only approach that allows for the combination of diverse methodologies" (p. 546).

3 METHOD

3.1 Procedure

The procedure followed Cooper's [14] seven steps for conducting a research synthesis: formulate the problem, search the literature, extract information, evaluate the quality and importance, analyze and integrate the findings, interpret the evidence, and present the results. At each step, we addressed Suri and Clarke's [74] three guiding principles for a quality research synthesis: informed subjectivity and reflexivity, purposefully informed selective inclusivity, and audience-appropriate transparency. Four people were involved to various degrees at each phase of the work. The goal at every step was to minimize bias by including more than one person in the process, making epistemological assumptions explicit, and being transparent about key decisions such as search strategies, inclusion and exclusion criteria, and classification procedures. The transparency allows the reader to make their own decisions about the implications of our findings. Additional detail about our synthesis process can be found in Denner, Marsh and Campe [58].

3.1.1 Formulate the Problem. Based on our preliminary literature review, the initial goal of the synthesis was to understand the benefits to children of doing computer game programming, although the focus of the synthesis evolved over time, as described above. As is common in integrative meta-synthesis projects, the goals were revised in response to the findings that emerged from a series of analyses. The original goal was to determine the benefits of CGD&P for children, to identify the mechanisms through which students benefit (or do not benefit) from CGD&P, and to generate new insights related to the most effective approaches for teaching and studying children doing CGD&P. The following were the initial research questions: (1) What kinds of children's learning or motivation is CGD&P best suited to address? (2) What kinds of students benefit the most? (3) What is known about effective pedagogy and teacher training? (4) What social and technical learning environments are most beneficial? (5) Where are the research gaps?

3.1.2 Search the Literature. Several strategies were used to locate articles that described the benefits of computer game programming in K–12th grade children. We searched databases for journal articles, books and book chapters, conference proceedings, and unpublished project reports from multiple disciplines (e.g., education, learning sciences, psychology, computer science (CS) education). Key databases were identified based on the authors' knowledge of the field. These included Google Scholar, ACM Digital Library, Academia.edu, and ResearchGate. The original keywords used in the search were: game programming, children game programming, pair programming games, video [computer] game design, video [computer] game programming; video [computer] game making; video [computer] game development; video [computer] game construction; video [computer] game creation. The searches yielded articles that included studies of non-digital game making so additional searches were narrowed down with the following keywords: digital, computer, and video. In addition, we included search terms that aimed to weed out studies of college students: novice, novice programmer, and schools. We also read through key conference

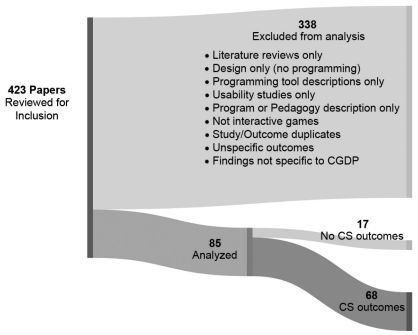

Fig. 1. Steps of article inclusion and exclusion.

programs for relevant articles, including Games, Learning, and Society, Foundations of Digital Games, and the European Conference on Games-Based Learning. We also searched research registries of US National Science Foundation (NSF)-funded projects to find studies in progress and unpublished results or project reports, thus minimizing publication bias.

To locate articles that were not published or did not come up in response to our keyword search, articles and people were also located by searching reference lists of articles we had already retrieved, using ancestry searches: the systematic review of citations. Requests were sent to authors who had published articles about CGD&P, following the "Invisible College" method of collecting input from experts that are geographically dispersed [13] and asked for their evaluation reports, conference presentations, and unpublished articles. A spreadsheet of names was used to track who was contacted, when, and information about what, if anything, each of them sent. The list of 105 names was generated from the existing pool of authors from whom we already had articles, as well as the Principal Investigators of NSF-funded projects that included game programming. Each was emailed a letter explaining the goal of the synthesis and asking for copies of articles they have written, links to anything they have written on the topic, and any other pertinent references and links. Articles that were identified through this mechanism were added to our database and evaluated along with the others to determine whether they met our criteria for inclusion in the analysis phase.

We did not search for or retain Master's theses or other unpublished student articles; we did include PhD level theses because they were more likely to have undergone a rigorous review. The last search was conducted in January 2016, so the most recent articles are from 2015. There was no limit on how old the article was, as long as it met the criteria. The literature search yielded over 400 articles about K–12th grade students; the articles included a combination of academic/research and educator perspectives. Figure 1 shows the number of articles collected, analyzed, and reasons for exclusion described in more detail.

3.1.3 Extract Information. The initial 423 articles were briefly reviewed to determine whether they fit our criteria to be retained for the next phase of the synthesis. Two people read each articles methods section to determine whether it was an original study of children designing and programming games, and whether there was research evidence. We define game programming in an educational setting as where students are instructed to create or change the interactive behavior of a computer program in which rules determine the actions of the game player and goals determine whether the game player's actions result in a win or loss. Our definition builds on Juul's definition of games from his Dictionary of Video Game Theory where "A game is a rule-based system with a variable and quantifiable outcome, where different outcomes are assigned different

values, and the player exerts effort in order to influence the outcome. The player feels emotionally attached to the outcome, and the consequences of the activity are optional and negotiable" [36].

Based on this initial review, 208 articles were dropped from the study because they were not studies of children programming games. Most were descriptions of programming tools and/or pedagogy or program implementation without any data collected (92), literature reviews and commentaries (27), or studies of children designing but not programming games (16). A total of 215 articles were retained after this initial review.

3.1.4 Evaluate. The 215 articles were reviewed in more detail to determine whether they met our criteria for the analysis phase of the synthesis. Each article was read in its entirety at least three times and the following information was entered into a database: participant demographics, class length and approach, and measured outcomes (e.g., academic learning or interest, and CS learning or interest). The details were reviewed by two researchers to determine which articles to retain. Our inclusion criteria were based on that of Thomas & Harden [77]. First, the study needed to be relevant (some or all of the participants were in kindergarten through 12th grade, primary through secondary school), and the findings said something about the benefits of CGD&P. The second criterion was that the article contained sufficient detail about the participants (so it was clear which results were for children), and the procedures (so it was clear that the students made computer games). The third criterion was relevant for studies that produced multiple articles on a specific outcome from the same sample and usually by the same authors. We included the article with the greatest relevance and the most detail as determined by the extent to which the methods and results were clear and supported the conclusions.

Based on these criteria, 130 articles were excluded from further analysis. The most common reasons were that: (a) the article did not include a description of targeted outcomes or data (e.g., it only described how children made their games), (b) it was unclear whether or the extent to which children made interactive games (rather than animations), and (c) there was another article from the same study and sample that was a more comprehensive description of the sample, methods, or results. Also in this excluded group were articles that, upon further review, were found to only include participant feedback on tools or the process and/or descriptions of the program or what participants created.

3.1.5 Analyze. This left a total of 85 articles that described benefits of CGD&P and met our quality criteria. The studies involved a wide range of age groups, programming tools, amounts of time spent on CGD&P, different ways of defining and measuring different outcomes, and a range of research methods. This variation made it important to use an integrated meta-synthesis approach to analysis, as described earlier. For the purposes of this article, we dropped 17 articles that measured only non-computer science related but academic outcomes (e.g., math, science).

Table 1 describes the characteristics of the 68 studies that were retained for the analysis phase. Most included both girls and boys, and race/ethnicity was not reported in two-thirds of the articles. Studies ranged in size and methods, although the most common methods were qualitative. The research took place either in school or outside of regular school hours (e.g., afterschool, summer, weekends—at school or community sites) but rarely in both. Of the 68 articles, 11 were unpublished PhD dissertations, reports, or articles. This group of studies differs from Earp's [20] review, where the majority of studies were in formal educational settings, and where 54% had sample sizes of 1–50. However, Earp [20] based the numbers on the article Abstracts, and only 17% of the articles he reviewed identified the number. Table 1 does not include information about age or grade level, due to challenges (also reported by Earp [20]) of inconsistent reporting, which includes age ranges and grade levels that vary across countries. However, the majority are in what is considered middle school in the U.S.

Table 1. Demographics and Methods for Analyzed Articles (N = 68)

Gender of Participants		Race/Ethnicity of Participants*		Number of Participants		Methods Used		Setting of Study**	
Female & Male	42	Hispanic	20	N < 30	22	Qualitative	33	In school only	30
Female only	6	White	16	N = 30-50	7	Quantitative	24	Out of school only	21
Male only	2	African American	15	N > 50	37	Mixed	11	In/out of school	6
Unclear	18	Asian	6	Unclear	2			Unclear	11
		Native American	5						
		Other	7						
		Unclear	43						

*For articles of which race/ethnicity was indicated, participants are counted for each race/ethnicity (not exclusive categories).
**Out of school includes after school, summer, weekend, camp, and community.

The analysis required careful reading of articles that others had cited to justify their own claims about benefits. This process revealed methodological weaknesses, logical inconsistencies, and conclusions that were not always supported by the data presented in the article. For example, many research reports failed to include the kind of information needed to answer the original questions, such as a description of the sample, pedagogy, or how the outcomes were measured. Thus, our focus shifted to describing the extent to which studies provide evidence of the widely held beliefs. The synthesis presented here is organized around the following question: To what extent is there evidence to support widely held assumptions about the benefits of CGD&P?

The next step of the analysis was a critical review of each articles findings to determine the type and extent of evidence to support the authors' conclusions and to begin to create conceptual descriptions of the benefits across articles. For each article, one researcher initially generated a brief summary of the findings and this summary was discussed with a second researcher and revised as needed. This involved an iterative process of generating brief descriptions of the outcomes of a few articles, looking across them to identify repeating themes and key groupings, as well as conflicting findings within each outcome, and then revising the descriptions based on a review of more articles. This process required that the results and discussion sections of each article be read at least three times. When there were disagreements between researchers in the conclusions that could be reached about each article, the conclusions were discussed with the larger project team until consensus was reached. We agreed that the findings would not be "weighted" based on methods or type of evidence; once a article was included in the analysis phase, it was considered strong enough to be included. Thus, a large quantitative study of learning did not count more than a small, in-depth descriptive study. This integrative and critical approach revealed two types of meaningful taxonomies for interpreting the results: evidence and pedagogy.

The analysis revealed that it was conceptually important to distinguish between different kinds of evidence: change evidence and descriptive evidence. For example, some studies measured changes in participants' interests, attitudes, or learning as a result of doing CGD&P. Measures of change included several types of data: pre–post surveys; in-depth interviews or post surveys where students reported how they had changed; detailed observations over time; and post-only surveys with the inclusion of a comparison group. Studies that provided this type of evidence described learning or changes in beliefs or attitudes. The other kind of evidence was descriptive and included details about what students were doing, or what students reported knowing (at one point in time only). These measures included students' games, post-test surveys, observations at one time point, and interviews that did not ask about change. The distinction between descriptive and change evidence is consistent with the view that learning occurs as a result of participating in science or engineering practices, including the development of repertoires of practice [56].

Analyzing across the studies also revealed the importance of distinguishing between different approaches to teaching CGD&P. Some studies used structured curriculum while others encouraged students to learn by exploring; and some required a certain kind of game content or mechanic while others allowed students to choose. There was also variation in whether and how they incorporated social interaction: the approaches included pair programming, team game design and programming, and peer play-testing and feedback. There was also a range in the extent to which students reflected on what they were doing; and variation in the tools used, the settings, the number of hours spent making games; and in the extent to which CGD&P was the sole focus, or part of a larger set of activities. In addition, there was a range of goals or reasons for teaching CGD&P.

4 RESULTS

This section describes the results of our meta-synthesis focused on the question: To what extent is there evidence to support widely held assumptions about the benefits of CGD&P? As described earlier, this research question emerged during our meta-synthesis process, when we realized that there was not enough evidence to answer our original question about the benefits of CGD&P. The findings show the extent to which research evidence supports some of the widely held beliefs about the benefits of CGD&P for children in their CS learning, attitude, and confidence.

The answer to the question about the extent to which there is evidence to support widely held beliefs cannot be captured on a linear continuum, where evidence is ranked from weak to strong. Instead, the answer is more nuanced; the studies provide certain kinds of evidence for certain kinds of benefits, and under specific circumstances. This approach is in line with a realist philosophy in which the focus is not on whether an intervention works, but for whom, how, and in what contexts [71]. The findings highlight the amount and the type of evidence to support the belief that CGD&P has benefits for children's CS learning and motivation.

The results are summarized in Table 2. The table indicates whether each article provided evidence of different kinds of benefits. Positive benefits (+) means there was evidence that CGD&P did benefit students. Negative or neutral (−) benefits means that either the data indicated a negative result (e.g., attitudes toward computing became more negative, or scores on a math test declined), or the study did not provide evidence of a benefit of CGD&P (e.g., no significant difference between students who did and did not make games). Some articles included evidence of both positive results and either negative or neutral results (+/−). The tables distinguish between types of evidence (change vs. description) and the pedagogical approach used to teach CGD&P (when indicated in the article).

Table 2 also provides details about each study, such as the duration, setting/location, software/tool, the methods used to collect and analyze data, and the participants. There was great variation in the programming language or environment (software/tool) used, from more constrained and very game-specific (e.g., Gamestar Mechanic) to less constrained (e.g., Alice, Scratch) that allows students to create many things with the software besides making games. There was also variation in the methods; they were labeled as quantitative when they included collecting data using surveys, logging files, and performance assessments; or qualitative if data collection was more contextualized or required an inductive process to analyze it, such as analysis of games, observation data, and interviews. Methods were labeled as mixed if they included both qualitative and quantitative data.

In the next section, we describe the findings from the meta-synthesis. The summaries include the type of evidence across studies within each type of outcome, the extent to which there were common and generalizable findings, and whether the evidence varies across different kinds of students and pedagogical approaches. Examples from original studies are included to support our conclusions, but page limits prevent the inclusion of detail about each study.

Table 2. Study Details, Outcomes, and Pedagogies

				Outcomes								Pedagogy		
				Programming Knowledge		PS&D		CS Attitude		CS Confidence				
Author(s)	Year	Program & Study Details	Participant Details	CE	DE	CE	DE	CE	DE	CE	DE	DBT	SWI	SI
Adams & Webster	2012	Program Length: Unclear duration and hours Setting: Camp; US Software/Tool: Alice, Scratch Methods: quantitative	N: 322 Age/Grade: 11–14 years old Gender: Mixed sex Race/Ethnicity: Unclear Prior tech experience: Unclear	+								U	U	U
Adamson, Lowe, Hoyles, Tholandar, & Noss	2002	Program Length: 4 sessions; 4 total hours; additional one-day workshops Setting: Unclear; England & Sweden Software/Tool: ToonTalk Methods: qualitative	N: 9 Age/Grade: Elementary school Gender: Mixed sex but more girls Race/Ethnicity: Unclear Prior tech experience: Unclear		+								✓	✓
Akcaoglu	2013	Program Length: 8 sessions; 40 total hours (summer groups); 4–5 sessions; 15–16 total hours (school year groups) Setting: Unclear; U.S. & Turkey Software/Tool: Kodu Methods: quantitative	N: 97 (73 tx; 24 control) Age/Grade: 11–13 years old Gender: Mixed sex Race/Ethnicity: Unclear Prior tech experience: Unclear			+							✓	
Akcaoglu, Boyer, & Kereluik	2012	Program Length: 10 days; hours unclear Setting: In School and Summer; U.S. Software/Tool: Kodu Methods: quantitative	N: 21 Age/Grade: 12–15 years old Gender: Mixed sex Race/Ethnicity: Unclear Prior tech experience: Yes	+				+		+		✓		
Akpinar & Aslan	2015	Program Length: 9 weeks; 12 total hours Setting: In School; Turkey Software/Tool: Scratch Methods: qualitative	N: 30 Age/Grade: Middle school Gender: Unclear Race/Ethnicity: Unclear Prior tech experience: Range		+							✓		

(Continued)

Table 2. Continued

				Outcomes								Pedagogy		
				Programming Knowledge		PS&D		CS Attitude		CS Confidence				
Author(s)	Year	Program & Study Details	Participant Details	CE	DE	CE	DE	CE	DE	CE	DE	DBT	SWI	SI
Al-Bow, Austin, Edgington, Fajardo, Fishburn, Lara,...,& Meyer	2009	Program Length: 10 days; 80 total hours Setting: Summer; U.S. Software/Tool: Greenfoot Methods: quantitative	N: 27 Age/Grade: 14–15 years old (9th/10th grade) Gender: All girls Race/Ethnicity: White/AA/Hispanic/Other Prior tech experience: Unclear	+				+		+		✓		✓
Anton, Harris, Oshner, Salmon, Rothschild, Berland, & Squire	2013	Program Length: 2–4 weeks; 45–90 mins/session; unclear hours Setting: Unclear; U.S. Software/Tool: Kodu Methods: quantitative	N: 76 Age/Grade: Unclear Gender: Unclear Race/Ethnicity: Unclear Prior tech experience: Unclear	+									✓	
Basawapatna, Koh, & Repenning	2010	Program Length: 1–2 weeks; unclear hours Setting: In school; U.S. Software/Tool: AgentSheets Methods: qualitative	N: 197 Age/Grade: Middle school Gender: Mixed sex Race/Ethnicity: Unclear Prior tech experience: Unclear		+				+				✓	
Baytak	2009	Program Length: 23 days; unclear hours Setting: In school; U.S. Software/Tool: Scratch Methods: qualitative	N: 10 Age/Grade: 5th grade Gender: Mixed sex Race/Ethnicity: Unclear Prior tech experience: Yes		+			+				✓		✓
Baytak, Land, Park & Smith	2008	Program Length: Unclear duration; 12 total hours Setting: In school; U.S. Software/Tool: GameMaker Methods: qualitative	N: 10 Age/Grade: 5th grade Gender: Mixed sex Race/Ethnicity: Unclear Prior tech experience: Unclear	+								✓		✓
Campe, Werner, & Denner	2005	Program Length: 12 weeks; 48 total hours Setting: After school; U.S. Software/Tool: Flash MX Methods: qualitative	N: Unclear Age/Grade: Middle school Gender: Unclear Race/Ethnicity: Unclear Prior tech experience: Unclear		+							✓		✓

(Continued)

Table 2. Continued

				Outcomes								Pedagogy		
				Programming Knowledge		PS&D		CS Attitude		CS Confidence				
Author(s)	Year	Program & Study Details	Participant Details	CE	DE	CE	DE	CE	DE	CE	DE	DBT	SWI	SI
Carbonaro, Szafron, Cutumisu, & Schaeffer	2010	Program Length: Unclear duration; 12 total hours Setting: Workshop at University and In school; U.S. Software/Tool: ScriptEase, Aurora Toolset Methods: qualitative	N: 50 Age/Grade: High school Gender: Mixed sex Race/Ethnicity: Unclear Prior tech experience: No		+								✓	
Denner & Bean	2005	Program Length: Unclear duration and hours Setting: After school and Summer; U.S. Software/Tool: Flash Methods: mixed	N: 126 Age/Grade: 10–14 years old Gender: All girls Race/Ethnicity: White/Hispanic Prior tech experience: Range		+	+				+		✓		✓
Denner, Werner, Campe, & Ortiz	2014	Program Length: 1 semester; 20 total hours Setting: In school and After school; U.S. Software/Tool: Storytelling Alice, Alice 2.2 Methods: quantitative	N: 320 Age/Grade: 10–14 years old Gender: Mixed sex Race/Ethnicity: White/Hispanic Prior tech experience: Unclear	+				+		+		✓	✓	✓
Denner, Werner, & Ortiz	2011	Program Length: 2 sessions/week (school year) across 14 months; 3 weeks Setting: After school and Summer; U.S. Software/Tool: Stagecast Creator Methods: qualitative	N: 59 Age/Grade: Middle school Gender: All girls Race/Ethnicity: White/Hispanic Prior tech experience: No		+								✓	
El-Nasr & Smith	2006	Program Length: 9 days; 9 total hours Setting: Summer; U.S. Software/Tool: WarCraft III Methods: qualitative	N: 20 Age/Grade: High school Gender: Unclear Race/Ethnicity: Unclear Prior tech experience: Unclear		+								✓	

(Continued)

Table 2. Continued

				Outcomes								Pedagogy		
				Programming Knowledge		PS&D		CS Attitude		CS Confidence				
Author(s)	Year	Program & Study Details	Participant Details	CE	DE	CE	DE	CE	DE	CE	DE	DBT	SWI	SI
El-Nasr, Yucel, Zupko, Tapia, & Smith	2007	Program Length: 1 week; unclear hours Setting: Summer; U.S. Software/Tool: RPG Maker XP Methods: mixed	N: 25 Age/Grade: Middle school Gender: All girls Race/Ethnicity: Unclear Prior tech experience: Range		+				+	+			✓	
Ernst & Clark & Smith	2012	Program Length: Unclear duration; 8 total hours Setting: Virtual school; U.S. Software/Tool: GameMaker Methods: mixed	N: 28 Age/Grade: Middle and High school Gender: Unclear Race/Ethnicity: Unclear Prior tech experience: Unclear		+			+/−					✓	
Feng & Chen	2014	Program Length: 8 weeks; 5 total hours Setting: In school; Taiwan Software/Tool: Scratch Methods: mixed	N: 232 Age/Grade: 11–12 years old Gender: Mixed sex Race/Ethnicity: Unclear Prior tech experience: No	+		+							✓	
Foster, El-Nasr, & Maygoli	2011	Program Length: 22 weeks; 33 total hours (Canada workshop); 12 weeks, 48–72 hours (each of 3 Mexico workshops) Setting: After school; Canada & Mexico Software/Tool: 1) Scratch, 2) Scratch & GameMaker Methods: qualitative	N: 62 Age/Grade: 9–19 years old Gender: Mixed sex Race/Ethnicity: White/Hispanic/Asian/Other Prior tech experience: Unclear		+								✓	
Fowler	2012	Program Length: Unclear duration and hours Setting: In school; U.S., U.K., & New Zealand Software/Tool: Kodu Methods: quantitative	N: 60 Age/Grade: Middle school Gender: Mixed sex Race/Ethnicity: White/AA/Asian/Native American/Other Prior tech experience: Yes					+					✓	

(Continued)

Table 2. Continued

				Outcomes								Pedagogy		
				Programming Knowledge		PS&D		CS Attitude		CS Confidence				
Author(s)	Year	Program & Study Details	Participant Details	CE	DE	CE	DE	CE	DE	CE	DE	DBT	SWI	SI
Fowler & Cusack	2011	Program Length: 1 school term; 3 hours/week; unclear hours Setting: In school; New Zealand Software/Tool: Kodu Methods: quantitative	N: 49 Age/Grade: Middle school Gender: Mixed sex Race/Ethnicity: White/Asian/Other Prior tech experience: Range					+					✓	
Games & Kane	2011	Program Length: 3 months; unclear hours Setting: After school; U.S. Software/Tool: Gamestar Mechanic/Kodu; Flash Actionscript Methods: qualitative	N: 36 Age/Grade: 13–15 years old Gender: Mixed sex Race/Ethnicity: Unclear Prior tech experience: Unclear			+						✓	✓	✓
Hadad	2013	Program Length: 6 weeks; 27 total hours Setting: Summer; U.S. Software/Tool: Scratch, Xbox 360 Kinect Methods: mixed	N: 14 Age/Grade: High school Gender: Mixed sex Race/Ethnicity: Hispanic Prior tech experience: Unclear	+				+					✓	✓
Howland & Good	2015	Program Length: Unclear duration; 16 total hours Setting: In school; England Software/Tool: Flip programming language Methods: mixed	N: 55 Age/Grade: Middle school Gender: Mixed sex Race/Ethnicity: Unclear Prior tech experience: Range	+								✓		
Hwang, Hung, & Chen	2013	Program Length: Unclear duration; 21 total hours Setting: In school; Taiwan Software/Tool: Kodu Methods: mixed	N: 167 (82 tx; 85 control) Age/Grade: Middle school Gender: Unclear Race/Ethnicity: Unclear Prior tech experience: Unclear			+							✓	✓

(Continued)

Table 2. Continued

				Outcomes								Pedagogy		
				Programming Knowledge		PS&D		CS Attitude		CS Confidence				
Author(s)	Year	Program & Study Details	Participant Details	CE	DE	CE	DE	CE	DE	CE	DE	DBT	SWI	SI
Ioannidou, Repenning, & Webb	2009	Program Length: Unclear duration; less than 5 hours Setting: After school; U.S. Software/Tool: AgentCubes Methods: mixed	N: 40 Age/Grade: Middle school Gender: Mixed sex Race/Ethnicity: White/AA/Hispanic/Asian/Native American/Other Prior tech experience: Unclear		+		+						✓	
Jenson & Droumeva	2015	Program Length: Unclear duration; 15 hours Setting: In school; Canada Software/Tool: GameMaker Methods: quantitative	N: 67 Age/Grade: Middle school Gender: Mixed sex Race/Ethnicity: Unclear Prior tech experience: Range	+									✓	✓
Kafai	1995	Program Length: Unclear duration; 112 total hours Setting: In school; U.S. Software/Tool: Logo Methods: mixed	N: 16 Age/Grade: Elementary school Gender: Mixed sex Race/Ethnicity: White/AA/Hispanic/Other Prior tech experience: Yes	+								✓		✓
Ke	2014	Program Length: 6 weeks; 12 total hours Setting: Unclear; U.S. Software/Tool: Scratch Methods: qualitative	N: 64 Age/Grade: Middle school Gender: Mixed sex Race/Ethnicity: Hispanic/Native American Prior tech experience: Unclear		+							✓		✓
Ke & Im	2013	Program Length: 6 weeks; 12 total hours Setting: Unclear; U.S. Software/Tool: Scratch Methods: qualitative	N: 64 Age/Grade: Middle school Gender: Mixed sex Race/Ethnicity: Hispanic/Native American Prior tech experience: Unclear		+							✓		✓

(Continued)

Table 2. Continued

				Outcomes								Pedagogy		
				Programming Knowledge		PS&D		CS Attitude		CS Confidence				
Author(s)	Year	Program & Study Details	Participant Details	CE	DE	CE	DE	CE	DE	CE	DE	DBT	SWI	SI
Lakanen, Isomöttönen, & Lappalainen	2014	Program Length: 5 days; 25 total hours Setting: Summer; Finland Software/Tool: Jypeli Methods: quantitative	N: 462 Age/Grade: 12–18 years old Gender: Mixed sex Race/Ethnicity: Unclear Prior tech experience: Range					+/−					✓	
Li	2010	Program Length: 2 weeks; 7 total hours Setting: Summer; Canada Software/Tool: Scratch Methods: qualitative	N: 21 Age/Grade: 7–11 years old Gender: Mixed sex Race/Ethnicity: Unclear Prior tech experience: Unclear			+				+		✓		
Lin, Yen, Yang, & Chen	2005	Program Length: Unclear duration; 30 total hours Setting: In school; Taiwan Software/Tool: Stagecast Creator, HANDS, Visual Basic Methods: qualitative	N: 81 Age/Grade: Elementary and Middle school Gender: Mixed sex Race/Ethnicity: Unclear Prior tech experience: Yes		−				+				✓	
Martin, Walter, & Barron	2009	Program Length: 2 weeks; 10 total hours Setting: Unclear; U.S. Software/Tool: AgentSheets Methods: qualitative	N: 28 Age/Grade: 11–13 years old Gender: Mixed sex Race/Ethnicity: Unclear Prior tech experience: Range		+				+				✓	✓
Maxim, Grosky, & Baugh	2007	Program Length: 1 week; unclear hours Setting: Unclear; U.S. Software/Tool: Game Factory & component libraries Methods: quantitative	N: 11 Age/Grade: Middle and High school Gender: All boys Race/Ethnicity: Unclear Prior tech experience: Unclear							+/-			✓	
Minnigerode	n.d.	Program Length: 1–2 school years; 75 minutes each day; unclear hours Setting: In school; U.S. Software/Tool: Flash actionscript Methods: quantitative	N: 162 Age/Grade: Middle school Gender: Unclear Race/Ethnicity: AA/Hispanic Prior tech experience: Unclear					+				✓		✓

(Continued)

Table 2. Continued

				Outcomes								Pedagogy		
				Programming Knowledge		PS&D		CS Attitude		CS Confidence				
Author(s)	Year	Program & Study Details	Participant Details	CE	DE	CE	DE	CE	DE	CE	DE	DBT	SWI	SI
Minnigerode	2013	Program Length: Unclear duration and hours Setting: In school; U.S. Software/Tool: Unclear Methods: quantitative	N: 190 Age/Grade: Middle school Gender: Mixed sex Race/Ethnicity: White/AA/Hispanic Prior tech experience: Range						+			✓		✓
Minnigerode & Malerba	2012	Program Length: 1–2 school years; 75 minutes every day; unclear hours Setting: In school; U.S. Software/Tool: Flash actionscript Methods: quantitative	N: 189 Age/Grade: Middle school Gender: Mixed sex Race/Ethnicity: AA/Hispanic Prior tech experience: Range							+		✓		✓
Minnigerode & Reynolds	2013	Program Length: Unclear duration and hours Setting: In school; U.S. Software/Tool: Flash actionscript Methods: qualitative	N: 2 Age/Grade: Middle school Gender: All girls Race/Ethnicity: Unclear Prior tech experience: Unclear		+		+			+		✓		✓
Navarrete & Minnigerode	2013	Program Length: 1–3 school years; 75 minutes each day; unclear hours Setting: In school; U.S. Software/Tool: Flash actionscript Methods: qualitative	N: 12 Age/Grade: Middle school Gender: Unclear Race/Ethnicity: AA/Hispanic/Other Prior tech experience: Unclear		+		+					✓		✓
Patton	2013	Program Length: 5 days; 25 total hours Setting: In school; U.S. Software/Tool: GameMaker Methods: qualitative	N: 70 Age/Grade: 8–13 years old Gender: Unclear Race/Ethnicity: Unclear Prior tech experience: Range	+				+/−				✓		
Repenning & Ioannidou	2008	Program Length: Varied duration and hours Setting: After school and Summer; U.S. Software/Tool: AgentSheets Methods: qualitative	N: Unclear Age/Grade: Elementary, Middle and High school Gender: Unclear Race/Ethnicity: Unclear Prior tech experience: Unclear		+		+						✓	

(Continued)

Table 2. Continued

				Outcomes								Pedagogy		
				Programming Knowledge		PS&D		CS Attitude		CS Confidence				
Author(s)	Year	Program & Study Details	Participant Details	CE	DE	CE	DE	CE	DE	CE	DE	DBT	SWI	SI
Repenning & Ioannidou	2008	Program Length: Varied duration and hours Setting: After school and Summer; U.S. Software/Tool: AgentSheets Methods: qualitative	N: Unclear Age/Grade: Elementary, Middle and High school Gender: Unclear Race/Ethnicity: Unclear Prior tech experience: Unclear		+		+						✓	
Repenning, Web, Koh, Nickerson, Miller, Brand,...Grover	2008	Program Length: 8 weeks; unclear hours Setting: In school; U.S. Software/Tool: AgentSheets Methods: qualitative	N: 33 Age/Grade: Middle school Gender: Unclear Race/Ethnicity: Unclear Prior tech experience: Unclear	+			+						✓	
Reynolds	2010a	Program Length: 3–9 months; unclear hours Setting: In school; U.S. Software/Tool: Flash actionscript Methods: mixed	N: 386 Age/Grade: Middle and High school Gender: Mixed sex Race/Ethnicity: Unclear Prior tech experience: Unclear	+	+							✓		✓
Reynolds	2010b	Program Length: School year; 82 mins. daily sessions; unclear hours Setting: In school; U.S. Software/Tool: Flash actionscript Methods: qualitative	N: 2 Age/Grade: Middle school Gender: Mixed sex Race/Ethnicity: Unclear Prior tech experience: Yes	+	+							✓		✓
Reynolds & Chiu	2012	Program Length: 1 year; unclear hours Setting: In school; U.S. Software/Tool: Flash actionscript Methods: qualitative	N: 277 Age/Grade: Middle and High school Gender: Unclear Race/Ethnicity: Unclear Prior tech experience: Unclear		+							✓		✓
Reynolds & Chiu	2013	Program Length: 1 year; unclear hours Setting: In school and After school; U.S. Software/Tool: Flash actionscript Methods: quantitative	N: 93 Age/Grade: Middle school Gender: Mixed sex Race/Ethnicity: White/AA/Hispanic/Asian Prior tech experience: Range								–	✓		✓

(Continued)

Table 2. Continued

				Outcomes								Pedagogy		
				Programming Knowledge		PS&D		CS Attitude		CS Confidence				
Author(s)	Year	Program & Study Details	Participant Details	CE	DE	CE	DE	CE	DE	CE	DE	DBT	SWI	SI
Robertson	2013	Program Length: 9 weeks; 18 total hours Setting: In school; U.K. Software/Tool: Adventure Author Methods: quantitative	N: 225 Age/Grade: Elementary, Middle, High school Gender: Mixed sex Race/Ethnicity: Unclear Prior tech experience: Unclear					–				✓		✓
Ruggiero, de Hurtado, & Watson	2013	Program Length: 2 weeks; 10 total hours Setting: Camp; U.S. Software/Tool: Scratch Methods: qualitative	N: 10 Age/Grade: 13–19 years old (7th-11th grade) Gender: Mixed sex Race/Ethnicity: White/AA Prior tech experience: Unclear						–				✓	✓
Sanford & Madill	2007	Program Length: 2 months; unclear hours Setting: In school; Canada Software/Tool: Kid's Programming Language (KPL), GameMaker Methods: qualitative	N: 6 Age/Grade: High school Gender: Mixed sex Race/Ethnicity: Unclear Prior tech experience: Range				+					U	U	U
Sheridan, Clark, & Peters	2009	Program Length: 150 instructional hours available over the course of multiple years; unclear hours Setting: Weekend and summer; U.S. Software/Tool: Maya, Virtools, Alice, GameMaker, MissionMaker Methods: quantitative	N: 200+ Age/Grade: Elementary, Middle, and High school Gender: Unclear Race/Ethnicity: AA Prior tech experience: Range			+								✓
Simmons, DiSalvo, & Guzdial	2012	Program Length: 8 weeks; 24–48 total hours Setting: Summer; U.S. Software/Tool: GreenFoot Methods: qualitative	N: 15 Age/Grade: High school Gender: All boys Race/Ethnicity: AA Prior tech experience: Range		+					–		✓	✓	✓

(Continued)

Table 2. Continued

				Outcomes								Pedagogy		
				Programming Knowledge		PS&D		CS Attitude		CS Confidence				
Author(s)	Year	Program & Study Details	Participant Details	CE	DE	CE	DE	CE	DE	CE	DE	DBT	SWI	SI
Tapia, El-Nasr, Yucel, Zupko, & Maldonado	2007	Program Length: 5 weeks; 20 total hours (2 weekend groups); 1 week, unclear hours (summer group) Setting: Weekend and Summer; U.S. Software/Tool: Warcraft III, GameMaker, RPG Maker XP Methods: quantitative	N: 60–78 Age/Grade: Middle and High school Gender: All girls Race/Ethnicity: Unclear Prior tech experience: Had some computer skills					+					✓	
Thomas, Ge, & Greene	2011	Program Length: 2 weeks; unclear hours Setting: In school; U.S. Software/Tool: Quest Atlantis Methods: qualitative	N: 12 Age/Grade: High school Gender: Mixed sex Race/Ethnicity: Unclear Prior tech experience: Yes		+		+						✓	✓
Vartel & Riconscente	2012	Program Length: 10 weeks; 4 sessions/week; unclear hours Setting: After school; U.S. Software/Tool: MathMaker Methods: quantitative	N: 10 Age/Grade: High school Gender: Unclear Race/Ethnicity: AA/Hispanic Prior tech experience: Unclear						+			✓		
Wan, Eow, Mahmud, & Baki	2011	Program Length: 4 weeks; 16 total hours Setting: Unclear; Malaysia Software/Tool: GameMaker Methods: quantitative	N: 69 (34 tx; 34 control) Age/Grade: 13–14 years old Gender: Unclear Race/Ethnicity: Unclear Prior tech experience: Yes							+			✓	✓
Wang & Chen	2010	Program Length: 6 weeks; unclear hours Setting: Unclear; Unclear-Authors from Taiwan Software/Tool: Scratch, Flash Methods: mixed	N: 119 Age/Grade: Middle school Gender: Unclear Race/Ethnicity: Unclear Prior tech experience: Unclear		+								✓	

(Continued)

Table 2. Continued

				Outcomes								Pedagogy		
				Programming Knowledge		PS&D		CS Attitude		CS Confidence				
Author(s)	Year	Program & Study Details	Participant Details	CE	DE	CE	DE	CE	DE	CE	DE	DBT	SWI	SI
Wang & Chen	2011	Program Length: 4 weeks; unclear hours Setting: Unclear; Unclear-Authors from Taiwan Software/Tool: Scratch Methods: qualitative	N: 242 Age/Grade: 8th grade Gender: Unclear Race/Ethnicity: Unclear Prior tech experience: No		+							✓		
Wang & Chen	2012	Program Length: 12 weeks; unclear hours Setting: Unclear; Unclear-Authors from Taiwan Software/Tool: Scratch Methods: qualitative	N: 189 Age/Grade: 7th grade Gender: Unclear Race/Ethnicity: Unclear Prior tech experience: Unclear		+		+						✓	
Webb, Repenning, & Koh	2012	Program Length: Multi-week unit; unclear hours Setting: In school; U.S. Software/Tool: AgentSheets Methods: quantitative	N: 894 (year 1); 526 (year 2) Age/Grade: 6th-8th grade Gender: Mixed sex Race/Ethnicity: White/AA/Hispanic/Native American Prior tech experience: Unclear						+				✓	
Werner, Campe, & Denner	2012	Program Length: 1 semester; 20 hours Setting: In school and After school; U.S. Software/Tool: Storytelling Alice, Alice 2.2 Methods: qualitative	N: 325 Age/Grade: 10–14 years old (Middle school) Gender: Mixed sex Race/Ethnicity: White/Hispanic Prior tech experience: Unclear		+							✓	✓	✓
Werner, Denner, Bliesner, & Rex	2009	Program Length: 2 weeks; 20 total hours Setting: Community center; U.S. Software/Tool: Storytelling Alice Methods: qualitative	N: 22 Age/Grade: Middle school Gender: Mixed sex Race/Ethnicity: White/AA/Hispanic/Other Prior tech experience: Range		+								✓	✓

(Continued)

Table 2. Continued

				Outcomes								Pedagogy		
				Programming Knowledge		PS&D		CS Attitude		CS Confidence				
Author(s)	Year	Program & Study Details	Participant Details	CE	DE	CE	DE	CE	DE	CE	DE	DBT	SWI	SI
Werner, Denner, Campe, & Kawamoto	2012	Program Length: 1 semester; 20 total hours Setting: In school and After school; U.S. Software/Tool: Storytelling Alice, Alice 2.2 Methods: quantitative	N: 311 Age/Grade: Middle school Gender: Mixed sex Race/Ethnicity: White/Hispanic Prior tech experience: Unclear		+		+					✓	✓	✓
Wilson, Connolly, Hainey, & Moffat	2011	Program Length: 8 weeks; 8 total hours Setting: In school; Scotland Software/Tool: Scratch Methods: quantitative	N: 46 Age/Grade: 7–9 years old Gender: Mixed sex Race/Ethnicity: Unclear Prior tech experience: Unclear	+									✓	
Wilson, Hainey, & Connolly	2012	Program Length: 8 weeks; 8 total hours Setting: In school; Scotland Software/Tool: Scratch Methods: qualitative	N: 60 Age/Grade: 8–11 years old Gender: Mixed sex Race/Ethnicity: Unclear Prior tech experience: Unclear		+								✓	
Wu, Ashcraft, DuBow, & Reynolds	2012	Program Length: Unclear duration and hours Setting: In school; U.S. Software/Tool: Flash Actionscript Methods: quantitative	N: 539 Age/Grade: Middle and High school Gender: Mixed sex Race/Ethnicity: Unclear Prior tech experience: Unclear					-		-		✓		✓
Yang & Chang	2013	Program Length: 19 weeks; 14.25 total hours Setting: In school; Unclear-Authors from Taiwan Software/Tool: Flash, RPG Maker Methods: qualitative	N: 67 Age/Grade: 13–14 years old (7th grade) Gender: Mixed sex Race/Ethnicity: Unclear Prior tech experience: Unclear			+							✓	✓

[PS&D]Problem Specification & Design; [CE]Change Evidence; [DE]Descriptive Evidence; [DBT]Design-Build-Test; [SWI]Stepwise Instruction; [SI]Social Interaction; [U]Unspecified; [+]evidence that CGD&P did benefit students; [−]evidence indicated a negative or neutral result; [+/−]evidence of both positive results and either negative or neutral results; [✓]has the pedagogy that is checked.

4.1 Computer Science Learning, Attitude, and Confidence

This section describes the results of our analysis that aimed to determine the extent to which there is evidence to support the widely held belief that CGD&P promotes CS learning, attitudes, and confidence. Our definition of computer science learning is based on recent efforts to define computational thinking. In 2006, Jeannette Wing wrote that "computer science is the study of computation—what can be computed and how to compute it" [90]. During the following decade, many worked on building an operational definition of computational thinking (CT), including a recent one published by International Society for Technology in Education (ISTE) and Computer Science Teachers Association (CSTA) from which we draw our analytic categories [1]. Their operational definition states that:

CT is a problem-solving process that includes (but is not limited to) the following characteristics:

(1) Formulating problems in a way that enables us to use a computer and other tools to help solve them
(2) Logically organizing and analyzing data
(3) Representing data through abstractions such as models and simulations
(4) Automating solutions through algorithmic thinking (a series of ordered steps)
(5) Identifying, analyzing, and implementing possible solutions with the goal of achieving the most efficient and effective combination of steps and resources
(6) Generalizing and transferring this problem-solving process to a wide variety of problems

Based on this definition and the results of our analysis, we identified two important subsets of computational thinking—programming knowledge and problem specification and design (PS&D)—as two of the key analytic categories for the meta-synthesis. More than half of the articles described studies in which evidence was gathered for only the programming part of the ISTE/CSTA's operational definition of problem solving (step 4 above). More specifically, this category refers to knowledge critical for developing a solution in a way that computers can use to solve a problem. These articles, at a minimum, investigated aspects of the knowledge of a programming language's constructs but also many articles included investigations of aspects of the algorithmic thinking necessary to arrange the correct constructs into a series of ordered steps to create programs that implemented games. The PS&D category refers to articles that described studies that in almost all cases referred to their investigation of students' problem-solving abilities, even though, at most, only the first four steps of the ISTE/CSTA operational definition were included. None of the articles included optimization or efficiency aspects found in step 5 or generalization or transference aspects found in step 6. A third analytic category, CS confidence, can also be found in the ISTE and CSTA operational definition, which lists "confidence in dealing with complexity" as one of the dispositions and attitudes that are an essential part of computational thinking. The fourth analytic category we use to synthesize the research that emerged from our analysis is CS attitudes, which includes an interest in and valuing of CS.

We analyzed 68 studies to understand the role of CGD&P in CS-related learning, CS attitudes, and confidence with computers and CS. During the analysis process, the team identified whether studies provided change and/or descriptive evidence related to the four categories. As shown in Table 3, the most common category measured in the studies was programming knowledge, followed by CS attitude. The evidence for most studies of programming knowledge is descriptive, while half of the PS&D studies measured change only. There was overlap between the categories: nine studies measured both programming knowledge and PS&D; only one study measured change (vs. descriptive) in both programming knowledge and PS&D [23]. Studies that measured CS attitudes or CS confidence were more likely to measure change than to provide descriptions of those constructs.

Table 3. Number of Studies with Descriptive and Change Evidence

Academic Benefits	Total Number of Studies	Descriptive Evidence Only	Change Evidence Only	Descriptive + Change Evidence
Programming Knowledge	45	29	14	2
Problem Specification & Design	16	8	8	0
CS Attitude	22	7	15	0
CS Confidence	12	1	11	0

The analysis also revealed three common pedagogical approaches to engaging students in CS learning. The pedagogical approaches emerged from reading the articles; categories were initially created by the second author, and then verified by the first and third authors. Not all studies described their pedagogy, so it was not possible to get an accurate count of the studies that used each approach, or to identify the length of time that students spent making their games.

The first pedagogical approach focuses on teaching students about game design and taking them through a design-build-test cycle that includes a combination of the steps of planning, programming, testing (playing), revising, finalizing, and sharing their games where revising repeats some or all of the previous steps. This approach was used in 30 studies. For example, the Globaloria program used an iterative design process of Play, Plan, Prototype, Program, and Publish [62]. Their classes typically allowed students to drive some of the content or type of game they made. The second pedagogical approach, used in 40 studies, consists of teachers using stepwise instruction to structure and scaffold learning. In these studies, students learned to use the tool, along with CS concepts and/or skills, and sometimes modified or fixed existing games before designing their own game. Tutorials and/or programming challenges were often used to take students through different incremental programming steps before creating their own games, or students were led through a process of making simple to more complex games. For example, the Studio K curriculum includes completing design challenge missions to learn programming elements, then fixing a broken game before creating one's own game [6]. Thirty-five studies also incorporate a third strategy—social interaction, where students work in teams, in pairs, and/or get feedback on their game or game-making process; one of these studies indicated only using this strategy. Of all 68 studies, two did not specify any pedagogical approach.

In Table 4 below, we elaborate on the findings in Table 3, including details about whether the evidence is based on measures of change or is descriptive, and how CGD&P was taught. Some studies use more than one pedagogical approach and address multiple outcomes with different types of evidence.

The definitions of programming knowledge were similar enough across studies to allow for an integrative synthesis; we found this to be the case for PS&D as well. However, we found that the measurement strategies varied between studies and within analytic categories. As stated earlier, the focus is on the integration of findings across studies, and space limits prevent the inclusion of detail about every study; instead, that can be found in Table 2, including the tool used and the age of participants. Where relevant comparisons and contrasts can be made, more information about the studies is included.

4.2 Programming Knowledge

This section describes the results of our analysis of 45 studies of programming knowledge which, at the most basic level, is the understanding and use of a computer programming language's

Table 4. Number of Studies for Each Outcome by Pedagogy and Type of Evidence

Outcome	Pedagogy	Total	+ Change	+ Description	− Change	− Description
Programming Knowledge (n=46)	Design-Build-Test	22	9	15	0	0
	Stepwise Instruction	26	7	19	0	1
	Social Interaction	23	8	17	0	0
	Unspecified	1	1	0	0	0
Problem Specification & Design (n=16)	Design-Build-Test	6	3	3	0	0
	Stepwise Instruction	10	4	6	0	0
	Social Interaction	9	4	5	0	0
	Unspecified	1	0	1	0	0
CS Attitude (n=22)	Design-Build-Test	9	6	1	3	0
	Stepwise Instruction	14	8	5	2	1
	Social Interaction	10	5	2	2	1
	Unspecified	0	0	0	0	0
CS Confidence (n=12)	Design-Build-Test	9	6	0	3	0
	Stepwise Instruction	4	2	1	2	0
	Social Interaction	8	4	1	3	0
	Unspecified	0	0	0	0	0

constructs (e.g., assignments, variables, if statements, loops). We add to this basic knowledge the ability to arrange these constructs into "series of ordered steps" or programs using a process of algorithmic thinking [1]. The most common way that programming knowledge was measured was by analyzing student games. Some studies used other strategies to gauge students' understanding of CS constructs, such as surveys, observations, and performance assessments. Below is a summary of findings about the benefits of computer game programming based on both evidence of change and descriptive evidence. Where findings are available, we show what they say about the benefits of CGD&P over other computer-based activities, as well as what the results say about which pedagogical approach(es) have been shown to increase or engage students' programming knowledge.

The analysis across studies found consistent evidence that teaching children to program games can result in increases in programming knowledge. As shown in Table 3, 16 studies found positive changes in programming knowledge (two also included descriptive data). Benefits include increases in code sophistication [6] and increases in programming test scores [23]. The data used to measure change included pre–post surveys, and the results were similar across a range of age groups and programming tools. For example, high school students' understanding of programming concepts increased after making a game using Greenfoot [5] and Scratch [30]. Increases were also found in grade 6 students' knowledge of basic computational terminology and GameMaker domain knowledge [34] and in 12- to 15 year old's knowledge of how to use the Kodu programming tool [4]. One study looked at the application of knowledge and found increases in elementary and middle school students' ability to connect concepts used in their game (made using GameMaker) to other video games [54]. Observations over time were another source of data; one study of elementary school children found that their programming skills increased as they made nutrition games using GameMaker [9].

Few studies described how the learning conditions or students' prior knowledge determined the extent to which they changed. One study, in which the Alice programming environment was used, found that students learned more when they worked with a partner, particularly when they had less prior experience making games [18]. Another study of 11–12 year old students

compared four conditions that varied the goal specificity and scaffolding type of instruction on how to make Scratch games and found higher programming comprehension scores among those with non-specific goals and structuring scaffolds [23]. And a study of 7–9 year old children making games with Scratch found that increases in programming test scores were greater among those with lower academic performance [88].

The 32 studies with descriptive evidence provided more detailed information about the kinds of programming knowledge that students engage in while learning to make games, as well as the challenges they faced. The cross-study analysis showed consistent evidence that students demonstrate an understanding of programming knowledge, regardless of whether the data source is their games [19, 24, 89], a performance assessment [22, 33], or observations [7]. For example, the analysis of game patterns programmed by high school students using ScriptEase and Aurora Toolset suggests that most students mastered functions, modules, and procedural constructs, but only one quarter mastered conditional/control programming constructs [11]. Similarly, a study of middle school students making games with Alice found that students demonstrated an understanding of programming code in their games, but limited ability to debug someone else's game using programming logic [86]. Others described the difficulties that elementary and middle school students had with some programming knowledge, such as identifying the sources of error when trying to debug and not understanding the program logic [45].

Only a few descriptive studies produced findings to help us understand the conditions under which a certain CGD&P pedagogy may be preferable. For example, students that benefited the most include higher socioeconomic groups [82], and those who had greater initial intrinsic motivation, were younger, and had an experienced teacher [61]. Only one study found that working in a team while learning to build games may distract some students from spending time on programming tasks [41]. A study compared different stepwise instruction approaches and found that the benefits were the same, regardless of whether students learned general programming or game programming first [81].

There was limited evidence to suggest that there are benefits of game programming over other types of computer-based activities. One found that middle school students making games with Scratch used more programming constructs like variables and loops than students making music videos with Scratch or storytelling projects with Alice [2]. However, several methodological issues limit the generalizability of these findings. For example, there was a much smaller number of students making music videos, and they used a different platform for storytelling. In addition, this article did not indicate what type of pedagogy was used. Another study [38] found that elementary school children who programmed games with Logo scored higher on a test of programming commands than those that created instructional software. That study by Yasmin Kafai was published in 1995 and still provides some of the strongest evidence for the benefits of CGD&P over other programming tasks for the development of children's programming knowledge.

By looking across studies, we find some evidence that each of the three pedagogical approaches show promise for engaging students in or increasing their programming knowledge. Studies that used stepwise instruction but incorporated at least one other pedagogical approach had similar positive results for programming knowledge [30, 47] and time spent on programming tasks [85]. Studies that incorporated all three pedagogical approaches showed evidence of programming knowledge [71, 84, 86] and that making a game with a partner results in a larger change in programming knowledge than working alone, particularly among less experienced students [18]. The results were similar for studies that did and did not include social interaction. No studies compared conditions with and without social interaction, making it difficult to conclude whether it is an essential part of developing programming knowledge or not. Similarly, no studies tried to compare design-build-test to stepwise instruction (many included both), so it is not possible to

conclude whether one is more effective than the other. Thus, this small body of evidence is not enough to make recommendations about which pedagogical approach to use when the goal is to increase or engage children in programming knowledge.

In summary, there is evidence that CGD&P can be used to engage and build students' programming knowledge regardless of the pedagogical approach used. There is more consistent and more plentiful evidence that CGD&P engages children in programming knowledge than there is evidence that it can increase that knowledge. There is some indication that the benefits are different across age, socioeconomic status, academic achievement, motivation, and teacher experience, but the conclusions are restricted because most studies focused on the middle school age range. In addition, the generalizability of these findings is limited by a lack of data on whether benefits extend beyond the class, as well as a lack of data that can be used to form conclusions about what kinds of game mechanics or game content result in different kinds of programming knowledge.

4.3 Problem Specification & Design

This section describes the results of our analysis across the 16 studies that measured PS&D skills and confidence. For our purpose, PS&D is stating the problem in a way that a computer can be used to help solve it, analyzing and organizing the data involved in the problem solution, designing models or simulations of possible solutions, and programming the game. PS&D can be considered as more than programming because the student is engaged in analysis of the problem to determine what part of it can be solved using a computer. In addition, the measurement of it often involves analysis of the process. The results are organized by studies that measured changes in PS&D, those that described engagement in PS&D, and then a summary of what the results say about the use of different CGD&P pedagogical approach(es) for promoting PS&D.

The cross-study analysis revealed consistent evidence that CGD&P can result in increased PS&D capacity, regardless of which aspects of PS&D were measured. Several types of data were used to measure change in PS&D, including surveys, performance assessments, interviews, and observations. For example, elementary school students reported increases in problem solving skills after making games with Scratch [44] and middle school girls who made games using Flash reported an increased willingness and perceived ability to problem solve on the computer when pair programming [35]. Observations [27, 70] and game analysis data [27] suggest that teens developed more systematic PS&D strategies as they worked on making their game across a range of tools. In a study of 11–12 year old students that compared four conditions of varying goal specificity and scaffolding, there were higher problem-solving skills among students with non-specific goals and problematizing scaffolds [23].

Only two studies compared those that did and did not make games, but the findings were consistent across the two different middle school samples using different tools. Among middle school students using Flash and RPG Maker, there were increases in critical thinking skills as measured by a pre–post survey, and the change was greater among those that made games compared to those that did not [92]. Among middle school students making games with Kodu, there were greater increases in three types of PS&D skills (e.g., systems analysis and design, decision making, and trouble-shooting) as measured by a performance assessment compared to students who did not make games [3].

The synthesis across studies also suggests that there is descriptive evidence that students engage in PS&D as they learn CGD&P using a variety of tools, though the majority of results focus on middle school students. For example, an analysis of middle school students' Scratch games suggested that some engaged in analytic and quantitative reasoning that included modeling a real-world problem using math expressions and symbols [41]. In another study, middle school students showed evidence of managing problems in faulty situations as measured by a trouble shooting

assessment; this finding was supported by observations that were done when they were making games with AgentCubes [33]. Several other studies used observation data to conclude that students engaged in PS&D in order to make their game using a range of tools [59, 67], but the methods were not clearly described. Researchers also reported rich descriptions of student engagement in PS&D when middle school students worked with peers to create Flash games that address and teach social issues [50, 51]. Another study used a post-only performance assessment and concluded that middle school students showed limited ability to engage in complex PS&D that involved debugging someone else's code using conditional logic [86].

Several studies suggest that peer feedback may engage students in PS&D. Data included observations of interactions between high school and elementary school students [78], and pre–post surveys that showed greater increases in PS&D among students that got feedback from a peer on their Kodu game compared to those who did not [32]. In addition, observations and analysis of video recordings in one study found that the PS&D process of high school students making games with Quest Atlantis was positively influenced by feedback from others [78]. Two of the 16 studies measuring PS&D used all three pedagogical approaches and the results varied. One showed an increase in engagement of more complex and systematic PS&D [27], while another found that only a few students demonstrated PS&D [86].

Although the evidence of PS&D is promising, the limited number of studies as well as the use of more than one pedagogical approach means no conclusions can be made about whether any particular pedagogical approach by itself will benefit children's PS&D skills. Individual studies provide minimal insight into the role of pedagogy. For example, certain kinds of stepwise instruction, such as when teachers give more open-ended goals and "what if" questions rather than specific goals or structured scaffolding, appear to be more beneficial for promoting PS&D skills [23].

In summary, our analysis shows that there is evidence that CGD&P can increase PS&D skills and engage children in PS&D activities that include stating the problem in a way that a computer can be used to help solve it, analyzing and organizing the data involved in the problem solution, designing models or simulations of possible solutions, and programming the game. The findings were positive regardless of the tool or pedagogy used. The most common approach in the studies was stepwise instruction, and it showed equal promise when it was implemented alone or with other approaches. Several studies suggest that getting feedback on their game from other students will engage students in PS&D, but these results are primarily descriptive so there is no way to determine whether or not that feedback played a role in students' learning above and beyond making the games. Additional research is needed to compare different pedagogical approaches in order to understand which are more likely to promote PS&D, and for whom. However, the great variation in how PS&D was conceptualized and measured is one of the limitations of the results. Some studies described specific practices they measured, while others asked students to tell them if they feel more confident to solve computational problems. We need more clarity and consistency in the definitions of PS&D in order to make conclusions about which types benefit the most from CGD&P.

4.4 CS Attitude

This section describes the results of our analysis across the 22 studies that looked at whether CGD&P is related to how much students value computing, as well as their interest in doing more programming, taking more computing classes, or pursuing a computing career. The most common measures were observations, interviews, and surveys.

The results of the analysis suggest that the extent to which CGD&P led to changes in attitudes toward computing varied. In 10 studies, there was evidence of positive change in things like the value students placed on computer programming [4] and in their belief in the value of learning

about technology [50]. These studies were primarily with middle school students, and they used a range of programming tools and pedagogies. One of these studies found that the increase in positive attitudes toward computing among middle school students making games with Alice was greatest for students who worked with a partner and students who initially had more computer "creating" experience than their partners [18]. Three studies found mixed results. For example, less than half of the middle and high school students using Jypeli or Agent Sheets reported a change in attitude or wanted to continue programming after the class [43, 47]. Similarly, in one study, after creating games with GameMaker, middle and high school students reported seeing the value of computers and technology, but were not interested in making more computer games [22]. One study found that after elementary through high school students made games with Adventure Author, they were less likely to like computing or want to find out more about computing [63], while another used descriptive data to describe how middle and high school students lost interest in using Scratch when they went from playing around to trying to add rules for games [64].

Results about whether CGD&P can increase programming interest varied depending on the type of evidence and type of attitude measured. Fifteen studies reported a mix of positive and negative changes in programming interest, while seven studies reported descriptive evidence that was all positive. For example, in some studies, there were increases in students' interest to do more CGD&P outside the class [8, 54], take a more advanced class [68], or study or work in a computing-related field [5, 30, 43, 75]. But there were decreases in interest in taking more classes [54] and in doing programming, software development, and game design [26, 63], particularly among middle school boys [91]. One international study found that while U.S. students became more interested in becoming a computer programmer, this was not true for the UK or NZ students [25]. Descriptive evidence was more consistently positive [7, 45, 83], and boys were more interested when there was a higher boy to girl ratio in their class [83].

Overall, the evidence suggests that CGD&P can have a positive impact on attitudes toward computing, but the benefit varies across students and type of attitude. For example, the benefits for interest in taking more computing classes or pursuing a computing career varied—one-third of the findings were mixed or negative. Studies that measured changes in the extent to which students value computers and technology were more likely to be positive. The findings did not appear to vary across the type of pedagogy used. Additional studies are needed to determine for whom and under what conditions CGD&P can increase students' interest and valuing of computer science-related activities.

4.5 CS Confidence

The analysis of 12 studies suggests that there is also mixed evidence that CGD&P has benefits for students' confidence with programming or PS&D on the computer. Data were collected using surveys and interviews. Six studies found increases in reported confidence or identity as someone who can program computers and/or games [4, 5, 35, 44]. For example, middle school girls reported increases in their computer self-efficacy, particularly around programming and game building, as well as in their confidence to design a game [68]. Similarly, there were increases in middle and high school students' perceived ability to solve problems on the computer and in their confidence to pursue computer education or work, but a decrease in their perceived ability to program [48]. One of the studies found that gains in self-efficacy to persist in the face of challenges were greater for those who spent more time in class and for girls [49]. Three studies reported only negative findings. Results included no change in self-efficacy for doing programming, using software, or online research [62], and a decrease in confidence to learn game design or programming, particularly among high school students [91]. In one study, where programming confidence increased for some students but overall there was a shift toward a more neutral perception, the authors conclude

that many students developed a more realistic sense of their programming ability, which is also a benefit of CGD&P [94].

The analysis suggests that CGD&P can build confidence to program games, figure things out, get help from others, and make things on the computer. Most studies used the design-build-test cycle approach, alone or in combination with other approaches, so no conclusions can be made about the most promising pedagogy for building CS confidence. One of the rare studies that compared two pedagogical approaches to CGD&P found that students' intrinsic motivation to seek out challenges was higher when the class involved more structured stepwise instruction (some of which included group work) than when students made games at their own pace [21]. The studies did not provide enough information to form conclusions about why there is variation in who gains and who does not, and what supports need to be in place for different kinds of learners to build confidence.

4.6 Summary of CS Learning, Attitude, and Confidence

In summary, the analysis suggests that there is some evidence that CGD&P can be beneficial for CS learning, attitudes, and confidence, particularly for middle school students, and regardless of program tool or pedagogical approach. However, additional research is needed to make strong conclusions. The findings were overwhelmingly positive, despite the variation in how constructs such as programming knowledge, PS&D, and CS attitude were measured, and the variation in age groups and software/tools. However, the decrease or lack of change in several measures of CS attitude require further investigation.

The data did not allow us to make conclusions about which pedagogical approach holds the most promise for CS learning; studies that compare these approaches are needed to determine whether there are differences in their impact, and for whom. Stepwise instruction was most often used in studies, and the results were the same regardless of whether or not it was used with other pedagogical approaches. There were more studies of CS interest and attitude than of CS confidence, but the findings for confidence were more consistently positive.

4.7 Overall Summary

The analysis of studies suggests that there is only modest evidence to support widely held beliefs about the benefits of CGD&P. The strongest evidence is that CGD&P can engage students in PS&D, increase programming knowledge, and improve attitudes toward computer science. These findings suggest that CGD&P is a promising strategy for learning and engagement, but the amount and type of the evidence varied across the four outcomes and, in some cases, by student characteristics. The largest number of studies focused on programming knowledge, but the evidence was primarily descriptive, so the strongest evidence (based on 32 studies) is that CGD&P engages students in aspects of programming, but does not necessarily increase their programming knowledge or skills. There were 16 studies that reported changes in programming knowledge (sample sizes ranged from 2–386), and 15 that reported changes in CS attitude (sample sizes ranged from 10–462).

5 DISCUSSION

This article describes the results of a meta-synthesis, a critical approach to determining the extent to which assumptions about the benefits of CGD&P are supported by evidence. This is the first summary of CGD&P research where the results are based on a systematic review and integration of findings across studies. The focus was on whether there is evidence for widely held beliefs about the benefits for CS learning and attitudes. The conclusions describe the extent of evidence for particular outcomes across studies, as well as the types of evidence presented, whether particular groups of students benefited more, and the extent to which the results were different depending on the pedagogy used.

The results of the synthesis challenge widely held beliefs that overgeneralize the benefits of game design and programming for learning and motivation. The evidence is primarily descriptive for benefits in CS learning, and primarily based on measures of change for CS attitudes and CS confidence. While there is some evidence that CGD&P can produce changes in specific learning outcomes, there is more evidence that students engage in computing-related practices. Others have cited the importance of understanding the practices students use when making games [55], and this synthesis suggests that students do use programming knowledge to make games, and this is true across different age groups, programming tools, and pedagogies. There is also more modest evidence that they engage in PS&D. The findings are mostly positive, however it is unclear whether this is due to a lack of reporting of neutral or negative findings; conflicting findings were most common for CS attitudes. This may be due to the low likelihood that descriptive studies will be looking for a lack of programming knowledge or an absence of PS&D.

The lack of studies that compare CGD&P to other conditions means that no conclusions can be drawn about the benefits of making games over other computer-based or programming activities. However, the few studies do suggest that at the middle and elementary school level, CGD&P has greater benefits for children's PS&D and programming knowledge than other computer-based activities.

The findings show the most common types of pedagogy used for teaching children to design and program games, and provide some indication that social interaction may have benefits for learning. Both the design-build-test and stepwise instruction pedagogical approaches yielded similar findings, with or without opportunities for social interaction. Several studies suggest that social interaction, particularly feedback from peers, can engage students in PS&D, although the findings are primarily descriptive. Only one study of programming knowledge compared students working with a partner and alone, and it found that pair programming can lead to increases in knowledge, particularly among those with less programming experience.

A meta-synthesis approach allowed for the integration across different studies with different methodological approaches. We believe that different kinds of measures can be considered a source of strong evidence, but it depends on how they are carried out (e.g., an unvalidated pre–post survey with 10 students provides weaker evidence than a validated survey with 400 students). However, the lack of additional detail about the methods and procedures made it difficult for us to critique each individual study. Instead, as we state in the methods section, as long as a study met our basic criteria for relevance and detail, it was given equal weight in the synthesis. There was only one notable difference in the results produced by qualitative and quantitative data. Although most studies reported positive results, when negative results were found, they were more likely to happen when quantitative measures were used.

The analysis phase revealed several limitations to our efforts to synthesize across studies. Given the overwhelmingly positive results across studies, there was little room to compare and contrast the findings, which left us with little insight into who benefits, and under what conditions. Instead, the analysis resulted in three conceptual approaches for thinking about the extent to which there is evidence of the benefits of CGD&P. First, we identified two main categories of CS learning that are measured in studies of CGD&P: programming knowledge and PS&D. Second, we provided a framework that educators and researchers can use to interpret the results, including looking at whether the evidence is descriptive or includes a measure of change. Third, we identified the two most common pedagogical approaches and showed that there was great variation in the extent to which they also incorporated social interaction.

The inclusion criteria limit the generalizability of the findings. For example, we focused on studies where CGD&P was the primary activity and did not include studies where the effects of CGD&P could not be separated from other activities [16, 19, 42]. In addition, the synthesis included

a limited number of studies of children outside the U.S., due in part to our lack of resources to read articles not written in the English language. In addition, studies where children modified games but did not program a change in interaction were not included; examples of these are reviewed in Gee and Tran [28].

6 SUGGESTED DESIGN ELEMENTS FOR CGD&P RESEARCH

The results of the synthesis have led us to identify key design elements for research on CGD&P, due to the great variation in how research on CGD&P is carried out and described. Overall, greater consistency in methods and reporting is needed in order to compare and contrast the findings from each individual study, and to form strong conclusions about the benefits of CGD&P. Specifically, there were inconsistencies in whether and how details were included about the type of program, the pedagogy, and the participants. For example, 34 studies did not fully report the total number of program hours, 11 did not describe the setting, and many lacked detail about participants' gender (18), race/ethnicity (43), and prior technology experience (39). In addition, reports about age made cross-study comparisons difficult because they were often missing, or varied from citing the ages in years, to describing a grade level in school, or a type of school (e.g., elementary school). Previous reviews of game making for learning have also found that a large number of studies did not report participants' demographics and background, and the location of the classes [20, 39]. These details about study participants are needed in order to inform conclusions about whether the benefits are stronger for certain kinds of students, and how to support different types of learners.

In order to evaluate the extent to which study claims are warranted, and to inform additional studies of CGD&P, there needs to be an adequate description of the research design and methods that were employed. For example, in several studies it was not clear whether changes in learning were measured, and if so, for what proportion of the participants. In many of the studies that used a comparison group, the similarities and differences at baseline were not always described. More detail about the pedagogy used to teach CGD&P is needed. It was not always possible to discern the amount of time that children spent designing or programming games, or the instructional approach(es) that were used.

To this end, we developed a list of criteria that can be used to guide research studies of CGD&P activities for children and youth. The list is consistent with other calls for more detailed information about methods and participants that allow readers to verify the authors' claims [15]. The criteria can also be used by educators to evaluate research studies, including whether the results of a given study should be to inform their decisions about what CGD&P approach to use, given their goals and population. The most trustworthy studies describe:

- Clear objectives or expected benefits, including definitions of outcomes and how they build on prior studies
- Details about the study (e.g., the participants, the setting, the pedagogy or curriculum, the teacher, the length and duration of program and activities, and the programming language)
- The research design (e.g., whether evidence is based on measures of change or engagement)
- Measures and data collection methods used (e.g., psychometric properties of instruments, and details about how observations are done)
- Detailed results, including information about who benefits and who does not, and any negative or neutral results

This list can be used as criteria to help educators, researchers, and policymakers evaluate the existence and type of research evidence about the benefits of CGD&P and to inform decisions—by educators, parents, and program developers—about how to design computer game programming activities for specific populations of children and teens. We also need more studies that compare the

benefits of programming games to other types of computer programming, as well as to traditional instruction. In addition, studies are needed that compare different approaches to teaching CGD&P, and explore whether the benefits vary depending on the programming tool or the type of game that was made. There is a need for more studies that include students from multiple geographical regions to increase the generalizability of the results. And we need more consistency in measures in order to look across studies, more studies that report negative or neutral findings in addition to their positive findings, and more studies that collect evidence of delayed benefits beyond the last day of class.

7 CONCLUSION

In an era of increasing emphasis on games, learning, and creating, researchers, practitioners, funders, and policy makers need research evidence to guide their decisions. This article describes a systematic and critical review of the existing research on children designing and programming computer games, and it identifies the most common pedagogical strategies being used to teach CGD&P across a range of goals. The results suggest that children engage in programming knowledge and PS&D while making computer games, and some increase their skills in these areas. The findings also suggest that CGD&P can result in more positive attitudes toward and confidence with CS, although the results were mixed. In addition, the results highlight the gaps in research that must be filled to inform the next generation of programming languages and environments, activities, and research studies in CGD&P.

REFERENCES

[1] 2011. An Operational Definition of Computational Thinking. Retrieved from http://www.iste.org/docs/ct-documents/computational-thinking-operational-definition-flyer.pdf?sfvrsn=2.

[2] Joel C. Adams and Andrew R. Webster. 2012. What do students learn about programming from game, music video, and storytelling projects? In *Proceedings of the 43rd ACM Technical Symposium on Computer Science Education (SIGCSE'12)*. ACM, New York, NY, 643–648. DOI: https://doi.org/10.1145/2157136.2157319

[3] Mete Akcaoglu. 2013. *Cognitive and Motivational Impacts of Learning Game Design on Middle School Children*. Michigan State University.

[4] Mete Akcaoglu. 2014. Learning problem-solving through making games at the game design and learning summer program. *Educational Technology Research and Development* 62, 5 (2014), 583–600.

[5] Mohammed Al-Bow, Debra Austin, Jeffrey Edgington, Rafael Fajardo, Joshua Fishburn, Carlos Lara, Scott Leutenegger, and Susan Meyer. 2009. Using game creation for teaching computer programming to high school students and teachers. *ACM SIGCSE Bulletin* 41, 3 (2009), 104–108.

[6] Gabriella Anton, Shannon Harris, Amanda Ochsner, and Kurt Squire. 2013. Patterns of play: Understanding computational thinking through game design. In *Proceedings of the Games, Learning, and Society Conference*, Vol. 3.

[7] Ashok R Basawapatna, Kyu Han Koh, and Alexander Repenning. 2010. Using scalable game design to teach computer science from middle school to graduate school. In *Proceedings of the 15th Annual Conference on Innovation and Technology in Computer Science Education*. ACM, 224–228.

[8] Ahmet Baytak. 2009. An investigation of the artifacts, outcomes, and processes of constructing computer games about environmental science in a fifth grade science classroom. Dissertation, Pennsylvania State University.

[9] Ahmet Baytak, Susan M. Land, Brian K. Smith, and S. Park. 2008. An exploratory study of kids as educational game designers. In *Proceedings of the 31st Annual Convention of the Association for Educational Communications and Technology. Orlando, FL*. 6–9.

[10] Quinn Burke and Yasmin B. Kafai. 2014. Decade of game making for learning: From tools to communities. *Handbook of Digital Games* (2014), 689–709.

[11] Mike Carbonaro, Duane Szafron, Maria Cutumisu, and Jonathan Schaeffer. 2010. Computer-game construction: A gender-neutral attractor to computing science. *Computers & Education* 55, 3 (2010), 1098–1111.

[12] Lisa Castaneda and Manrita Sidhu. 2015. Beyond Programming: The Power of Making Games. Retrieved from https://thejournal.com/articles/2015/02/18/beyond-programming-the-power-of-making-games.aspx.

[13] Vicki S. Conn, Sang-arun Isaramalai, Sabyasachi Rath, Peeranuch Jantarakupt, Rohini Wadhawan, and Yashodhara Dash. 2003. Beyond MEDLINE for literature searches. *Journal of Nursing Scholarship* 35, 2 (2003), 177–182.

[14] Harris Cooper. 2017. Research synthesis and meta-analysis: A step-by-step approach. Sage Publications, Los Angeles.
[15] Phillip Dawson, Jacques van der Meer, Jane Skalicky, and Kym Cowley. 2014. On the effectiveness of supplemental instruction: A systematic review of supplemental instruction and peer-assisted study sessions literature between 2001 and 2010. *Review of Educational Research* 84, 4 (2014), 609–639.
[16] Jill Denner, Steve Bean, and Jacob Martinez. 2009. The girl game company: Engaging Latina girls in information technology. *Afterschool Matters* 8 (2009), 26–35.
[17] Jill Denner and Linda Werner. 2007. Computer programming in middle school: How pairs respond to challenges. *Journal of Educational Computing Research* 37, 2 (2007), 131–150.
[18] Jill Denner, Linda Werner, Shannon Campe, and Eloy Ortiz. 2014. Pair programming: Under what conditions is it advantageous for middle school students? *Journal of Research on Technology in Education* 46, 3 (2014), 277–296.
[19] Jill Denner, Linda Werner, and Eloy Ortiz. 2012. Computer games created by middle school girls: Can they be used to measure understanding of computer science concepts? *Computers & Education* 58, 1 (2012), 240–249.
[20] Jeffrey Earp. 2015. Game making for learning: A systematic review of the research literature. In *Proceedings of 8th International Conference of Education, Research and Innovation (ICERI2015)*. 6426–6435.
[21] Yee Leng Eow, Wan Ali Wan Zah, Mahmud Rosnaini, and Baki Roselan. 2011. Appreciative learning approach as a pedagogical strategy and computer game development as a technological tool in enhancing students' creativity. *Journal of the Research Center for Educational Technology* 7, 2 (2011), 58–85.
[22] Jeremy V. Ernst and Aaron C. Clark. 2012. Fundamental computer science conceptual understandings for high school students using original computer game design. *Journal of STEM Education* 13, 5 (2012), 40–45.
[23] Chia-Yen Feng and Ming-Puu Chen. 2014. The effects of goal specificity and scaffolding on programming performance and self-regulation in game design. *British Journal of Educational Technology* 45, 2 (2014), 285–302.
[24] W. Foster, M. S. El-Nasr, and T. Maygoli. 2011. IT fluency through game development: A study of the utility of game workshops as an after school activity for middle and high schools. (2011).
[25] Allan Fowler. 2012. Enriching student learning programming through using Kodu. In *Proceedings of the 3rd Annual Conference of Computing and Information Technology, Education and Research in New Zealand (incorporating 24th Annual NACCQ)*.
[26] Allan Fowler and Brian Cusack. 2011. Kodu game lab: Improving the motivation for learning programming concepts. In *Proceedings of the 6th International Conference on Foundations of Digital Games*. ACM, 238–240.
[27] Alex Games and Luke Kane. 2011. Exploring adolescent's STEM learning through scaffolded game design. In *Proceedings of the 6th International Conference on Foundations of Digital Games*. ACM, 1–8.
[28] Elisabeth R. Gee and Kelly M. Tran. 2016. Video game making and modding. In *Handbook of Research on the Societal Impact of Digital Media*. IGI Global, 238–267.
[29] Mario A. M. Guimaraes. 2011. Game development with GameMaker, Flash and Unity. In *Proceedings of the 49th Annual Southeast Regional Conference*. ACM, 9.
[30] Roxana Hadad. 2013. Using game design as a means to make computer science accessible to adolescents. In *Cases on Digital Game-Based Learning: Methods, Models, and Strategies*. IGI Global, 279–300.
[31] Elisabeth R. Hayes and Ivan Alex Games. 2008. Making computer games and design thinking: A review of current software and strategies. *Games and Culture* 3, 3–4 (2008), 309–332.
[32] Gwo-Jen Hwang, Chun-Ming Hung, and Nian-Shing Chen. 2014. Improving learning achievements, motivations and problem-solving skills through a peer assessment-based game development approach. *Educational Technology Research and Development* 62, 2 (2014), 129–145.
[33] Andri Ioannidou, Alexander Repenning, and David C. Webb. 2009. AgentCubes: Incremental 3D end-user development. *Journal of Visual Languages & Computing* 20, 4 (2009), 236–251.
[34] Jennifer Jenson and Milena Droumeva. 2015. Making games with Game Maker: A computational thinking curriculum case study. In *9th European Conference on Games Based Learning (ECGBL2015)*. Academic Conferences and publishing limited, 260.
[35] Denner Jill and Steve Bean. 2005. Learning as intrepid exploration: A new model for teaching children to work with computers. In *Proceedings of the International Conference on Computers and Advanced Technology for Education (IASTED 2005)*.
[36] Jesper Juul. 2011. *Half-real: Video Games between Real Rules and Fictional Worlds*. MIT press.
[37] Yasmin B. Kafai. 2006. Playing and making games for learning: Instructionist and constructionist perspectives for game studies. *Games and Culture* 1, 1 (2006), 36–40.
[38] Yasmin B. Kafai. 2012. *Minds in Play: Computer Game Design as a Context for Children's Learning*. Routledge.
[39] Yasmin B. Kafai and Quinn Burke. 2015. Constructionist gaming: Understanding the benefits of making games for learning. *Educational Psychologist* 50, 4 (2015), 313–334.
[40] Yasmin B. Kafai and Quinn Burke. 2016. *Connected Gaming: What Making Video Games Can Teach Us about Learning and Literacy*. MIT Press.

[41] Fengfeng Ke. 2014. An implementation of design-based learning through creating educational computer games: A case study on mathematics learning during design and computing. *Computers & Education* 73 (2014), 26–39.
[42] Melissa Koch. 2009. Build IT: Girls Building Information Technology Fluency through Design. (2009).
[43] Antti-Jussi Lakanen, Ville Isomöttönen, and Vesa Lappalainen. 2014. Five years of game programming outreach: Understanding student differences. In *Proceedings of the 45th ACM Technical Symposium on Computer Science Education*. ACM, 647–652.
[44] Qing Li. 2010. Digital game building: Learning in a participatory culture. *Educational Research* 52, 4 (2010), 427–443.
[45] Janet Mei-Chuen Lin, Long-Yuen Yen, Mei-Ching Yang, and Chiao-Fun Chen. 2005. Teaching computer programming in elementary schools: A pilot study. In *National Educational Computing Conference*. Citeseer.
[46] Claire Howell Major and Maggi Savin-Baden. 2011. Integration of qualitative evidence: Towards construction of academic knowledge in social science and professional fields. *Qualitative Research* 11, 6 (2011), 645–663.
[47] C. K. Martin, S. Walter, and B. Barron. 2009. Looking at learning through student designed computer games: A rubric approach with novice programming projects. *Unpublished article, Stanford University* (2009).
[48] Bruce R. Maxim, William I. Grosky, and John P. Baugh. 2007. Work in progress-introducing information technology through game design. In *ASEE/IEEE Frontiers in Education Conference*.
[49] Laura Minnigerode and Catherine Malerba. 2012. Engineering self-efficacy in economically disadvantaged and English language learner middle school students learning game design. Unpublished paper, Worldwideworkshop.com.
[50] Laura Minnigerode. 2013. Quantitative data report and analysis school year 2011-2012. (2013).
[51] Cesar C. Navarrete and Laura Minnigerode. 2013. Exploring 21st century learning: Game design and creation, the students' experience. In *Proceedings of the World Conference on Educational Multimedia, Hypermedia and Telecommunications*. 282–293.
[52] Seymour Papert and Idit Harel. 1991. Situating constructionism. In *Constructionism*, Idit Harel and Seymour Papert (Eds.). Ablex Publishing Company.
[53] Seymour Papert. 1993. *The Children's Machine: Rethinking School in the Age of the Computer*. ERIC.
[54] Ryan M. Patton. 2013. Games as an artistic medium: Investigating complexity thinking in game-based art pedagogy. *Studies in Art Education* 55, 1 (2013), 35–50.
[55] Caroline Pelletier, Andrew Burn, and David Buckingham. 2010. Game design as textual poaching: Media literacy, creativity and game-making. *E-learning and Digital Media* 7, 1 (2010), 90–107.
[56] William R. Penuel. 2016. Studying science and engineering learning in practice. *Cultural Studies of Science Education* 11, 1 (2016), 89–104.
[57] Catherine Pope, Nicholas Mays, and Jennie Popay. 2007. *Synthesising Qualitative and Quantitative Health Evidence: A Guide to Methods: A Guide to Methods*. McGraw-Hill Education (UK).
[58] Jill Denner, Erica Marsh, and Shannon Campe. 2016. Approaches to reviewing research in education. *The BERA/SAGE Handbook of Educational Research*, 143–164.
[59] Alexander Repenning and Andri Ioannidou. 2008. Broadening participation through scalable game design. In *ACM SIGCSE Bulletin*, Vol. 40. ACM, 305–309.
[60] Mitchel Resnick, John Maloney, Andres Monroy-Hernández, Natalie Rusk, Evelyn Eastmond, Karen Brennan, Amon Millner, Eric Rosenbaum, Jay Silver, Brian Silverman, et al. 2009. Scratch: Programming for all. *Communication of the ACM* 52, 11 (2009), 60–67.
[61] Rebecca Reynolds and Ming Ming Chiu. 2012. Contribution of motivational orientations to student outcomes in a discovery-based program of game design learning. (2012).
[62] Rebecca Reynolds and Ming Ming Chiu. 2013. Formal and informal context factors as contributors to student engagement in a guided discovery-based program of game design learning. *Learning, Media and Technology* 38, 4 (2013), 429–462.
[63] Judy Robertson. 2013. The influence of a game-making project on male and female learners' attitudes to computing. *Computer Science Education* 23, 1 (2013), 58–83.
[64] Dana Ruggiero, Belen Garcia de Hurtado, and William R. Watson. 2013. Juvenile offenders: Developing motivation, engagement, and meaning-making through video game creation. *International Journal of Game-Based Learning (IJGBL)* 3, 2 (2013), 112–129.
[65] Michael Saini and Aron Shlonsky. 2012. *Systematic Synthesis of Qualitative Research*. OUP USA.
[66] Margarete Sandelowski, Julie Barroso, and Corrine I. Voils. 2007. Using qualitative metasummary to synthesize qualitative and quantitative descriptive findings. *Research in Nursing & Health* 30, 1 (2007), 99–111.
[67] Kathy Sanford and Leanna Madill. 2007. Recognizing new literacies: Teachers and students negotiating the creation of video games in school. In *Proceedings of the DiGRA Conference*.
[68] Magy Seif El-Nasr, Ibrahim Yucel, Joseph Zupko, Andrea Tapia, and Brian Smith. 2007. Middle-to-high school girls as game designers—What are the implications?. In *Second Annual Microsoft Academic Days Conference on Game Development*.

[69] Richard Shavelson and Lisa Towne. 2002. Scientific research in education. Committee on scientific principles for education research. Center for Education. Division of Behavioral and Social Sciences and Education. National Research Council. National Academy Press, Washington, DC.
[70] Kimberly Sheridan, Kevin Clark, and Erin Peters. 2009. How scientific inquiry emerges from game design. In *Society for Information Technology & Teacher Education International Conference*. Association for the Advancement of Computing in Education (AACE), 1555–1563.
[71] Steven Simmons, Betsy DiSalvo, and Mark Guzdial. 2012. Using game development to reveal programming competency. In *Proceedings of the International Conference on the Foundations of Digital Games*. ACM, 89–96.
[72] James C. Spohrer and Elliot Soloway. 1989. Simulating student programmers. *International Joint Conferences on Artificial Intelligences*.
[73] Harsh Suri. 2013. *Towards Methodologically Inclusive Research Syntheses: Expanding Possibilities*. Routledge.
[74] Harsh Suri and David Clarke. 2009. Advancements in research synthesis methods: From a methodologically inclusive perspective. *Review of Educational Research* 79, 1 (2009), 395–430.
[75] Andrea H. Tapia, Magy Seif El-Nasr, Ibrahim Yucel, Joseph Zupko, and Edgard Maldonado. 2007. Building virtual spaces. In *Virtuality and Virtualization*. Springer, 317–334.
[76] Katie Salen Tekinbas, Robert Torres, Loretta Wolozin, Rebecca Rufo-Tepper, and Arana Shapiro. 2010. *Quest to Learn: Developing the School for Digital Kids*. MIT Press.
[77] James Thomas and Angela Harden. 2008. Methods for the thematic synthesis of qualitative research in systematic reviews. *BMC Medical Research Methodology* 8, 1 (2008), 45.
[78] Michael K. Thomas, Xun Ge, and Barbara A. Greene. 2011. Fostering 21st century skill development by engaging students in authentic game design projects in a high school computer programming class. *Journal of Educational Computing Research* 44, 4 (2011), 391–408.
[79] Ian Utting, Stephen Cooper, Michael Kölling, John Maloney, and Mitchel Resnick. 2010. Alice, Greenfoot, and Scratch–A discussion. *ACM Transactions on Computing Education (TOCE)* 10, 4 (2010), 17.
[80] Lucien Vattel and Michelle Riconscente. 2012. MathMaker: Teaching math through game design and development. In *Proceedings of the Games, Learning, Society*. 313–322.
[81] Li-Chun Wang and Ming-Puu Chen. 2010. Learning programming concepts through game design: A PCT perspective. In *Proceedings of the 2010 3rd IEEE International Conference on Digital Game and Intelligent Toy Enhanced Learning (DIGITEL)*. IEEE, 219–221.
[82] Li-Chun Wang and Ming-Puu Chen. 2011. The relationships of social economic status and learners. In *2011 11th IEEE International Conference on Advanced Learning Technologies*. IEEE, 524–528.
[83] David C. Webb, Alexander Repenning, and Kyu Han Koh. 2012. Toward an emergent theory of broadening participation in computer science education. In *Proceedings of the 43rd ACM Technical Symposium on Computer Science Education*. ACM, 173–178.
[84] Linda Werner, Shannon Campe, and Jill Denner. 2012. Children learning computer science concepts via Alice game-programming. In *Proceedings of the 43rd ACM Technical Symposium on Computer Science Education*. ACM, 427–432.
[85] Linda Werner, Jill Denner, Michelle Bliesner, and Pat Rex. 2009. Can middle-schoolers use Storytelling Alice to make games?: Results of a pilot study. In *Proceedings of the 4th International Conference on Foundations of Digital Games*. ACM, 207–214.
[86] Linda Werner, Jill Denner, Shannon Campe, and Damon Chizuru Kawamoto. 2012. The fairy performance assessment: Measuring computational thinking in middle school. In *Proceedings of the 43rd ACM Technical Symposium on Computer Science Education*. ACM, 215–220.
[87] Robin Whittemore and Kathleen Knafl. 2005. The integrative review: Updated methodology. *Journal of Advanced Nursing* 52, 5 (2005), 546–553.
[88] Amanda Wilson, Thomas Connolly, Thomas Hainey, and David Moffat. 2011. Evaluation of introducing programming to younger school children using a computer game making tool. In *Proceedings of the 5th European Conference on Games Based Learning*. 639–649.
[89] Amanda Wilson, Thomas Hainey, and Thomas Connolly. 2012. Evaluation of computer games developed by primary school children to gauge understanding of programming concepts. In *Proceedings of the 6th European Conference on Games-Based Learning (ECGBL)*. 4–5.
[90] Jeannette M. Wing. 2006. Computational thinking. *Commun. ACM* 49, 3 (2006), 33–35.
[91] Zhen Wu, Catherine Ashcraft, Wendy DuBow, et al. 2012. Assessing girls' interest, confidence, and participation in computing activities: Results for Globaloria in West Virginia. Unpublished paper from http://www.ncwit.org.
[92] Ya-Ting Carolyn Yang and Chao-Hsiang Chang. 2013. Empowering students through digital game authorship: Enhancing concentration, critical thinking, and academic achievement. *Computers & Education* 68 (2013), 334–344.

Received October 2017; revised July 2018; accepted August 2018