Jan 17th, 12:00 AM

Game Development Based Approach for Learning to Program: A Systematic Literature Review

Nikolaj Bewer
University of Potsdam, Germany, nbewer@uni-potsdam.de

Margarita Gladkaya
University of Potsdam, Germany, gladkaya@uni-potsdam.de

Game Development Based Approach for Learning to Program: A Systematic Literature Review

Nikolaj Bewer[1,3] and Margarita Gladkaya[2,3]

[1] codary, Berlin, Germany
{n.bewer}@codary.org
[2] Weizenbaum Institute for the Networked Society, Digital Technologies and Well-being, Berlin, Germany
[3] University of Potsdam, Chair of Social Media and Society, Potsdam, Germany
{gladkaya,nikolaj.bewer}@uni-potsdam.de

Abstract. Digitalization is advancing rapidly around the globe. Sufficient training in programming, computer science, and IT should be provided to ensure that upcoming generations do not just experience digitalization but rather actively participate in the process. One emergent educational approach aimed to lower educational barriers and increase the engagement of children and teenagers in programming is game development based learning (GDBL). In this study, we conduct a systematic literature review of empirical papers applying GDBL within programming and computer science courses. The attributes and learning outcomes of GDBL courses for various educational settings are reported and discussed. The results suggest that GDBL in computer science and programming education is an effective approach. We conclude with implications for future research and practice.

Keywords: GDBL, game development based learning, programming, review

1 Introduction

Digitalization is advancing at a rapid pace around the globe. It is evident from the unprecedentedly high number of smartphone users worldwide [1]. People's communication, information consumption and processing are steadily shifting into the digital realm [2]. Indeed, many processes that used to happen offline in an analog form are getting digitalized, with new jobs emerging in such areas. Not surprisingly, more and more vacancies with IT skills as a requirement open up [3]. For example, economists and business administrators must process, clean up and analyze the ever-increasing amounts of data. Marketers must deeply understand search engine optimization and be able to drive digital campaigns on social media. Graphic designers are often expected to have at least basic website software skills. Hence, elemental-level knowledge of the processes behind algorithms, websites, and apps is increasingly in demand, with such requirements going above and beyond purely technical professions.

In order to not only experience digitalization but to actively shape it, the new generation needs to receive adequate training in the areas of programming and computer science (CS) from an early age. However, the contemporary institutions of formal education risk falling short of the modern age expectations. Instilling the interest in programming

17th International Conference on Wirtschaftsinformatik
February 2022, Nürnberg, Germany

and computer science in young children and teenagers with no prior knowledge is challenging [4–6]. At the same time, university instructors struggle to maintain students' engagement in programming courses and battle low completion rates and low enrollment rates at Science, Technology, Engineering, and Math – STEM degree programs [7–9].

Against this backdrop, novel educational methods relying on games or incorporating elements of the game design emerge [10, 11]. The present paper focuses on game development based learning (GDBL). This game-based method uniquely mobilizes students' enthusiasm about video games for educational purposes [11] allowing for hands-on learning. In application to CS and programming, the method started to gain scholarly attention in the mid 2000s [10, 11]. The first synthesis of the GDBL research was published in 2012 [11]. Still, the scientific evidence on GDBL application and effects remains fragmented. We conduct a systematic literature review of GDBL use for teaching programming. The review incorporates novel findings from the last decade of research in a scientific discussion and generates new insights into the effects of GDBL. With our review, we set out to answer the following research questions: *(1) What are the attributes of GDBL-based programming courses? (2) What are the outcomes of learning a programming language through the GDBL educational approach?*

Twenty two empirical studies – describing the application and effects of teaching a programming language through GDBL – were included in the review and analyzed along the following categories: (1) educational setting; (2) the type of programming language; (3) the type of games to be developed and modified; and (4) learning outcomes. The results contribute to the academic research into GDBL as an educational method and offer valuable insights for practitioners. First, we identify specific attributes and differences between GDBL courses taught at the school and higher education levels. Thereby, our synthesis of the papers introduces interested teachers and university instructors to the appropriate tools for a given education level: programming languages, integrated development environments (IDEs), and other materials necessary for the GDBL application within their courses. Second, we provide a nuanced perspective on the reported effects of GDBL programming courses by accounting for the study design and data analysis methods. Third, we formulate methodological recommendations and point out avenues for future research.

2 Game-Related Educational Methods

Several game-based educational approaches can be identified in the academic literature. The most common terms are *game based learning* (GBL), *serious games*, *game development based learning* (GDBL), and *gamification*. Some of these approaches are related, others are distinct.

GBL refers to the educational procedure that imparts knowledge by allowing learners to play games [10]. In this approach, instructors convey knowledge subtly through the game, pursuing the goal of motivating participants who are hard to engage with conservative learning methods. The GBL method is sometimes referred to as "edutainment" [10], owing to the idea of making the learning process more entertaining. *Serious games* refer to a subset within the GBL methods. Here, the games designed for particular learning objectives are used [10, 12].

Compared to other approaches, *gamification* is relatively new [10] and refers to the application of game design philosophy and game elements to real-life environments [13]. Thereby, without playing actual games, a game-like experience gets created to increase motivation, engagement, and performance [13].

Within GDBL, the learning occurs in the process of game development. The game is not a full-fledged educational product; it is to be built or modified by learners [11]. Similar to other game-related approaches, GDBL introduces the fun factor to the educational context [14]. However, by enabling the necessary hands-on experience, GDBL stands out as it best reflects the practical nature of programming and computer science courses. For instance, this game-related method is often described in studies about software engineering education [10]. Against this background, the GDBL method has become the focus of the present study.

To the best of our knowledge, there are two other literature reviews on game-related learning methods [10, 11]. The study by Wu and Wang [11] was published in 2012 and, therefore, does not cover recent technological and methodological developments in the research domain. Souza et al. [10] have reviewed papers about GDBL and other game-related methods in application to a very specific subject (i.e., software engineering). In the face of rapid digitalization, this paper improves over the previous works by considering a broader scope of the GDBL applications for teaching programming and incorporating the most recent studies in the review.

3 Methodology: Systematic Literature Review Procedures

The systematic literature review was conducted following the guidelines outlined by Webster and Watson [15] and vom Brocke et al. [16]. We provide a detailed description of the literature search below, ensuring transparency of procedures and results' replicability. We begin with presenting the search strategy, keywords, and searched databases. Next, the inclusion and exclusion criteria are discussed.

3.1 Database Search

An initial, non-systematic literature search was conducted to determine the scope of the systematic review. As a result, we identified three concepts focal for the present study: (1) computer science, (2) knowledge transfer, and (3) GDBL. These concepts and associated keywords determined the comprehensive search formula. "Computer science" was supplemented by "programming." Knowledge transfer was broken down into "education," "teaching," "learning," and "e-learning." Lastly, "GDBL" was additionally written out as "game development based learning" and also captured with "game development." Furthermore, we used "wildcards" (asterisks which are typically placed at the end of a word stem) in order to account for possible suffixes and endings that a word might have.

The searched databases were Scopus, IEEEXplore, ACM Digital Library, EBSCO, and ERIC. The keywords had to be mentioned in the abstract. No constraints on the publication period were applied. Although the initial literature search revealed that the term GDBL was first coined in 2012 by Wu and Wang [11], the research on game development for learning purposes goes back in time.

3.2 Eligibility Criteria

The database searches yielded 168 records. Then, in a step-by-step process, some studies were sorted out, as not fulfilling the eligibility criteria; and some studies were added. The process is illustrated with Figure 1.

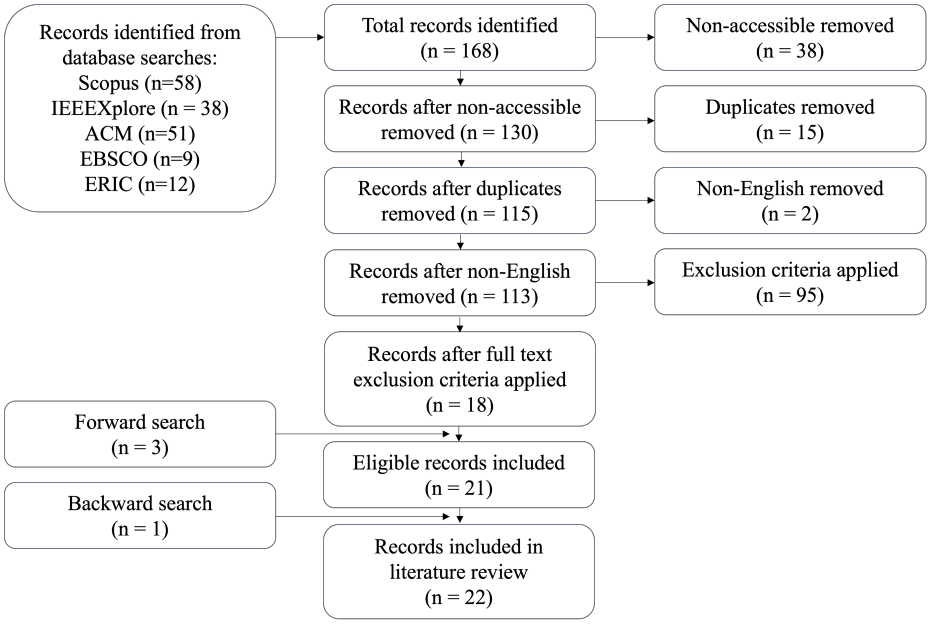

Figure 1. Flow diagram for the paper inclusion process

First, 38 articles that we could not access (including any of their versions, e.g., pre-prints) were excluded, reducing the number of records to 130. In the following step, 15 duplicates were removed. Then, articles written in the non-English language were sorted out: one – in Portuguese and one – in Spanish.

Next, full texts of 113 remaining studies were analyzed and subjected to pre-established inclusion criteria. First, the GDBL educational method had to be applied. Hence, we excluded the papers concerning games and video games in a general sense, as well as the game based learning and serious games methods. Second, articles examining the use of GDBL outside the educational context (i.e., school, college, university, holiday camp with a focus on education) were excluded. Third, we were interested in schoolchildren and students as course participants. Thus, papers focusing on teachers and university instructors were sorted out. Fourth, the type of study was considered. Only empirical academic works were qualified as eligible for the review. If the paper merely described some GDBL setup without presenting any results of its application, the article was excluded. Fifth, the course or material to be taught had to be programming-related: computer science, programming skills, and programming concepts. The papers had to fulfill all five eligibility criteria. As a result, the number of records went down to 18.

Subsequently, we performed forward and backward literature searches that yielded four additional studies satisfying all inclusion criteria (see Figure 1). A total of 22 studies met the criteria and comprised the pool of review articles.

4 Results

In the following, the synthesis of information from the 22 articles is presented. We provide a general overview over the research stream of GDBL for learning to program and then delve into specific characteristics of studies: attributes of GDBL course-setups and learning outcomes.

The earliest article is dated 2008. Most articles (86%) were published after 2012, with almost a quarter – in 2020 and 2021. Hence, the application of GDBL for learning to program increasingly attracts scholarly attention. Judging by sample characteristics, the research evidence comes from eight different countries. More than one-third of studies (8) recruited participants from the USA. Other sample populations are from Brazil (3 papers), Greece (3), Canada (2), Poland (2), Austria (2), Croatia (1), Thailand (1), and the UK (1). This might indicate a higher interest in the GDBL approach to teaching programming and computer science in North American, South American and European countries.

Most articles (14) present quantitative study designs: five one-time surveys were dominated by pre- and post-surveys of learners, e.g., [17–19]. Among the 14 quantitative studies, there are three experimental studies [5, 20, 21], where GDBL represents one of the experimental conditions compared with the conventional learning (control group). Only (2) papers describe qualitative methods [4, 22], i.e., interviews. The remaining (6) adopt a mixed-methods design. Predominantly, GDBL courses were organized offline, with only three studies reporting procedures that took place online [6, 23, 24].

4.1 Educational Setting

Studies under the review present various educational levels at which GDBL programming courses are taught. The educational levels range from middle school [20, 24] through high school [6, 17], to higher education [18, 25]. School systems differ from country to country. Hence, to facilitate comparisons, we combined the papers describing GDBL application at the middle and high school levels in one group. Besides, the middle school level is represented by only two papers [20, 24]. Various educational levels have distinct characteristics, and thus we can refer to them as different educational settings. Hereafter, we differentiate between *school*, *university*, and *summer camp* settings.

School Setting The articles targeting the school setting are characterized by smaller and younger samples and shorter course duration. Specifically, the participants are between 11 and 18 years old, with on average 30% fewer participants than at university courses. A course goes on for a few weeks [23, 26] or months [6, 22] and rarer for a full semester [17]. Importantly, schoolchildren typically have no previous experience in computer science and have not been introduced to programming prior to taking

the GDBL course [4–6, 9, 22, 23, 26]. As Garneli et al. [20, p.36] put it, "Introducing programming to young children is a very challenging process." This is reflected in the GDBL scholars' choices of programming languages for the courses discussed below.

University Setting In the university setting, the average age of participants is expectedly higher and ranges from 18 to 29. In addition, the average duration of courses is longer – typically one semester [8, 18, 27, 28]. The average participants' number is also noticeably higher. Participants of university GDBL-courses are more likely to have some prior experience in programming [7–9, 27–29]. However, the hurdle of making computer science and programming more accessible to novice learners from outside traditional computer science programs remains [18].

Summer Camps Setting GDBL courses at game development summer camps are described in two studies [19, 30]. Participants of these studies were schoolchildren – 4th and 5th graders in [19], 9th and 10th graders in [30] – invited to residential summer camps with a view to increasing their interest in science, technology, engineering, and math (STEM) disciplines. Due to the specific intensive learning schedule in the summer camp setting, we did not merge the summer camp and the school educational setting in one category.

4.2 Programming Languages

Next, we focus on programming languages and integrated development environments (IDEs) used to develop and modify games. Multiple programming languages, frameworks, and methods facilitating GDBL were identified from the articles under review. We divided the languages, IDEs, and frameworks into three groups by their complexity.

Some studies describe applying such frameworks (e.g., Construct2) that do not require the use of any programming language [6, 19, 27]. Specifically, participants did not program via writing code but via "manipulating graphical elements according to the programming logic to be used (events, conditions, actions, etc.)" [23, p.2]. The learning experience was aimed at developing "a programming logic" [23] and applying it to game creation. Such frameworks make up the first group that we refer to as *Code-free*.

Second group, *block-based* programming languages and programming applications (e.g., Blockly [31], Scratch [26, 32, 33], GameSalad [7, 8], and Alice [34]) work with a drag-and-drop mechanism. *Block-based* programming languages consist of individual blocks that can be effortlessly connected and disconnected to create programs and functions. While having all the basic programming concepts, such as loops, variables, conditional statements, functions and comparison operators, these programming languages are characterized by lower complexity. Designed to be particularly visual and child-friendly, *block-based* coding comes with little to no text, which prevents syntactic errors and makes the programming languages suitable for novices [5].

Third group of languages and frameworks used for GDBL is comprised of the *text-based* programming languages such as Java [30, 35], C++ [9], C [9], and Assembly [28]. These programming languages are more abstract and machine-oriented, which makes

them relatively the most complex. The relative language complexity is accompanied by a high degree of flexibility, which is why the *text-based* languages are used by professional developers. Nevertheless, Assembly, C++ and Java are typically considered low-level programming languages or even machine code and therefore often taught in early introductory computer science courses at universities [28].

Not surprisingly, *code-free* and *block-based* programming applications are particularly common in studies targeting younger learners (middle and high school, younger learners at the summer camp [19]). Learners of older age (university students), on the other hand, are predominantly taught *text-based* and sometimes *block-based* programming languages in GDBL classes. Table 1 groups the studies by the educational setting and the type of programming language. The total number of studies per a language-setting combination is provided in brackets. Note that one study by Haselberger et al. [4] uses both a *block-based* and a *text-based* programming language and hence appears twice in Table 1.

Table 1. Grouping by educational setting and the type of programming language

	Code-Free	*Block-Based*	*Text-Based*
School	2 [6, 23]	7* [4, 5, 17, 20, 22, 24, 26]	2* [4, 35]
University	1 [27]	4 [7, 8, 18, 32]	5 [9, 21, 25, 28, 29]
Summer Camp	1 [19]	0	1 [30]
Overall	4 out of 22	11 out of 22	8 out of 22

*Note: The first number in each cell is the count of papers with a given type of programming language and educational setting; * [4] use both a block-based and a text-based programming language.*

4.3 Games

The core of all GDBL courses is mobilizing young people's interests involving computer or mobile games [11]. Table 1 demonstrates that some language types are more common for particular educational settings. In turn, a programming language can determine the games to be programmed or modified within the course. Here again, a dividing line between the school and university setting can be seen.

The studies conducted with schoolchildren predominantly, but not exclusively, rely on *block-based* programming environments such as Scratch, GameSalad, and Construct2, as we have shown in the above subsection. This has an impact on the games. Specifically, simple jump 'n' run games [5, 6, 17] are often developed. Other games are Escape House and a recycling game (where a park has to be kept clean) [24, 26]. These games are simple and particularly visual with a lot of graphic interaction, where a player's skill is not the main focus.

In GDBL programming courses within the university setup, *text-based* programming languages are used most often. The game development frameworks are more technically-demanding when compared to the school setting: Arduino, a programmable mini-computer, Java ME, and Alice [9, 25, 27–29]. Despite their complex setup and

operation, these tools allow creating more sophisticated programs. In most cases, students re-develop and modify so-called arcade games: Tetris, Pac-man, Space Invaders, or Snake [9, 25, 27–29]. These games witnessed extreme popularity in the 1970s and 1980s and are characterized by a plain graphical interface that is all about players' skills and dexterity. The simplicity of the arcade games allows class participants to program their games from scratch rather than modify single missions or graphical features.

When reviewing the games at the summer camp level, we notice the following. Younger learners – 4th and 5th graders – created very visual and simple games with the *code-free* Kodu Game Lab framework [19]. Namely, participants had to create an alien world, fill it with different rocks, and program a Rover (game object) to inspect each rock. Older participants of the [30] study – 9th, 10th graders – were invited to work within the Greenfoot IDE and write Java programs. Participants were asked, for example, to program various Little Red Riding Hood scenarios. Focusing on the game-play, i.e., a game's plot, course instructors aimed to demonstrate that "fun games can be created with modest visual assets" [30, p.56].

Thus, the games too (not only programming language types) differ fundamentally with the educational setting that serves as a proxy for learners' age and prior knowledge. Our review shows that the combination of such games' characteristics as graphics and game-play illustrate the differences the best. Simpler games – more common for the school setting – are visually richer and offer easy game-play (see [17, 24]). When more complexity is introduced in a GDBL-class – more typical for the university setting – the developed games are less visual and more focused on the game-play (see [9, 25, 29]).

4.4 Learning Outcomes of GDBL

Most outcomes investigated in the 22 studies can be subsumed under *assessment of computing skills* and *perceptions of computer science and programming*.

The assessment of computing skills covers, among others, programming skills (see e.g., [7, 21, 24]), knowledge of computer science (CS) and programming concepts [22, 30, 35], and problem solving skills [8]. Table 2 provides an overview. Measuring the outcomes listed in Table 2, authors investigated whether teaching programming languages with the game development based method improves students' skills and comprehension of various concepts. Note that such constructs as programming knowledge can be measured differently and with multiple dimensions, e.g., knowledge of loops structure, conditional structure, variables, etc. [26]. When the results on different dimensions were distinct, we reported mixed results, "+/-" in Tables 2 and 3.

As the name of the second outcomes' group suggests, the *perceptions* include learners' perception of computer science discipline [4, 19], attitudes towards programming [7, 8, 32], perceived usefulness [25] and helpfulness of new knowledge [9]. Other related outcomes are enjoyment [18], interest [29, 30, 35], and motivation to go on with learning [5, 6], as well as experiences of IT roles, i.e., software developer, multimedia designer, programmer [27].

Note that summer camps as an educational setting served a specific purpose of making computer science and STEM disciplines more accessible for young children in general. Hence, both studies on GDBL courses at summer camps appear only in Table 3,

Table 2. Assessment of computing skills

	Authors/Year	*SD*	*ES*	*Outcome*	*Effect*
[8]	Dekhane et al. 2013	qnt*	Uni	Problem solving skills	+
[7]	Dekhane and Xu 2012	qnt	Uni	Programming skills	+
[21]	Poolsawas and Niranatlamphong 2017	qnt*	Uni	Programming skills	+
[35]	Doerschuk et al. 2013	qnt*	Sch	Computing concepts	+
[20]	Garneli et al. 2015	qnt*	Sch	Change in programming habits	+/-
[24]	Holenko Dlab and Hoic-Bozic 2021	qnt*	Sch	Programming skills	+
[22]	Johnson 2017	qlt	Sch	Programming concepts	+/-
[5]	Papadakis 2020	qnt*	Sch	Programming skills	+
[17]	Seaborn et al. 2012	qnt*	Sch	CS concepts comprehension	+
[26]	Seralidou and Douligeris 2020	mix	Sch	Programming knowledge	+/-
[30]	Al-Bow et al. 2008	qnt	Cmp	Programming concepts	+

*Note: ES = Educational Setting, Uni = University, Sch = School, Cmp = Summer Camp; SD = Study Design, qnt = Quantitative, * = Significance Testing Performed, qlt = Qualitative, mix = Mixed-Methods Design; +/- = Mixed Results*

listing outcomes related to students' interests and perceptions. The differences between the school and the university setting are expectedly not that salient when it comes to the examined outcomes. Table 2 and Table 3 demonstrate that the investigated learning outcomes within various educational settings – skills-related and perceptions-related respectively – are mostly comparable. Not surprisingly, such outcomes as interest in a certain career path or experience with particular IT roles or jobs are only observed within the university setting, as there learners are closer to entering the job market. In general, instilling knowledge in schoolchildren and students, as well as building a positive attitude towards the subject of computer science and programming, appears to be the ultimate value and criteria of the educational method's success.

The majority of identified outcomes are positive, and only a few are mixed. For instance, Collier et al. [18, p.157] report that "although the majority [of students] believed the game programming experience enhanced their learning overall, another majority reported that the project itself was not enjoyable." Yet, Fowler and Khosmood [19, p.5] report that their learners perceive the GDBL course as an "enjoyable hardship." These quotes illustrate well the current state of research on GDBL. First, the mixed effects are owed to the various dimensions of the studied constructs and the complexity of the GDBL as a phenomenon. While ratings of some learning aspects are high, others may receive lower scores. Second, we observe that same constructs, e.g., learning enjoyment, can be operationalized with a measurement item [18] or examined with an open-ended question [19]. As a result, the comparison and integration of studies in the extant body of literature, even exploring the same outcomes, is complicated. This issue is especially pronounced in case of *perceptions of computer science and programming* summarized in Table 3. Third, many papers, even employing quantitative design, only report the results of descriptive data analysis (e.g., bar charts and frequency tables). At the same time, we observe that most authors administer pre- and post-treatment surveys of course participants and report frequencies obtained from both time periods. Lastly, a common

Table 3. Perceptions of computer science and programming

Authors/Year	*SD*	*ES*	*Outcome*	*Effect*
[18] Collier and Kawash 2014	qnt	Uni	Own learning assessment	+/-
[8] Dekhane et al. 2013	qnt	Uni	Attitudes towards computing	+
[7] Dekhane and Xu 2012	qnt	Uni	Attitudes towards computing	+
[25] Duch and Jaworski 2018	qnt	Uni	Perceived knowledge usefulness	+
[27] Frydenberg 2015	mix	Uni	Experiencing IT roles	+/-
[9] Perenc et al. 2019	qnt	Uni	Perceived knowledge helpfulness	+
[32] Bittencourt et al. 2015	qnt	Uni	Perception of block-based language	+
[29] Kurkovsky 2009	mix	Uni	Interest in CS career	+
[23] da Silva and da Silva Aranha 2015	qnt	Sch	Learning engagement	+
[6] da Silva and da Silva Aranha 2016	qnt	Sch	Learning motivation	+
[35] Doerschuk et al. 2013	qnt	Sch	Interest in CS	+
[4] Haselberger et al. 2020	qlt	Sch	Perception of CS	+
[22] Johnson 2017	qlt	Sch	Challenging learning	-
[5] Papadakis 2020	qnt*	Sch	Learning motivation	+
[30] Al-Bow et al. 2008	qnt	Cmp	Interest in technology	+
[19] Fowler and Khosmood 2018	mix*	Cmp	Perception of CS	+/-

*Note: ES = Educational Setting, Uni = University, Sch = School, Cmp = Summer Camp; SD = Study Design, qnt = Quantitative, * = Significance Testing Performed, qlt = Qualitative, mix = Mixed-Methods Design; +/- = Mixed Results*

limitation of articles under review is that the data are obtained from one course iteration, e.g., one semester, and hence from a small participants' number (see, for instance, [27]). Only four studies have sample sizes larger than a hundred [9, 18, 24, 25]. Small sample sizes might also be the reason why we do not see more results of statistical testing.

Against this methodological backdrop, we conclude by highlighting the results obtained within experimental designs [5, 20, 21]. In all experiments, GDBL-class represented experimental condition. Students in control condition received traditional teaching instructions. Test scores [5, 20, 21] and course deliverables, i.e., post test' projects, were used to assess students' achievements [20]. The studies [5, 21] demonstrate effectiveness of the GDBL educational method. Garneli et al. [20] present more nuanced results. The control group "mostly experimented with more complex programming curricula [conditionals, variables, operators, and sequence primitives], while the game development group improved in all the primitive categories [loops, coordination/synchronization/event handlers, sensing, and sequence primitives" [20]. Statistically significant difference was additionally found in the control group as they used more variables and operators and made more errors in comparison with the video game development groups.

5 Discussion

School teachers face the challenge of making programming more accessible to novices without any prior computing experience [4–6, 9, 22, 23, 26]. Participants of university

GDBL-courses are more likely to have some related knowledge [7–9, 27–29]. Yet, instructors at higher education institutions battle the skeptical, dismissive attitude of students [32], declining enrollment rates at STEM disciplines and high drop-out rates at programming courses [7–9]. Adoption of GDBL is considered as a way to lower entry barriers and increase students' engagement. Game development makes lectures less theoretical, more application-oriented, and engaging for students [28].

This paper answers two research questions concerning the GDBL approach. To answer the first question *"What are the attributes of GDBL-based programming courses?"*, we focused on the characteristics of game development set-ups described in empirical studies. As a result, the following key attributes were identified: (1) educational setting, (2) type of programming language, (3) games to be developed or modified. We conclude that the way of implementing GDBL depends on the education level.

Schoolchildren mostly work with block-based programming environments, such as Blockly [31], Scratch [26, 32, 33], GameSalad [7, 8], and Alice [34]. As a result, they develop and modify small visual games and missions. Much of the computing power is taken up by game development frameworks. The visual and playful components tie together young learners' interests with computer science [29].

At the Higher Education level, GDBL-based programming courses employ higher-level text-based programming languages such as Java [30, 35], C++ [9], C [9], and Assembly [28]. The setup and application of the tools for game development are more complex. On the other hand, study participants get the opportunity to program their games from scratch and with more sophisticated game-play. Tetris, Pac-man, Space Invaders, and Snake (different arcade games) are often re-developed [9, 25, 27–29].

Interestingly, only three out of 22 papers under review arranged online game development based learning classes. Most schools and universities continue to offer in-class courses. The COVID-19 pandemic, however, forces instructors to create digital educational content and to convert their teaching to digital in order to continue the provision of educational services. Hence, there is a need for more evidence on online GDBL programming and computer science courses.

To address the second research question *"What are the outcomes of learning a programming language through the GDBL educational approach?,"* we reviewed the results of reviewed articles. Our review gives a positive picture of learning to program with the GDBL educational approach. While all studies show at least some positive effect [18,20,22,26,27], a majority of papers reports exclusively positive ones [5,8,23,28]. The positive effects manifested in higher test scored of the GDBL experimental groups compared to conventional learning groups [5, 21] strengthen the argument that GDBL is an effective educational method.

All investigated learning outcomes can be subsumed under computing skills (e.g., problem solving [8], programming skills [24], comprehension of computer science concepts [17]) and perceptions of computer science and programming (students' interest [29], motivation [6], and attitudes [8]). Within different educational settings (summer camp, school, university), teachers aim to open up the world of programming to young people in a sustainable way – in that learners are not deterred by technical or theoretical hurdles but are picked up where their interests lie [11]. Hence, regardless of the educational setting, comparable learning outcomes are investigated.

The reviewed articles present qualitative, quantitative, and mixed-methods designs, but quantitative surveys and descriptive analyses prevail. Only three papers conducted experiments [5, 20, 21]. Two papers relied on qualitative interviews with schoolchildren for deeper insights into the advantages of game development based learning [4, 22]. GDBL represents a complex phenomenon involving teaching staff, students, and game development frameworks. Hence, after almost two decades of research into GDBL for programming and computer science education – summarized in [11] (published in 2012) and this paper – there is still a need for empirical evidence. Collecting sufficient data at multiple time points (for tracking within-person changes in knowledge, skills, and perceptions towards programming) for the application of appropriate statistical analyses represents a perspective avenue for future research. Experimental studies into the effects of GDBL, compared to traditional teaching instructions, are needed most as they allow for causal inference. Lastly, changes in technology use [29] drive changes in game-related interests of students (from the popularity of computer games to mobile games) and consequently affect the choice of game development frameworks [5]. Thus, advances in the technology for gaming translate into potentials for GDBL research.

We believe that our synthesis of the papers on programming with GDBL will introduce interested teachers and university instructors to appropriate tools for a given education level: programming languages, integrated development environments and other materials necessary for game-development in the class. Informing interested parties about recent developments in teaching programming and computer science and equipping them with the tools that have been applied in practice represent crucial steps for delivering more accessible training to new generations of students.

6 Conclusion

As digitalization advances, young people face new challenges. Day-to-day activities [2] and especially new requirements posed by the job market [3] increasingly demand expertise in the area of programming and computer science. The accessible and sufficient training in corresponding areas is of immense importance. As a result, novel game-based educational approaches and, in particular, game development based learning [11] have emerged and are applied in STEM disciplines. The GDBL approach allows instructors to engage learners by meeting them where their interests lie, namely, computer and mobile games [36]. We conducted a systematic literature review [15, 16] of studies applying the GDBL approach within programming courses and investigating its learning outcomes. Our results based on 22 research papers suggest that the use of GDBL in computer science education is an effective educational approach. Nevertheless, there exist a need for more research evidence and methodological rigor.

7 Acknowledgements

This work has been funded by the Federal Ministry of Education and Research of Germany (BMBF) under grant no. 16DII127 ("Deutsches Internet-Institut").

References

1. Statista: Number of smartphone users from 2016 to 2021 (2021), https://www.statista.com/statistics/330695/number-of-smartphone-users-worldwide/
2. Koetsier, J.: Global Online Content Consumption Doubled In 2020 (2020), https://www.forbes.com/sites/johnkoetsier/2020/09/26/global-online-content-consumption-doubled-in-2020/?sh=2a82336c2fde
3. Curtarelli, M., Gualtieri, V., Shater Jannati, M., Donlevy, V.: ICT for work: Digital skills in the workplace (2014)
4. Haselberger, D., Motschnig, R., Comber, O., Mayer, H., Hörbe, M.: Experiential Factors Supporting Pupils' Perceived Competence In Coding - An Evaluative Qualitative Content Analysis. In: 2020 IEEE Frontiers in Education Conference (FIE). pp. 1–9 (2020)
5. Papadakis, S.: Evaluating a game-development approach to teach introductory programming concepts in secondary education. International Journal of Technology Enhanced Learning 12(2), 127–145 (2020)
6. Da Silva, T.R., Da Silva Aranha, E.H.: An empirical study of online K-12 education for programming games. Proceedings - IEEE 16th International Conference on Advanced Learning Technologies, ICALT 2016 pp. 255–259 (2016)
7. Dekhane, S., Xu, X.: Engaging students in computing using GameSalad: a pilot study. Journal of Computing Sciences in Colleges 28(2), 117–123 (2012)
8. Dekhane, S., Xu, X., Tsoi, M.Y.: Mobile app development to increase student engagement and problem solving skills. Journal of Information Systems Education 24(4), 299–308 (2013)
9. Perenc, I., Jaworski, T., Duch, P.: Teaching programming using dedicated Arduino Educational Board. Computer Applications in Engineering Education 27(4), 943–954 (2019)
10. Maurício, R.d.A., Veado, L., Moreira, R.T., Figueiredo, E., Costa, H.: A systematic mapping study on game-related methods for software engineering education. Information and software technology 95, 201–218 (2018)
11. Wu, B., Wang, A.I.: A guideline for game development-based learning: a literature review. International Journal of Computer Games Technology 2012 (2012)
12. Alhammad, M.M., Moreno, A.M.: Gamification in software engineering education: A systematic mapping. Journal of Systems and Software 141, 131–150 (2018), https://doi.org/10.1016/j.jss.2018.03.065
13. Pedreira, O., García, F., Brisaboa, N., Piattini, M.: Gamification in software engineering–a systematic mapping. Information and software technology 57, 157–168 (2015)
14. Krusche, S., Reichart, B., Tolstoi, P., Bruegge, B.: Experiences from an experiential learning course on games development. In: Proceedings of the 47th ACM Technical Symposium on Computing Science Education. pp. 582–587 (2016)
15. Webster, J., Watson, R.T.: Analyzing the Past to Prepare for the Future: Writing a Literature Review. MIS Quarterly 26(2), xiii—-xxiii (2002)
16. vom Brocke, J., Simons, A., Niehaves, B., Niehaves, B., Reimer, K., Plattfaut, R., Cleven, A.: Reconstructing the giant: On the importance of rigour in documenting the literature search process (2009)
17. Seaborn, K., Seif El-Nasr, M., Milam, D., Yung, D.: Programming, PWNed: Using Digital Game Development to Enhance Learners' Competency and Self-Efficacy in a High School Computing Science Course. In: Proceedings of the 43rd ACM Technical Symposium on Computer Science Education. pp. 93–98. SIGCSE '12, Association for Computing Machinery, New York, NY, USA (2012), https://doi.org/10.1145/2157136.2157169
18. Collier, R., Kawash, J.: Lessons learned and recommended strategies for game development components in a computer literacy course. In: SIGCSE 2014 - Proceedings of the 45th ACM Technical Symposium on Computer Science Education. pp. 157–162 (2014)

19. Fowler, A., Khosmood, F.: The Potential of Young Learners Making Games: An Exploratory Study. In: 2018 IEEE Games, Entertainment, Media Conference (GEM). pp. 1–9 (2018)
20. Garneli, V., Giannakos, M.N., Chorianopoulos, K., Jaccheri, L.: Serious Game Development as a Creative Learning Experience: Lessons Learnt. In: 2015 IEEE/ACM 4th International Workshop on Games and Software Engineering. pp. 36–42 (2015)
21. Poolsawas, B., Niranatlamphong, W.: Using a game development platform to improve advanced programming skills. Journal of Reviews on Global Economics 6, 328–334 (2017)
22. Johnson, C.: Learning Basic Programming Concepts with Game Maker. In: International Journal of Computer Science Education in Schools, v1 n2 May 2017 (2017)
23. Da Silva, T.R., Da Silva Aranha, E.H.: Online game-based programming learning for high school students - A case study. Proceedings - Frontiers in Education Conference, FIE 2015 (2015)
24. Holenko Dlab, M., Hoic-Bozic, N.: Effectiveness of game development-based learning for acquiring programming skills in lower secondary education in Croatia. Education and Information Technologies (2021)
25. Duch, P., Jaworski, T.: Enriching computer science programming classes with arduino game development. In: Proceedings - 2018 11th International Conference on Human System Interaction, HSI 2018. pp. 148–154 (2018)
26. Seralidou, E., Douligeris, C.: Learning programming by creating games through the use of structured activities in secondary education in Greece. Education and Information Technologies 26(1), 859–898 (2021), https://doi.org/10.1007/s10639-020-10255-8
27. Frydenberg, M.: Achieving digital literacy through game development: an authentic learning experience. Interactive Technology and Smart Education 12(4), 256–269 (2015)
28. Kawash, J., Collier, R.: Using video game development to engage undergraduate students of assembly language programming. In: SIGITE 2013 - Proceedings of the 2013 ACM SIGITE Annual Conference on Information Technology Education. pp. 71–76 (2013)
29. Kurkovsky, S.: Can mobile game development foster student interest in computer science? In: 2009 International IEEE Consumer Electronics Society's Games Innovations Conference. pp. 92–100 (2009)
30. Al-Bow, M., Austin, D., Edgington, J., Fajardo, R., Fishburn, J., Lara, C., Leutenegger, S., Meyer, S.: Using Greenfoot and Games to Teach Rising 9th and 10th Grade Novice Programmers. In: Proceedings of the 2008 ACM SIGGRAPH Symposium on Video Games. pp. 55–59. Sandbox '08, Association for Computing Machinery, New York, NY, USA (2008), https://doi.org/10.1145/1401843.1401853
31. Fraser, N.: Google blockly-a visual programming editor (2014)
32. Bittencourt, R.A., dos Santos, D.M.B., Rodrigues, C.A., Batista, W.P., Chalegre, H.S.: Learning programming with peer support, games, challenges and scratch. In: 2015 IEEE Frontiers in Education Conference (FIE). pp. 1–9 (2015)
33. Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., Millner, A., Rosenbaum, E., Silver, J., Silverman, B., Others: Scratch: programming for all. Communications of the ACM 52(11), 60–67 (2009)
34. Cooper, S., Dann, W., Pausch, R.: Alice: a 3-D tool for introductory programming concepts. Journal of computing sciences in colleges 15(5), 107–116 (2000)
35. Doerschuk, P., Juarez, V., Liu, J., Vincent, D., Doss, K., Mann, J.: Introducing programming concepts through video game creation. In: 2013 IEEE Frontiers in Education Conference (FIE). pp. 523–529 (2013)
36. Statista: Share of 13- to 18-year-olds who play video games every day in the United States as of April 2019, by platform (2019)