Computers in Human Behavior 99 (2019) 415–427

Contents lists available at ScienceDirect

Computers in Human Behavior

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

Full length article

Exploring children's learning experience in constructionism-based coding activities through design-based research

Sofia Papavlasopoulou*, Michail N. Giannakos, Letizia Jaccheri

Department of Computer Science, Norwegian University of Science and Technology (NTNU), Trondheim, Norway

ARTICLE INFO

Keywords:
Constructionism
Coding
Computational thinking
Engagement
Children
Design-based research

ABSTRACT

Over the last few years, the integration of coding activities for children in K-12 education has flourished. In addition, novel technological tools and programming environments have offered new opportunities and increased the need to design effective learning experiences. This paper presents a design-based research (DBR) approach conducted over two years, based on constructionism-based coding experiences for children, following the four stages of DBR. Three iterations (cycles) were designed and examined in total, with participants aged 8–17 years old, using mixed methods. Over the two years, we conducted workshops in which students used a block-based programming environment (i.e., Scratch) and collaboratively created a socially meaningful artifact (i.e., a game). The study identifies nine design principles that can help us to achieve higher engagement during the coding activity. Moreover, positive attitudes and high motivation were found to result in the better management of cognitive load. Our contribution lies in the theoretical grounding of the results in constructionism and the emerging design principles. In this way, we provide both theoretical and practical evidence of the value of constructionism-based coding activities.

1. Introduction

There is growing evidence supporting the introduction of computer science (CS) and computational thinking (CT) into K-12 education (Hubwieser, Armoni, Giannakos, & Mittermeir, 2014); (Horizon, 2015). According to Wing (2006 p.33) *"CT represents a universally applicable attitude and skill set everyone, not just computer scientists, would be eager to learn and use"*. CT involves problem solving, design of systems and understanding human behavior by employing central concepts of CS (Wing, 2006). Organizations like the Computer Science Teachers Association (CSTA), Informatics Europe, the Cyber Innovation Center, and the National Math and Science Initiative have developed standards applied to coding education (Hubwieser et al., 2015). Increasing interest in learning coding in pedagogical contexts has also been driven and disseminated by organizations like Code.org and Codeacademy, which argue for the need to create skills that support future career opportunities while highlighting the educational advantages that coding offers. CT and coding in education have become integral parts of the school curricula in many countries. For example, the United Kingdom has integrated computer programming as a mandatory course starting from primary school (Jones, 2013), while Denmark promotes digital literacy, focusing on the knowledge gained from building technologies (Tuhkala, Wagner, Nielsen, Iversen, & Kärkkäinen, 2018).

Pioneered by Seymour Papert (Papert, 1980), computer programming in education has received a lot of interest from educators and researchers seeking alternative ways of teaching complex problem-solving skills and providing dynamic learning experiences (Kalelioğlu, 2015; Lye & Koh, 2014). Nowadays, there are a variety of technological tools and child-friendly programming environments (Papavlasopoulou, Giannakos, & Jaccheri, 2017b). Many introductory experiences for K-12 have been designed around the use of block-based programming environments, such as Scratch, Alice, Blocky, and App Inventor (Zhang, Liu, de Pablos, & She, 2014); (Fields, Vasudevan, & Kafai, 2015) (Wagner, Gray, Corley, & Wolber, 2013). These environments do not require any special expertise but do require careful thinking to tell the computer what to do step by step. Papert's (1980) constructionism argues that through coding, children have an "object-to-think-with"; in the process, they learn about their own thinking (Guzdial, 2004). Constructionism-based learning activities have been widely studied in both formal and informal education (Papavlasopoulou, Giannakos, & Jaccheri, 2017a). Integrating coding into pedagogical contexts in an intuitive and engaging experience enhances children's logic, critical thinking, problem-solving, math, and higher-order skills and can change their attitudes towards computing (Sáez-López, Román-González, & Vázquez-Cano, 2016); (Kafai & Burke, 2015). There are strong arguments for children to learn how to code, supported by the

* Corresponding author. Sem Sælands vei 7-9, NO-7491 Trondheim, Norway.
E-mail addresses: spapav@ntnu.no (S. Papavlasopoulou), michailg@ntnu.no (M.N. Giannakos), letizia.jaccheri@ntnu.no (L. Jaccheri).

https://doi.org/10.1016/j.chb.2019.01.008
Received 2 May 2018; Received in revised form 26 December 2018; Accepted 13 January 2019
Available online 16 January 2019
0747-5632/

constructionist approach (Kafai & Burke, 2015) (Gallup, 2015). Children need to acquire 21st-century skills, empowering themselves with the required competences related to the digitalization of our society. Learning how to code has become equally valuable as learning math, reading, and writing (Horizon, 2015).

Several studies have focused on introducing computational literacy to children in different ways (Papavlasopoulou et al., 2017a). Various programmable and interactive objects exist showing the importance of involving children from a young age in learning coding (Fessakis, Gouli, & Mavroudi, 2013). In addition, environments like LiliPad Arduino (Buechley, Eisenberg, Catchen, & Crockett, 2008) have been developed to attract more girls to CS and CT. The combination of physical fabrication and coding has proven valuable for increasing engagement in programming concepts and practices (Kafai & Vasudevan, 2015), especially when it incorporates social and creative dimensions of learning (Giannakos & Jaccheri, 2018). In a study with sixth-grade students in Scotland, Robertson and Howells (Robertson & Howells, 2008) found that making a game is an authentic learning activity offering motivation, enthusiasm, and engagement with learning. Therefore, to overcome the various barriers with learning coding (e.g., difficulty, boredom, confusion, etc.), we need appropriately designed and engaging coding activities for children.

Constructionism theorizes that learner is seen as an active constructor of knowledge rather than being a passive recipient of information (Papert, 1993), with making and coding being the areas that constructionism theory has been widely applied (Kafai & Burke, 2015). Almost three decades after Papert's original ideas on constructionism, the idea remains relevant and has become ubiquitous in how learning theorists and educators aim to empirically ground and revamp constructionism-based teaching (Kao & Harrell, 2017). Such grounding would result in methodological advancements and a comprehensive understanding of children's experience in constructionism-based making activities. In this paper, we present a design-based research (DBR) effort comprising three cycles (iterations) conducted over two years. DBR combines empirical educational research with theory-driven design in learning contexts to understand how, when, and why educational innovations work in real settings (Collins, 1992). The main characteristic of DBR is the systematic and iterative cycle of design, exploration, and redesign (Collective, 2003). Many studies have used DBR in educational contexts (Grover, Pea, & Cooper, 2015; Parmaxi & Zaphiris, 2015) (Sáez-López et al., 2016); (Parmaxi, Zaphiris, & Ioannou, 2016) (Schmitz, Klemke, Walhout, & Specht, 2015), emphasizing the need for well-designed studies characterized by objectivity, reliability, and validity and providing critical evidence to establish outcomes beneficial for others.

This research aims to contribute to the theoretical notions of constructionism with regard to the effects of coding activities on children's learning experience. We designed and evaluated coding workshops for children (aged 8–17 years old). Both qualitative and quantitative methods were employed to evaluate our workshops, including interviews, surveys, observations, and physiological data (eye tracking). The coding activities were designed to impact children's learning outcomes, cognition, and social and emotional development. Thus, the overreaching goal of the study was framed with the following research questions:

- What elements of engagement exist in constructionism-based coding activities?
- What principles can guide us to facilitate constructionism-based learning environments that support children's learning experience?

The rest of the paper is structured as follows: the next section provides an overview of related work on the theoretical framework of constructionism and previous research on similar coding activities. The third section describes the methodology used, the designed coding activities in the three cycles, and the data collection and analysis. The fourth section presents the results based on the theory of constructionism and the main design principles that guided each of the iterations. In the fifth section, we discuss and highlight the design implications, derived from this intervention research. We conclude with the limitations of our study and avenues for future work.

2. Related work

2.1. Theoretical framework: constructionism

Our theoretical grounding is constructionism, which was developed by Papert (Papert, 1997), (Papert, 1980). Constructionism assumes that knowledge is better gained when children are deeply and actively involved in building their own meaningful constructions. Based on Piaget's (1954) theory, which focuses on how mental constructions are formed in someone's mind, Papert (Papert, 1980) focuses on explaining how construction is a valuable way to create mental constructions. The learner discovers their own knowledge, rather than being a passive receiver. Papert's constructionism sees the effectiveness of learning as achieved through making, where learners experience the active construction of visible-to-the-world artifacts. Computational culture supports the creation of building those artifacts by using digital media and computer-based technologies (Kafai & Resnick, 2012). The vital aspect of constructionism is the requirement of "objects-to-think-with" – *"objects in which there is an intersection of cultural presence, embedded knowledge and the possibility for personal identification"* (Papert, 1980), p. 11). The role of this object in Papert's *Mindstorms* is the "turtle", a digital animal within the Logo programming environment that can be controlled and moved by giving the appropriate commands. The "turtle" acts as a means to think, supporting and promoting a new way of thinking and learning. In Papert's (1980, p. 76) words: "*the child's encounter with this theorem is different in several ways from memorizing its Euclidean counterpart 'the sum of the internal angles of a triangle is 180 degrees.' First (at least in the context of Logo computers), the Total Turtle Trip Theorem is more powerful: The child can actually use it. Second, it is more general: It applies to squares and curves as well as to triangles. Third, it is more intelligible: Its proof is easy to grasp. And it is more personal: You can 'walk it through,' and it is a model for the general habit of relating mathematics to personal knowledge."*

Constructionism is not only valuable for the individual in building knowledge through experience and engagement in creating artifacts but also for enhancing the social setting (Kafai, 2006). Like in the well-known samba school example, a social setting strengthens the sense of belonging to a group with a common purpose, where learning becomes important for all and connections are made under the learning culture (Papert, 1980). In the same line (Kafai & Burke, 2015), mention three dimensions of constructionism involved in the process of making games for learning: personal, social, and cultural. More specifically, "personal" refers to the learning and the attitudes related to learning, "social" refers to the collaborative aspects in creating a shared artifact, and "cultural" relates to how gender and race could influence the activity and the possible cultural aspects that could influence participation.

In the process of making computer games, children plan and manage this complex development, placing themselves in control of their own leaning and thinking (Kafai & Kafai, 1995). Robertson and Howells (Robertson & Howells, 2008) argue that game design is a powerful learning activity that provides motivation, engagement, and enthusiasm. Constructionism's basic idea is that the most effective leaning experiences are those that include active creation, socially meaningful artifacts, interaction with others, and the use of elements that support one's own learning and thinking. Game-making activities not only involve learning how to use technological tools but also using these tools to discover new ways of thinking. In such activities, children are introduced to a culture that permits them to become producers of their own artifacts while building their knowledge in a social context.

2.2. Qualities of constructionism-based coding activities for children

Computer game design and development have been increasingly introduced in both formal and informal educational settings, supporting everything from programming courses and STEM educational topics to broader contexts of problem solving and arts (Papavlasopoulou et al., 2017a). The various technological tools available nowadays allow us to support learning activities based on constructionism and provide meaningful learning experiences for children. In these types of educational activities, children are the protagonists, as they have control of their own products. Coding activities for children not only relate to CS but also allow the development of computational competences and higher-order thinking skills (Grover & Pea, 2013). Children who actively participate in game-making activities enhance, among others, their problem-solving, critical thinking, CT, and collaborative skills (Papavlasopoulou et al., 2017a); (Grover & Pea, 2013).

The benefits of educational activities in which children use technological tools and digital fabrication to construct their own games are many and vary from learning programming concepts to behavioral and perceptual changes towards career paths in computing (Sáez-López et al., 2016) (Kafai & Vasudevan, 2015); (Denner, Werner, & Ortiz, 2012). Making games can be more beneficial than other project-based activities, supporting learning about storytelling, artwork, sound, mechanics, and math (Sung & Hwang, 2013). Moreover, children are familiar with video games from an early age (Granic, Lobel, & Engels, 2014). Visual programming environments provide opportunities for children to be introduced to programming concepts; owing to the fun and usefulness of the activity, children are highly motivated and have positive attitudes towards coding (Sáez-López et al., 2016). Block-based visual programming languages (like Scratch) have the advantage of using shapes that fit properly only when they make a logical sequence of orders. This gives relief to users and saves them from much of the heartache traditionally forced on learners by textual languages (Wilson & Moffat, 2010), p. 70). However, even advanced text-based programming languages like Java have been used to engage children aged 9–10 in coding (Esper, Foster, Griswold, Herrera, & Snyder, 2014). A combination of physical fabrication and coding can engage and enhance children's competences in programming concepts (e.g., loops, conditionals, and events) and practices (e.g., remixing, testing, and debugging) (Kafai & Burke, 2015); (Denner et al., 2012). In addition, digital game development was found to be beneficial for special education students, increasing their problem-solving skills through a process of representation, planning, execution, and evaluation of an artifact (Ruggiero & Green, 2017). Hence, further empirical studies are needed to investigate the different aspects and advantages of constructionism-based activities.

Gender discrepancy in coding has been related to negative educational experiences in early childhood (Teague, 2002). CS careers still tend to be highly stereotyped, with girls being less likely to choose this career path. However, studies have found that both girls and boys who get involved in different kinds of software development practices show a better understanding of and positive attitudes towards CS (Bonner & Dorneich, 2016); (Eordanidis, Gee, & Carmichael, 2017); (Robertson, 2013); (Papavlasopoulou, Sharma, Giannakos, & Jaccheri, 2017). Scaffolding examples can help girls' engagement and confidence when using a programming environment. Studies specifically focusing on girls have found that game design experiences intended to enhance computational skills affect their perceptions in seeing themselves as able to design computer games and encourage them to pursue careers in CS-related professions (Stewart-Gardiner, Carmichael, Latham, Lozano, & Greene, 2013). In a study involving middle-school girls creating games (Denner et al., 2012), found that they were engaged in the process and demonstrated adequate levels of complex programming activity. Thus, designing appropriate activities can be a promising approach to attracting and encouraging girls' interest in computing.

Generally, the skills gained in these educational contexts go beyond the use of a technological tool for making a game and CT. For instance, when children negotiate artifact construction in a supportive environment, they gain a sense of self-efficacy and belief in their capacities; they learn how to solve a problem, manage difficulties, cope with "failure", share resources, and communicate with peers (Chu, Schlegel, Quek, Christy, & Chen, 2017); (Çakır, Gass, Foster, & Lee, 2017); (Bers, 2012). These practices exist in constructionist learning and can be applied in subjects like math, language, arts, and others. The value is in the transferable skills uncovered through the experience of completing a successful project.

In a nutshell, constructionism-based coding activities, particularly when the focus is on game-making, provide a fruitful learning environment in which children are stimulated to use a technological tool, affecting their learning experience. Therefore, there is a need to investigate and get a deep understanding of how we can help learners to acquire knowledge, skills, and competences in coding in an engaging and meaningful manner.

3. Methodology

3.1. Design based research (DBR)

DBR is a systematic but agile methodology widely used in educational contexts (Fig. 1) (Anderson & Shattuck, 2012) (Wang & Hannafin, 2005); (Reeves, 2006). DBR offers a strategy to understand learning processes through design, exploration, enactment, evaluation, and redesign (Anderson, 2005). DBR is a hybrid method, as it is not a replacement of other methodologies but builds on the use of multiple procedures and methods from both design and research methodologies (Wang & Hannafin, 2005). The purpose of DBR is to influence real educative interventions and validate theoretical concepts. The difference between DBR and formative assessment is that it also has a theoretical goal (Barab & Squire, 2004). Researchers are actively involved and maintain constant collaboration with participants, other researchers, and practitioners to manage the research process in real-world settings. Their aim is to implement interventions with refined and improved designs that influence practice. In short, there are five basic characteristics of DBR: 1) it refines theory and practice, 2) it happens in real-world settings and is grounded in relevant contexts, 3) it is interactive, iterative, and flexible, 4) it uses mixed methods in accordance with potential new needs and emerging issues, and 5) it is contextual, meaning that the research findings are connected with the design process (Wang & Hannafin, 2005).

In our approach, based on all the above, we used constructionism theory and applied the DBR methodology to guide our iterations. More specifically, our research process used DBR methodology as it deals with the complexity of real-world educative contexts (in our case coding workshops) and it is grounded in theory (in our case constructionism theory). In addition, DBR approach is in line with the needs of our study, allowing a long period of time with continuous design, evaluation and redesign of our interventions. In this way, we had also the opportunity to conduct iterative and flexible revisions of the research design applying research methods from both qualitative and quantitative research. DBR methodology needs a detailed and comprehensive documentation of the whole process; this action helped

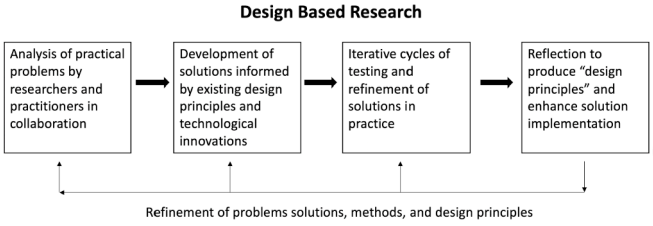

Fig. 1. The research cycle of DBR (Reeves, 2006).

the analysis of our data and especially the retrospective analysis, both to contribute to theory and practice. For all the four stages of the DBR (Fig. 1), constant collaboration with other researchers, experts in the field and instructors is required; this was essential aspect of our study in order to be able to improve the impact of the interventions, understand the learning experience processes, advance the initial designs and provide theoretical and practical impact extracting design principles.

We conducted three cycles (iterations) over two years, evaluating and refining our coding workshops with children. We applied theoretically and pedagogically aligned tasks to investigate their effectiveness on children's learning engagement, overall learning experience, and collaboration while developing an artifact (a game in our approach).

The main aspects of this study were: 1) the design of the coding workshops to facilitate children's use of the programming tool and to introduce them to coding, 2) the researchers working in close collaboration with the participants and assistants who ran the workshops, 3) the use of different methods to evaluate the effectiveness of our approach to increase the sustainability and scalability of this program, 4) grounding our findings in theory, and 5) identifying general design principles for future similar activities.

3.2. Description of the workshops

The participants' goal was to create an artifact, which in our case was a game using the Scratch programming tool. Students worked in teams for the development of the artifact. Teaching assistants, specifically trained, led the process and assisted students in achieving their goals.

Regarding the process of construction in the workshops, the most influential to our pedagogical approach was what Resnick calls the “kindergarten approach to learning”, with a spiral cycle of imagine, create, play, share, and reflect – a process that is repeated over and over (Resnick, 2007). Children imagine what they want to do and then create a project with their ideas, play/interact with their own creations, share their creations with others, and reflect on their experiences, leading to new ideas and projects. Adapting Resnick's spiral, ours also started with “inspire” to characterize the warming-up and inspiring activities that kicked off the children's creativity. In addition, to characterize the coding process and the use of the Scratch tool specifically, we focused on constantly experimenting and iterating: the children developed their artifacts gradually by trying new elements, using different concepts, and revising them (Fig. 2).

3.2.1. Cycles 1 and 2

We designed and implemented a coding activity in conjunction with an initiative organized at the Norwegian University of Science ans Technology (NTNU), in Trondheim, Norway, named Kodeløypa (meaning “the path towards coding”). The workshop activities were based on the constructionist approach, as one of the main principles of this is learning by making. The workshop was conducted in a largely informal setting, as an out-of-school activity, and lasted for four hours in total. Various student groups, in the range 8–17 years old, were invited to NTNU's specially designed rooms for creative purposes to interact with digital robots and to create games using Scratch and the Arduino hardware platform. Specifically, Arduino was attached to the digital robots to connect them with the computer. At that point, an extension of Scratch called Scratch for Arduino (S4A) provided the extra blocks needed to control the robots. The Scratch programming language uses colorful blocks grouped into categories (motion, looks, sound, pen, control, sensing, operators, and variables), with which children can develop stories, games, and any type of animation. In general, the children who attended the workshop worked collaboratively in triads or dyads (depending on the number of children). The workshop was designed for children without (or with minimum) previous experience in coding. The design of the activity (interacting with robots and creating games), and the use of Scratch programming language (suitable for all ages) provided flexibility and allowed the successful implementation of the workshop with participants from 8 to 17 years old students. Each of the workshops, had a specific age group of students, carefully selected, being within a small age range. During the workshop, student assistants were responsible for supporting each team as needed. Approximately one assistant observed and helped one or two teams. Three researchers were also present throughout the intervention, focusing on observing, writing notes, and taking care of the overall execution of the workshop. The workshop had two main sections (Fig. 3).

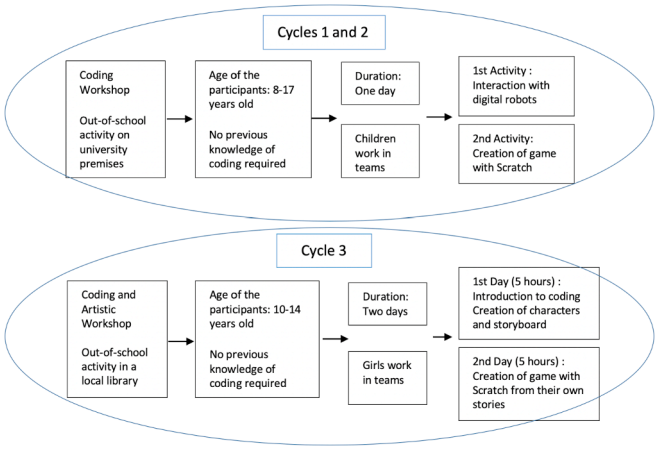

Fig. 2. Description of the three DBR cycles.

Interacting with the robots: In the first section, the children interacted with digital robots made by an artist (using recycling materials). The different robots were placed next to the computers (one for each team). When the children entered the room, one assistant welcomed them, told them to be seated, and briefly presented an overview of the workshop. The assistants then advised the children to pay attention to the paper tutorial and the worksheets placed on the desks (one for each student). First, the children filled in the worksheet to answer questions regarding the exact places and numbers of sensors and lights on the robots. The tutorial contained instructions with examples and pictures, similar to the robots they were using. The examples had little text and more images and described exactly how the children could interact with the robots. The children accomplished a series of simple loops that controlled the robots and made them react to the environment with visual effects (such as turning on a light when sensors detected that the light was below a certain threshold). Children could touch and play with the robots but not change any parts of them. Although the duration of the session was different for each team, it lasted between 45 min and 1.5 h and ended with a break before the next session.

Creating games using Scratch: This session focused on the creative implementation of simple game development concepts using Scratch. All children took another paper-based tutorial containing examples and visualizations to help them to ideate their own games. The tutorial comprised simple text explanations and included basic CT concepts and possible loops that the children were supposed to use in their own games. First, the assistants advised the children to concentrate on understanding the idea of the game, to discuss it with their team members, and to create a draft storyboard. The children then developed their own games by collaboratively designing and coding using Scratch. To accelerate the children's progress, they were given existing game characters and easy loops. While the children worked on their projects, help was provided whenever they asked for it, and complex programming concepts were introduced on an individual level according to the relevance to their project. Children created their games step by step by iteratively testing and coding them. After completing the games, all teams reflected on and played each other's games. This section lasted approximately three hours.

Fig. 3. Example of an interactive robot (left), children collaborating on game creation (middle), and example of a created game (right).

3.2.2. Cycle 3

We designed and implemented a two-day workshop in conjunction with the local library of Trondheim, Norway. The workshop activities focused on coding including artistic elements and were based on the constructionist approach. The call for participation was made to middle-school girls of the Trondheim region during the autumn 2017 school break. Previous experience was not a prerequisite for the girls' participation. The activities of each day were conducted in an informal setting and lasted for approximately five hours, including breaks. Female instructors, with previous experience in similar activities (also involved in Kodeløypa), facilitated the workshop and were responsible for supporting the girls during the process. During the workshop, the girls had to create storyboards based on solving particular environmental problems and then, based on their stories, create games using the Scratch programming language (Fig. 4). For the development of the storyboards, the girls could use different types of materials, like ribbons, colored cardboard, stickers, drawing pencils, etc., as provided by the library. The girls worked collaboratively in teams of two or three (depending on the number of participants). Two researchers were present for the whole duration of the workshop, assisting when needed for the smooth execution of the activities, including the data collection. The workshop is described below, based on the two days of activities.

First day of the workshop: On the first day, we introduced the basic skills of coding and other non-technical aspects of game development, like storyboard creation. The workshop started with a story from a book, based on a woman with children and everyday problems, who was also a mentor and a superhero helping people to succeed with their technology projects. The girls were inspired and were informed that they had to think of their own characters who needed to save the world from environmental issues of their choosing. Then, in order to give an introduction to coding, the instructors presented an example of a functional game with Scratch on a relevant environmental topic. Then, the girls were asked to individually complete basic coding exercises using Scratch. At the end of the first day, the teams prepared and presented their storyboards with three different scenes on paper/cardboard, including the title, theme, character, plot, conflict, and solution.

Second day of the workshop: Starting the second day of the workshop, the girls had to update, if they wanted, their storyboards and finalize them. Then, the rest of the day was dedicated to their game creation using Scratch. The girls completed a paper-based tutorial, created by the instructors, with simple text explanations and examples of basic CT concepts and possible loops that the girls were supposed to use in their own games, all based on Scratch. During the creation of their games, the girls had to use their storyboards exactly and “transfer” their ideas into games using Scratch. At any time, the girls could ask for help from the instructors, who even introduced complicated programming concepts, if it was necessary for their games. The girls created their games step by step and continuously testing and coding them. At the end of the day, all teams prepared presentations of their games and everyone played each other's games.

Fig. 4. Girls participating in the workshop (left), creation of the storyboard (middle), and game created using Scratch (right).

3.3. Sampling

All the participants of the three cycles were students from Trondheim region. The first two cycles took place at NTNU in specially designed rooms, and the last cycle took place in the local library. The data related to the three cycles were collected after receiving permission from the Norwegian Centre for Research Data (NSD), following all the regulations and recommendations for research with children. When the participants had been selected, a researcher contacted their teachers and parents in order to obtain the necessary consent from both the child and the legal guardian for the data collection. Their participation in the research project was voluntary and they could drop out at any time, with no consequences on their participation in the workshop.

3.3.1. Participants of cycle 1

Children from 3rd to 12th grade (aged 8–17 years old) participated in the coding activity. The activity took place during autumn 2016 with a sample of 12 girls (mean age: 12.64, SD: 2.838) and 32 boys (mean age: 12.35, SD: 2.773). Five workshops took place over two weeks, following the coding activity described in the previous section.

3.3.2. Participants of cycle 2

In autumn 2017, children from 8th to 10th grade (aged 13–16 years old) participated in the coding activity. The sample consisted of 105 participants in total, 69 boys and 36 girls (mean age: 14.55, SD: 0.650). Kodeløypa workshops were conducted every Friday for six weeks.

3.3.3. Participants of cycle 3

The sample of the third study consisted of eight girls from 6th to 10th grade (aged 10–14 years old) (mean age: 12.135, SD: 1.389). Girls participated in the two-day workshop during autumn 2017, following all the activities of the workshop described in the previous section at the local library.

3.4. Data collection

Following the DBR methodology described by (Reeves, 2006), our study involved four stages (Table 1). In stage 1, we conducted a critical literature review to identify theoretical and practical problems in constructionism-based coding activities. Then, in the second stage, after discussions with instructors and with experts in human–computer interaction (HCI) and technology-enhanced learning (TEL), we developed the design of the intervention based on constructionism. Stage 3 involved the testing and refinement of the iterative cycles in practice. Qualitative and quantitative data were collected during the three cycles using various instruments, including pre and post knowledge acquisition tests, attitudinal questionnaires, eye-tracking data, semi-structured interviews, field notes from observations, instructors' reflections, and the artifacts constructed by the children in different phases of the process. All data focused on exploring the children's learning experience

Table 1
Description of the different DBR stages.

Stage	Data collection method	Participants	Purpose
Analysis	Literature review	Researchers HCI experts TEL experts Instructors	Analyze and identify problems and gaps in constructionism-based coding activities
Development	Literature review Discussions Focus groups	Researchers HCI experts TEL experts Instructors	Identify the theoretical framework Design the interventions
Iterative cycles of testing and refinement in practice	**Iteration 1** Eye tracking Attitudinal questionnaire Knowledge acquisition test (pre and post) Artifact collection Instructors' reflections **Iterations 2 and 3** Semi-structured interviews Field notes from observations Artifact collection Instructors' reflections	**Iteration 1** 44 children aged 8–17 years old Instructors **Iteration 2** 105 children aged 13–16 years old Instructors **Iteration 3** 8 girls aged 10–14 years old Instructors	Get a comprehensive view of students' learning experience Design elements for the next iteration
Development of design principles	Focus groups Discussions Reflections and notes from all cycles	Researchers HCI experts TEL experts Instructors	Identify the prominent design principles

in our coding workshops and guided us to the improvement of the design of the next iteration. The fourth stage of DBR is the development of design principles that intend to provide feasible solutions with respect to the theoretical goals. This final stage contains all the reflections from the previous stages, including notes of the design issues that emerged from the analysis of the results at each iteration.

3.5. Data analysis

In the DBR methodology, all stages, from the analysis to the development of design principles, include interactive and iterative formative evaluations. From the beginning of the cycles' implementation, starting with the design, to the execution and evaluation of each workshop, the researchers and instructors were in constant collaboration. Their involvement throughout the project allowed them to gain valuable knowledge and competence in the analysis and interpretation of the data gathered in each cycle. All data collected from the three cycles were respectively analyzed according to their type. For example, quantitative data were analyzed using one-way analysis of variance (ANOVA) and Pearson correlation coefficient among other; while qualitative data were analyzed based on Saldaña (2015). All data were compared and cross-checked for triangulation. For this paper, the qualitative analysis was manually conducted by the researchers using both inductive and deductive approaches, based on (Saldaña, 2015) (Mayring, 2014).

During the two years of the project, after the end of each iteration (cycle), the researchers and instructors participated in focus groups discussing and revealing all the growing ideas emerged from the outcomes of the iteration. All ideas were connected to the results of the respective iteration, representing the codes for our qualitative analysis for this study. In order to synthesize the ideas and formulate themes, we focused mainly on the students' engagement in the coding activities. The students' engagement included interaction with the instructor and the learning tool and interaction with other students in the creation of an artifact. In our approach, we adopt the term “academic engagement” (Turner, Christensen, Kackar-Cam, Trucano, & Fulmer, 2014) to describe how the students were involved in and put effort into learning, understanding, and collaborating with their peers. Engagement during educational activities has many aspects and is connected with other theoretical constructs, like motivation and self-regulation (Henrie, Halverson, & Graham, 2015). According to (Fredricks, Blumenfeld, & Paris, 2004), there are three types of engagement: behavioral, emotional, and cognitive, which are interrelated within the individual. “Behavioral engagement” refers to participation, involvement, and attention, among others. “Emotional engagement” refers to the learner's feelings, like frustration or interest, expressions of positive effects, and social connection. “Cognitive engagement” refers to the learner's investment in understanding what they have been taught, their efforts related to the mind, their strategy use, and their self-regulatory and meta-cognitive behaviors.

Each idea was connected with one of the three types of engagement, depending on its content. For example, ideas representing children's cognitive processes, like the use of different gaze patterns during the coding activity, were placed under cognitive engagement. Respectively, we followed the same procedure to place, if possible, all ideas under the appropriate type of engagement, which also allowed us to see possible interconnections. Consequently, the most prominent themes emerged. It was an iterative process, with constant refinement and reflection on the ideas and themes during the three cycles. This helped us not only to see the connections and make decisions for the design but also to identify the most important theoretical aspects in our studies. The final step of the analysis, after removing similar themes, involved categorization to identify the most important findings. The categories were interpreted according to Papert's (1980) theoretical framework, with the agreement of the instructors and the HCI and TEL experts (Fig. 5).

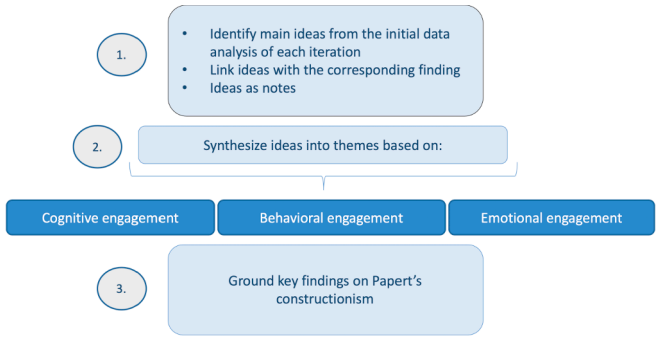

Fig. 5. Data analysis process.

In the next section, we present the findings for the first cycle, showing the important contributions based on the theoretical framework of Papert's constructionism. Then, for cycles 2 and 3, we first present the key findings emerging from the respective previous cycle related to the design of the activities and then the important contributions based on the theoretical framework.

4. Iterative design cycles, theoretical findings, and design elements

For each of the three cycles, we present the most prominent results as linked to Papert's constructionism. Therefore, there is no detailed representation of the results, as they were respectively analyzed according to their type during the process. However, when needed, there is a reference to the findings related to the data collection method in order to help the proper explanation of the specific outcome.

4.1. Cycle 1

Two theoretical ideas emerged from this cycle:

1) Learning to learn (different coding approaches result in different learning gain): According to Papert (Papert, 1980), in a constructionist learning environment, the child is able to construct their own knowledge and build on what they already know. In our workshop, the students produced socially meaningful and engaging artifacts: games. The findings from this study (cycle 1) showed that depending on their age, the students used different gaze patterns in the coding process, had different approaches to coding, and had different learning gain from the activity.

The younger students (kids) focused on the appearance of their games' characters, while the older ones (teens) had more-structured coding behavior. This was evident in the proportion of time that the kids and teens spent in specific areas of interest (based on eye tracking) in the Scratch programming environment and the transitions between them. The teens presented more "hypothesis-testing" behavior during their efforts in making the games and could shift their attention to the more-"meaningful" parts of the Scratch screen. By "the meaningful parts of the screen", we mean specific areas of interest in the Scratch interface that indicate the main areas of attention in coding: scripts, output, and commands. In addition, the teens were able to collaborate better than the kids were (had higher similarity gaze). The teens had a higher level of shared understanding and could communicate better during the coding activity. This confirms the teens' attitude towards helping each other more, contrasting with the kids, who wanted to have greater individual control. Eventually, "*by deliberately learning to imitate mechanical thinking, the learner becomes able to articulate what mechanical thinking is and what it is not. The exercise can lead to greater confidence about the ability to choose a cognitive style that suits the problem"* and *"what is most important in this is that through these experiences these children would be serving their apprenticeships as epistemologists, that is to say learning to think articulately about thinking"* (Papert, 1980). Children's coding processes represent their way of "thinking mechanically" and adopting the educational advantage of this way of deliberately thinking. Using a simple description of the process, trying to create/make a game is a way to combine appropriate orders and create programs to tell the computer what to do, step by step. This process includes logic, math, problem-solving, and critical thinking skills. In order for children to achieve their goals in such environments, they should find the appropriate cognitive style that will support them in the coding process of creating a shared artifact. This shows the importance of having appropriate tools and instructions for each age group. Different age groups differently organize their thinking and consequently their coding, so the way they approach the process of creating an artifact can be instrumental to their learning and the successful completion of the artifact. This notion is in accordance with Papert, as he presents a resemblance with juggling: *"It always takes time to learn necessary component skills. What can be eliminated are wasteful and inefficient methods. Learning enough juggling skill to keep three balls going takes many hours when the learner follows a poor learning strategy. When a good one is adopted the time is greatly reduced, often to as little as twenty or thirty minutes"* (Papert, 1980). Finding the appropriate methods to help children of different age groups will result in efficient and effective learning processes.

2) Cognitive effort and affective engagement: Positive attitudes and motivation are important to cognitive learning. There is a relation between children's attitudes and their cognitive processes while coding. Highly motivated children with positive attitudes have the ability to handle cognitive load and better manage the construction of their artifacts. This idea appeared in our findings from the measures used to examine cognition through the eye-tracking data and the relation with attitudes of perceived learning (seen as confidence, the degree that children indicate their performance), intention to perform coding again, and excitement. The children who were highly engaged and motivated during the construction of the artifact exhibited gaze behavior that showed lower cognitive overload. Papert (1980) describes the notion of "bricolage", which represents a qualitative way of organizing and planning when problem solving by constantly experimenting until finalizing the artifact. Effort and difficulty are prominent during the whole coding process and require motivational goals and determination from a child to commit themselves to the learning. This is an expected notion, as *"You can't learn bread-and-butter (basic) skills if you come to them with fear and the anticipation of hating them"* (Papert, 1980). The design of the coding activity of our workshop had an overall cognitive load that could become overwhelming for children, especially those who are novices to coding. From the complexity of the task, children might reach a point of feeling overloaded, which can lead to a critical condition where, without the proper pleasant and motivating environment, the learning experience can fail. It is not a surprising result that the children with more difficulties and cognitive load had lower scores in their attitudes.

4.2. Cycle 2

The key findings, as design elements, that emerged from cycle 1 and guided the refinement of the design of cycle 2 are described below.

Structured assistance, pleasant environment, and revised learning materials to:

a Guide students to focus on structured coding behavior.

Students should put a lot of effort and thinking into learning the necessary component skills, and they should be cognitively supported during the coding activity. As shown in the results of the eye-tracking data, those who shifted their attention to the meaningful parts of the screen (such as commands and output) had better learning gain, based on their knowledge acquisition tests. Therefore, the design of the activity should support students efficiently to ensure that they can take appropriate actions and know where to pay attention when they code to have an effective approach that is suitable for the task.

b Avoid cognitive overload.

Students can become easily overwhelmed in the process of creating an artifact, especially when they are new to coding. By using the "bricolage" style, in which they are constantly experimenting, students can feel overloaded as they seek to find the appropriate commands in the tool, manage different tasks, and make decisions during the activity. Consequently, supporting students when needed and providing relevant learning materials can reduce their cognitive load and provide a scaffold for managing their learning and thinking.

c Keep the participants motivated.

Students' positive attitudes are related to their cognitive load, as represented by their eye movements, based on the results from cycle 1. Highly motivated students with positive attitudes have better management of cognitive load. Hence, providing a pleasant environment that enhances students' enthusiasm for and engagement with learning will help students to have a fruitful experience.

d Enhance collaboration within the teams.

As students collaborate in teams to create a shared artifact, social interaction in learning during the coding activity is not something we can overlook, as it also unfolds team dynamics. Teams with better collaboration (higher gaze similarity) had higher team average learning gain, as calculated by the knowledge acquisition tests. It is important to encourage collaboration and good communication among team members so that they can benefit from each other's help.

In this cycle, the duration of the workshop for all groups of students was the same, as an out-of-school one-day activity. The results were based on the qualitative analysis of the interviews, observations, and evaluations of the students' artifacts. The children were able to express exactly their struggles and ways of thinking during the artifact creation, allowing us to detect the exact behavior of the children as they expressed it to reflect their cognitive processes (noticing debugging behavior and specific difficulties). Triangulation of the data helped us to validate our findings. The implementation of the Kodeløypa coding workshop took place over two years, with few differences in the design of the activity but with differences in the research design, evidence descriptions, and results of the different instruments used for data collection.

Two theoretical ideas emerged from this cycle:

1) Social aspect of creating an artifact: The "social" dimension refers to the role of collaboration in the coding activity. Children worked in teams of three (or two, depending on the total number of participants) to create a shared artifact. Collaboration and social interaction for a common goal have many benefits, including interacting with others, examining different perspectives, expressing understandings, and interpreting things differently. During the coding activity, the children were encouraged to work collaboratively to create a shared artifact that was meaningful for their peers too. The process also offered the opportunity to all participants to play each other's games and reflect on them. Collaboration was primarily examined between the members of the groups but also among the different teams. From the observations and interviews, the help they got from other team members was important. Half of the children expressed the highest level of satisfaction with the collaborative process in their team, while 72% showed high levels regarding being able to develop skills from the other members of the team. This interaction, which shows collaboration and help among the teams, had various aspects, from practical (what command they should use in Scratch to accomplish a task) to ideas for their games. This finding was confirmed from the artifact analysis: teams who were sitting close to one another had similarities in the programming concepts they used, as well as in their main game ideas (such as a maze or jumping on platforms). In addition, through the different versions of the artifacts, we observed that elements changed based on other teams' suggestions.

"When we didn't find anything, we looked at another group and saw how they did it"

For the team members, the construction of the artifact was not an individual task: it was a social interaction with a shared goal to create a game. The results showed that, most of the time, collaboration was efficient. The children acknowledged and expressed how valuable it was that they were working together to complete their artifacts.

"If I had my own project, I would probably not find anything"

"It is easier to code with someone than to code by yourself; if I had been alone, I wouldn't have managed to do the same"

"We both came up with ideas and equally contributed to the design and coding parts"

"I coded more, while they contributed with ideas on what should be incorporated. We were all important members of the team"

An important aspect of the good collaboration was the fact that the team members knew each other from before and/or had done other projects together.

"We knew each other, and we felt pretty safe around each other. We could discuss and agree easily on what had to be done"

Nevertheless, there were some indications of bad collaboration that caused frustration. This was mainly caused from having a "bad leader" in the group who wanted control. This was expressed from both sides.

"It was maybe that I took too much control. I should have let my partner decide a bit more"

"He didn't let me finish my task; he just wanted to have the control back"

Papert's (Papert, 1980) notion of the importance of social norms and interaction in learning is reflected in his research on samba schools: *"These are not schools as we know them; they are social clubs with memberships that may range from a few hundred to many thousands"*. The construction of games and other artifacts is not an isolated action but happens in a social context.

This resonates with Papert's (Papert, 1980) notion of social interaction: *"Although the work at the computer is usually private it increases the children's desire for interaction. These children want to get together with others engaged in similar activities because they have a lot to talk about. And what they have to say to one another is not limited to talking about their products: Logo is designed to make it easy to tell about the process of making them"*.

2) Powerful thinking (or learning about thinking): Papert (Papert, 1980) argues that children are able to recognize the different procedures in code, understand when the code does not run as expected, use debugging strategies, and act intentionally to improve the code. For the construction of their artifacts during the coding activity, the children worked with programming concepts and practices to successfully complete their task. Making a game requires deep engagement and strategy use to successfully manage the completion of the task. The children iteratively organized and documented their code. As described by Papert (1980, p. 28) regarding the Logo environment: *"teaching the Turtle to act or to 'think' can lead one to reflect on one's own actions and thinking. And as children move on, they program the computer to make more complex decisions and find themselves engaged in reflecting on more complex aspects of their own thinking."*

For the construction of the artifacts, the children had the opportunity to plan, problem solve, code, debug, collaborate, communicate, and reflect on their coding experience using Scratch. The participants realized that this was an iterative process, and for some it appeared to be difficult and challenging. Some found it fun to try out the different blocks, discovering the different functionalities. Whatever they made seemed to be suitable for their code; at the same time, the need to add a new function changed everything and triggered a new thinking and debugging process.

"The hardest thing was to put the different pieces of code together and make them work as one game"

"It was very challenging when we started to change different things to see what happened with the other code"

The most prominent difficulties related to movement, jumping, the

use of loops, and hiding/showing different sprites. These actions were the main problems that the children had to deal with from the beginning of their game creation and defined their thinking processes. This was also indicated by the artifact analysis of the first versions of their games. In order to make a character move and jump in Scratch, you often have to have an event block with a conditional combined with motion blocks for moving the sprite x steps or to place it in a certain y- or x-coordinate in a chosen direction. Observations showed that movement and jumping were the most common reasons the children asked for help, indicating that it was hard for them to articulate their knowledge about conditionals (if _ then; repeat until; and when key is pressed, do this), direction, and the coordinate system to achieve an appropriate order of blocks.

Coding in Scratch enables children to articulate their thoughts and watch the outcomes of their own decisions.

"If you did something, the result wasn't always what you expected"

After the initial trials with coding, by being more and more engaged in the process, the children had the opportunity to clarify their thinking and interpret the immediate feedback, acting accordingly.

"Before, I didn't understand that things wouldn't happen if you didn't explicitly give instructions"

"The ideas and code come really fast when you realize what kind of options you have"

4.3. Cycle 3

The key findings, as design elements, that emerged from cycle 2 and guided the refinement of the design of cycle 3 are described below:

a Allow an adequate amount of time for engagement during the workshop.

The analysis of the interview data revealed that the time the students had to complete the tasks was an important issue for them, as the allocated time was limited. More precisely, at the beginning of the activity, they spent a lot of time trying to familiarize themselves with the tool and the tasks and to bond with team members, especially in teams where they did not know each other from before. Giving additional time for social engagement between the team members will allow students to build common understanding and be more creative.

b Provide a specific theme for the game creation.

As mentioned earlier, the students spent a lot of time at the beginning of the workshop. One of the time-consuming actions was to decide the theme of the game. Time management is very important in such workshops: on the one hand, students need to have the freedom to decide their own themes; on the other hand, it is critical to have an adequate amount of time for the follow-up tasks. Therefore, having a specific theme for the game creation that is sufficiently broad, inspiring, and meaningful will give them the freedom to be creative but at the same time will prevent them from "getting lost." In addition, it will give a meaningful social and personal context to the learning process, foster their interest, and create a common ground for all teams.

c Inspire the participants with an example of a female game hero and a demonstration of a similar game by female assistants (as role models).

From cycle 2, focusing on the analysis of the data collected from the teams consisting only of girls, it is evident that stereotypes exist. Most of them expressed that they had not tried coding before and did not know what to create, as they thought game creation was only for "geeks." In their eyes, only boys like video games. To encourage interest and get the girls inspired and engaged, a storyboard and a game were used as examples, with the main character a heroine who had powers that could "solve problems".

d Focus on the design part of the game in a structured way (i.e. spend sufficient time on creating the storyboard first and having a presentation on it).

The results from the data from cycle 2 (interviews, observations, and game versions) showed that the teams who followed a more-structured approach (creating a draft storyboard with their idea before starting coding) were able to successfully manage and finish on time, as well as being less overwhelmed. Moreover, based on the different versions of the collected games, these students had a greater capacity to develop their initial ideas (designed in the storyboards), and this resulted in higher-quality games (more complete/advanced).

e Introduce coding individually.

The students participating in the workshop did not have the same experience with coding. This approach was geared towards helping the participants individually to familiarize themselves with the tool (in our case, Scratch), gain insights on what they could create, and develop basic skills. Having a common ground of basic knowledge among the team members will make everyone engaged and active. Thus, it is very important to have some individual activities at the beginning that prevent students with experience from dominating their teams, which could disengage novices.

One theoretical idea emerged from this cycle:

1) Use of powerful ideas: "Powerful ideas", as described by Papert (Papert, 1980), are central concepts of learning and should be a necessary part of constructionist activities. A "powerful idea" must be both personally and epistemologically useful, giving the opportunity to organize a way of thinking, appropriate each time for the specific task, building on previously gained skills and knowledge. Learners need to be highly explorative before they gain expertise; therefore, the task they are required to do needs to be engaging enough in order to commit them to the learning process. In his book *Mindstorms*, Papert shows the importance of powerfulness and the powerful nature of children's use of computers as tools and the Logo programming language, as well all the powerful ideas that emerge from children's engagement with computer-based activities.

What is important is to make a powerful idea part of intuitive thinking (Papert, 1980). In the design of the activity in the third cycle, "powerful" was a quality gained from the girls, as they were allowed to closely engage with the creation of the artifacts in multiple stages, using Scratch. This process brought the learners in touch with some powerful general ideas, for example planning an exciting project, using programming instructions, debugging, and designing, to mention a few.

The girls had an experience outside of the classroom in a local library, collaborating with girls of a similar age but with varied interests and background knowledge, which was in contrast with a single classroom experience. The duration of the workshop was critical not only for learning purposes but also because it allowed the participants to bond and exchange interests and gave them the proper amount of time to interact, negotiate, learn from each other, and finally achieve the goal of the creation of the artifact. In addition, by having a concrete context for the game (create a game that reflects an environmental issue) and a tool (Scratch) embedded in a meaningful environment, they could see the project's relevance to their lives.

"It was so fun and exciting to make a game for saving the world with Scratch and with new friends, who taught me so much about computers"

The girls gradually discovered the Scratch tool and how they could use it. As they became more engaged in the process and saw their

artifact become a reality, they enhanced their feelings of self-achievement and self-confidence. They found themselves confronting difficulties and learning things that they did not know about game design. The use of Scratch gave them new possibilities and made them "walk it through" and relate their personal knowledge to thinking effectively and happily to achieve the artifact construction.

"I thought it was much harder to make a game, but I could understand how to use it and at the end we managed to do everything we wanted"

"… some things were difficult, but we tried and made things happen"

"… we find out how things worked, and many times we had to go back and change stuff"

"I am so proud of what I did today … When you design a game in a storyboard, you don't think about using a timer, but with Scratch you can … you can do everything you can think of"

5. Discussion

The intended outcomes of this DBR were twofold: 1) to ground the main findings of interventions conducted over two years in constructionism, and 2) to identify reusable design principles that can inform coding activities for children and pedagogical tasks. In this study, we aimed to investigate children's learning experience as they constructed their own knowledge by using a digital programming tool (Scratch) and collaboratively creating socially meaningful artifacts: games.

Analysis of the different data collected from the various instruments over the two-year intervention helped us to explore the effectiveness of our coding workshops on children's engagement. We focused on how they enhanced participants' knowledge of basic programming concepts, their coding behavior, their social interaction and collaboration, and how they perceived their coding experience as a whole when introduced to coding.

It is important to have appropriate educational designs aiming to promote active learning with the support of constructionism. Including components like a balance of individual and social involvement and the use of a visual programming language, all employed under the common goal of creating an artifact, fosters children's deeper transferable CT skills, which are vital for our society's information revolution. Engaging children in a learning environment that embraces creative design, problem solving, collaboration, and communication strengthens their sense of competence and confidence, their compassion for others, and their moral character (Bers, 2010). Together with achieving a significant improvement in students' understanding of computational knowledge, like programming concepts and practices, it is essential to create high levels of motivation, fun, and commitment as part of an efficient pedagogical design, as reflected in our study.

5.1. Engagement in constructionism-based coding activities

Below we summarize the main characteristics of student engagement, as shown in our DBR approach and according to constructionism.

The students indicated that they were cognitively engaged during the workshops; they managed to adopt deliberative thinking and to understand and imitate mechanical thinking while coding. In order to achieve this, they had to use an appropriate cognitive strategy (e.g., a "hypothesis-testing" gaze pattern, as shown by the eye-tracking data) to approach the task and achieve some level of self-regulation (Papavlasopoulou, Sharma et al., 2017). There are different ways to approach a problem, and it takes time to learn the necessary skills. In our workshops, we used a visual programming tool (Scratch); one of the strengths of such tools is that computational practices become less cognitively challenging (Kelleher & Pausch, 2005), so students can focus on problem solving and creative thinking (Lin & Liu, 2012). Even with the use of such tools, during the coding process, cognitive load can be critical, as students use the "bricolage" style by constantly experimenting and trying different patterns. Instructors can help students to manage their learning and thinking and to adopt an effective approach to coding. This is not a new practice, as previous studies with Logo have used precise instructions for computational practices such as testing and debugging (Fay & Mayer, 1994) (Carver & Mayer, 1988).

Cognitive effort, as shown in our study, is also linked with students' behavioral and emotional engagement because positive attitudes have an effect on their load management. Students should be persistent, put effort in, and deal with difficulties; therefore, having positive attitudes and keeping themselves motivated result in better management of their cognitive load (Papavlasopoulou, Sharma, & Giannakos, 2018). In the same vein, Robertson and Howells (2008) argue that the game design experience is a powerful learning environment that supports motivation, engagement, and enthusiasm. Using a visual programming environment, students can be introduced to programming concepts in a fun and useful way through a design activity, making them highly motivated and positive towards coding (Giannakos & Jaccheri, 2018) (Sáez-López et al., 2016).

Social engagement is important as students work in front of the computer and reflect on their progress as a team, sharing the same goal to successfully create an artifact. Working as a team, in our workshops, the students built a group identity and at the same time engaged in social comparison with their peers. Students, especially novices to coding, usually have difficulties with simple coding actions, from relating different commands together to completing more-advanced actions, like debugging; collaboration helped the students in this study to confront those difficulties. In a similar study with girls creating games, good collaboration in debugging resulted in the girls being more persistent when coding on their own, without help from the instructors (Denner, 2007). In the present study, helping each other and sharing their challenges and successes were critical for our students, nurturing social engagement and avoiding a sense of isolation. Collaboration and reflection lead to better learning and powerful thinking. Reflection relates to their own learning experience or reflecting on their peer's code and actions. Previous studies have shown that students performed better when they were working in pair programming (Lye & Koh, 2014) (Werner, Denner, Campe, & Kawamoto, 2012); in a game-making study, when taking into account peers' recommendations and spending time applying these changes, girls produced higher-quality games (Robertson, 2012). Over time, the students in our workshops were able to understand more about coding and became more behaviorally and emotionally engaged. They were able to reflect on the more-complex aspects of their own thinking accordingly by making decisions and controlling the outcomes. Students who are actively part of game-making activities strengthen their problem-solving, critical thinking, and CT skills (Grover & Pea, 2013). During construction, students have to investigate different strategies, negotiate and make decisions about possible solutions, confront problems, and organize their thoughts and actions (Bers, Flannery, Kazakoff, & Sullivan, 2014).

One of the core aspects of a learning activity is the fact that the problem should be meaningful to the learners. In our case, they constructed shared artifacts that mattered to them. Different studies have used problems like designing games (Denner & Werner, 2007) or stories (Burke, 2012). A "powerful idea" must be both personally and epistemologically useful to ensure engagement. The students in our workshops saw themselves gaining a powerful quality by organizing a new way of thinking, building on their previous knowledge and skills. Nowadays, significant value is placed on transferable skills related to digital technology, as they are vital for children's role in the digital world and should be enhanced through activities that are connected to their lives (Iversen, Smith, & Dindler, 2018). In constructionist learning, students deal with difficulties, learn step by step to solve problems, develop belief in their skills, and share ideas with peers (Çakır et al., 2017) (Chu et al., 2017). In our study, this was confirmed: the students

increased their sense of achievement, self-confidence, and self-efficacy. At the end of the workshops, the students felt competent and proud of their achievements. After the workshop, compared to the boys, the girls expressed lower self-efficacy (a belief in one's capacity to succeed in tasks), possibly due to the fact that most of them did not have any previous experience with coding. A sense of self-efficacy is important and should be enhanced, as it is related to cognitive strategies, effort, and persistence in learning environments (Bandura, 1997).

5.2. Principles to facilitate constructionism-based learning environments that support children's learning experience

In summary, we identified the following nine principles emerging from our DBR study, which shed light on best practices in the design of coding activities for children based on constructionism. The principles emerged represent the knowledge gained from the two years of interventions and the comparative and retrospective analysis of the outcomes based also on the literature:

1) *Social interaction:* Collaboration between team members is a vital part of coding activities. It is essential to enhance this and to ensure that there is a sense of equality of effort, involvement, and participation between team members and among teams.
2) *Appropriate design according to age*: Different age groups (teens and kids) need different approaches and designs in order to engage with a coding activity. The instruction should consider the characteristics of each age group. One example is to promote a focus on functionality rather than graphics from the beginning of the activity to aid younger participants. Instructors should ensure that children receive guidelines on where to focus their attention when they code (such as commands and output in Scratch).
3) *Duration of the activity:* According to constructionism (Papert, 1980), when having children use technological tools, duration is key for them to become personally, intellectually, and emotionally involved. Workshops with longer hours can enable children to learn strategies, gain technological skills, make connections with their own practices, and engage with coding, helping to increase their knowledge.
4) *Relevance of the activity and meaningful content:* Offering a supportive theme for the artifact creation process, in which participants can meaningfully participate in real-life settings, is a key factor supporting the psychological and sociocultural elements for effective learning. Children become engaged and actively involved in the process of artifact creation when it is meaningful for them and related to a real-life context.
5) *Physical and digital artifacts:* The results of the present study showed that the inclusion of physical tasks was engaging and enabled the participants to enhance their skills. The initial task of designing and drawing in the traditional way (using pen and paper, as well as other tangible materials) immediately put players into action and created a physical and emotional peak in the process.
6) *Children's attitudes and motivation:* The learning process should be supported by providing tasks that encourage children to reflect, motivate them to collaborate, and give them meaningful reasons to complete their artifacts. In this vein, Papert (1980, p. 42) highlighted a resemblance with juggling: *"in a learning environment with the proper emotional and intellectual support, the 'uncoordinated' can learn circus arts like juggling and those with 'no head for figures' learn not only that they can do mathematics but that they can enjoy it as well."*
7) *Cognitive overload:* Coding activities for children can have a high cognitive load, which affects their performance and overall experience with the tasks. Proper organization and integration of the learning materials, with a coherent representation and instruction of the related digital tools, tasks, and activities, are required to avoid unnecessary streams of information and cognitive overload.
8) *Appropriate tasks:* To effectively implement a coding workshop, the tasks should make the children both interested and able to learn. The process should afford participants the opportunity to apply aspects of problem solving, coding, debugging, collaborating, planning, communicating, and reflecting on their work. The tasks should support children's and instructors' ability to work through the process of creating an artifact and benefit from an appropriate sequence of tasks that allows the maximum use of their abilities. The proposed tasks are: 1) a warm-up activity and an inspiring introduction, 2) explore/design, 3) construct/create the digital artifact, and 4) evaluate/get feedback from peers, all alongside collaborating with team members and receiving support from assistants/instructors.
9) *Meaningful framework for the involvement of the instructors:* In the construction of an artifact, children are not alone: practitioners (e.g., teachers and assistants) and anyone else who is responsible for the learning task are also involved. Therefore, they should strive to create more-articulate and -honest teaching relationships. Working with digital tools allows the teacher and the learner to share a common goal by trying to get the computer to do what they want and trying to understand what it does. As they create the artifact and encounter "bugs", children engage in conversations and develop the appropriate language to ask for help when they need it. As each artifact process is unique, new situations might occur that neither the teacher nor the learner has faced before. So, the teacher should be dynamically involved in the creation and the discussions that occur. In that way, there is an opportunity to find new ways to explain and show in real time the concepts needed to the children. As noted by Papert (Papert, 1980): *"sharing the problem and the experience of solving it allows a child to learn from an adult not 'by doing what teacher says' but 'by doing what teacher does.'"*

6. Limitations

This study had some limitations. First, our workshops were designed for children who had no previous experience of coding. The participants were randomly selected; therefore, the sample was not consistent in terms of the children's prior knowledge and interaction with coding. Even though we had an indication in our data collection to measure the children's previous knowledge, we could have used other methods to be more accurate. Second, the factors that might affect children's self-perceptions are much more complex than we might assume. Third, although the participants of the third cycle were committed for the two days' workshop and gave us high quality data, the sample is not large; this is due to unexpected matters from the participants' side prior to the scheduled dates of the workshop. In addition, the age range of the students in the study is big (8–17), maybe, focusing on a smaller range would have given a different perspective. Demographic variables and other characteristics (cognitive and motivational) that distinguish them from the rest of the population could have confounded the findings. Artifacts like games might be imperfect examples of what children learn, especially when they receive help during the process. Despite the fact that we observed the teams and made notes on the help they received, we might have underestimated or overestimated their understanding of programming concepts.

In addition, limitations due to the types of data collection methods and instruments used apply in our case. One limitation related to the eye tracking: the young age of the participants, their enthusiasm during the activities, and the fact that eye trackers are designed for adults made it difficult to gather good-quality data. Moreover, this project used Scratch as a programming environment for the development of the artifact: another technological tool might have had a different impact on the children's experience. Our choice was based on our literature review and the acknowledged benefits of this programming environment, which has been widely used over the last few years. Although we tried to apply all aspects of the DBR methodology in our study, showing the relationship between theory and practice (of the artifact construction activity), there were still some limitations. The data were extensive

and comprehensive, requiring extended time for collection and analysis; consequently, because time and resources were limited, some data might have been discarded or received less attention. Lastly, we defined in detail the setting of our study and how theory was linked with the context; by default, this has a bias, as it presents our own understanding of contextualizing the theory.

7. Future work

Future research should further explore gender differences. Although the main focus of our study was not to investigate gender differences in the process of creating an artifact, we found that girls like to make different type games from boys, in terms of both content (story/purpose of the game) and elements (colors and main character), and tend to handle the process slightly differently. In addition, future plans should include conducting our coding workshops in school settings to explore their effects under a traditional teaching approach. Among other aspects, researchers could explore the correlations with students' performance in the form of grades. Finally, in terms of theory, it would be interesting to see more studies in the area that ground their findings in constructionism. This would bring together researchers in the same area to build a common ground regarding outcomes.

Acknowledgements

The authors would like to express their gratitude to all of the children, teachers and parents for volunteering their time. Our very special thanks go to Kshitij Sharma, Uyen Dan Nguyen, Kristin Susanne Karlsen, Ioannis Leftheriotis, Amanda Jørgine Haug, Lidia Luque Fernandez, An Nguyen, Ton Mangmee, Marjeris Sofia Romero, Eline Stenwig and Kristoffer Venæs Monsen. The project has been recommended by the Data Protection Official for Research, Norwegian Centre for Research Data (NSD), following all the regulations and recommendations for research with children.

This work supported from the "Learning science the fun and creative way: coding, making and play as vehicles for informal science learning, in the 21st century" Project, under the European Commission's Horizon 2020 SwafS-11-2017 Program (Project Number: 787476). This article reflects the views only of the authors and it does not represent the opinion of neither the European Commission nor NTNU, and the European Commission and NTNU can not be held responsible for any use that might be made of its content. This work is also supported from the Norwegian Research Council under the project FUTURE LEARNING (number: 255129/H20), and by NOKUT under the Centre for Excellent IT Education (Excited) (number: 16/02049).

References

Anderson, T. (2005). Design-based research and its application to a call centre innovation in distance education. *Canadian Journal of Learning and Technology/La revue canadienne de l'apprentissage et de la technologie, 31*(2).

Anderson, T., & Shattuck, J. (2012). Design-based research: A decade of progress in education research? *Educational Researcher, 41*(1), 16–25.

Bandura, A. (1997). *Self-efficacy: The exercise of control.* Macmillan.

Barab, S., & Squire, K. (2004). Design-based research: Putting a stake in the ground. *The Journal of the Learning Sciences, 13*(1), 1–14.

Bers, M. U. (2010). Beyond computer literacy: Supporting youth's positive development through technology. *New Directions for Student Leadership, 2010*(128), 13–23.

Bers, M. U. (2012). *Designing digital experiences for positive youth development: From playpen to playground.* Oxford University Press.

Bers, M. U., Flannery, L., Kazakoff, E. R., & Sullivan, A. (2014). Computational thinking and tinkering: Exploration of an early childhood robotics curriculum. *Computers & Education, 72*, 145–157.

Bonner, D., & Dorneich, M. (2016). Developing game-based learning requirements to increase female middle school students interest in computer science. *Paper presented at the Proceedings of the human factors and ergonomics society annual meeting*.

Buechley, L., Eisenberg, M., Catchen, J., & Crockett, A. (2008). The LilyPad Arduino: Using computational textiles to investigate engagement, aesthetics, and diversity in computer science education. *Paper presented at the Proceedings of the SIGCHI conference on Human factors in computing systems*.

Burke, Q. (2012). The markings of a new pencil: Introducing programming-as-writing in the middle school classroom. *Journal of Media Literacy Education, 4*(2), 121–135.

Çakır, N. A., Gass, A., Foster, A., & Lee, F. J. (2017). Development of a game-design workshop to promote young girls' interest towards computing through identity exploration. *Computers & Education, 108*, 115–130.

Carver, S., & Mayer, R. (1988). Learning and transfer of debugging skills: Applying task analysis to curriculum design and assessment. In R. E. Mayer (Ed.). *Teaching and learning computer programming: Multiple research perspectives* (pp. 259–297). Hillsdale, NJ: Lawrence Erlbaum Associates.

Chu, S. L., Schlegel, R., Quek, F., Christy, A., & Chen, K. (2017). 'I make, therefore I Am': The effects of curriculum-aligned making on children's self-identity. *Paper presented at the proceedings of the 2017 CHI conference on human factors in computing systems*.

Collective, D.-B. R. (2003). Design-based research: An emerging paradigm for educational inquiry. *Educational Researcher, 32*(1), 5–8.

Collins, A. (1992). Toward a design science of education. *New directions in educational technology* (pp. 15–22). Springer.

Denner, J. (2007). The Girls Creating Games Program: An innovative approach to integrating technology into middle school. *Meridian: A Middle School Computer Technologies Journal, 1*(10).

Denner, J., & Werner, L. (2007). Computer programming in middle school: How pairs respond to challenges. *Journal of Educational Computing Research, 37*(2), 131–150.

Denner, J., Werner, L., & Ortiz, E. (2012). Computer games created by middle school girls: Can they be used to measure understanding of computer science concepts? *Computers & Education, 58*(1), 240–249. https://doi.org/10.1016/j.compedu.2011.08.006.

Eordanidis, S., Gee, E., & Carmichael, G. (2017). The effectiveness of pairing analog and digital games to teach computer science principles to female youth. *Journal of Computing Sciences in Colleges, 32*(3), 12–19.

Esper, S., Foster, S. R., Griswold, W. G., Herrera, C., & Snyder, W. (2014). CodeSpells: Bridging educational language features with industry-standard languages. *Paper presented at the proceedings of the 14th Koli calling international conference on computing education research*.

Fay, A. L., & Mayer, R. E. (1994). Benefits of teaching design skills before teaching logo computer programming: Evidence for syntax-independent learning. *Journal of Educational Computing Research, 11*(3), 187–210.

Fessakis, G., Gouli, E., & Mavroudi, E. (2013). Problem solving by 5–6 years old kindergarten children in a computer programming environment: A case study. *Computers & Education, 63*, 87–97. https://doi.org/10.1016/j.compedu.2012.11.016.

Fields, D., Vasudevan, V., & Kafai, Y. B. (2015). The programmers' collective: Fostering participatory culture by making music videos in a high school Scratch coding workshop. *Interactive Learning Environments, 23*(5), 613–633.

Fredricks, J. A., Blumenfeld, P. C., & Paris, A. H. (2004). School engagement: Potential of the concept, state of the evidence. *Review of Educational Research, 74*(1), 59–109.

Gallup, G. a. (2015). Searching for computer science: Access and barriers. *U.S. K–12 education*.

Giannakos, M. N., & Jaccheri, L. (2018). From players to makers: An empirical examination of factors that affect creative game development. *International Journal of Child-Computer Interaction, 18*, 27–36.

Granic, I., Lobel, A., & Engels, R. C. (2014). The benefits of playing video games. *American Psychologist, 69*(1), 66.

Grover, S., & Pea, R. (2013). Computational thinking in K–12 a review of the state of the field. *Educational Researcher, 42*(1), 38–43. https://doi.org/10.3102/0013189X12463051.

Grover, S., Pea, R., & Cooper, S. (2015). Designing for deeper learning in a blended computer science course for middle school students. *Computer Science Education, 25*(2), 199–237.

Guzdial, M. (2004). Programming environments for novices. *Computer Science Education Research, 2004*, 127–154.

Henrie, C. R., Halverson, L. R., & Graham, C. R. (2015). Measuring student engagement in technology-mediated learning: A review. *Computers & Education, 90*, 36–53.

Horizon, M. (October 5, 2015). *Horizon Media study reveals Americans prioritize STEM subjects over the arts; science is "cool," coding is new literacy.* PR Newswire. Retrieved from http://www.prnewswire.com/news-releases/horizon-mediastudy-reveals-americans-prioritize-stem-subjects-over-the-arts-science-is-cool-coding-is-new-literacy-300154137.html.

Hubwieser, P., Armoni, M., Giannakos, M. N., & Mittermeir, R. T. (2014). Perspectives and visions of computer science education in primary and secondary (K-12) Schools. *ACM Transactions on Computing Education (TOCE), 14*(2), 7.

Hubwieser, P., Giannakos, M. N., Berges, M., Brinda, T., Diethelm, I., Magenheim, J., & Jasute, E. (2015). A global snapshot of computer science education in K-12 schools. *Paper presented at the proceedings of the 2015 ITiCSE on working group reports*.

Iversen, O. S., Smith, R. C., & Dindler, C. (2018). From computational thinking to computational empowerment: A 21st century PD Agenda. *Paper presented at the participatory design conference*.

Jones, S. P. (2013). *Computing at school in the UK.* http://research.microsoft.com/en- us/um/people/simonpj/papers/cas/computingatschoolcacm.pdf.

Kafai, Y. B. (2006). Playing and making games for learning: instructionist and constructionist perspectives for game studies. *Games and Culture, 1*(1), 36–40.

Kafai, Y. B., & Burke, Q. (2015). Constructionist gaming: Understanding the benefits of making games for learning. *Educational Psychologist, 50*(4), 313–334. https://doi.org/10.1080/00461520.2015.1124022.

Kafai, Y. B., & Kafai, Y. B. (1995). *Minds in play: Computer game design as a context for children's learning.* Routledge.

Kafai, Y. B., & Resnick, M. (2012). Introduction. *Constructionism in practice* (pp. 13–20). Routledge.

Kafai, Y. B., & Vasudevan, V. (2015). Constructionist gaming beyond the screen: Middle school students' crafting and computing of touch pads, board games, and controllers. *Paper presented at the proceedings of the workshop in primary and secondary computing*

education.
Kalelioğlu, F. (2015). A new way of teaching programming skills to K-12 students: Code.org. *Computers in Human Behavior, 52*, 200–210.
Kao, D., & Harrell, D. F. (2017). MazeStar: A platform for studying virtual identity and computer science education. *Paper presented at the proceedings of the 12th international conference on the foundations of digital games*.
Kelleher, C., & Pausch, R. (2005). Lowering the barriers to programming: A taxonomy of programming environments and languages for novice programmers. *ACM Computing Surveys (CSUR), 37*(2), 83–137.
Lin, J. M.-C., & Liu, S.-F. (2012). An investigation into parent-child collaboration in learning computer programming. *Journal of Educational Technology & Society, 15*(1).
Lye, S. Y., & Koh, J. H. L. (2014). Review on teaching and learning of computational thinking through programming: What is next for K-12? *Computers in Human Behavior, 41*, 51–61.
Mayring, P. (2014). *Qualitative content analysis: Theoretical foundation, basic procedures and software solution*.
Papavlasopoulou, S., Giannakos, M. N., & Jaccheri, L. (2017a). Empirical studies on the maker movement, a promising approach to learning: A literature review. *Entertainment Computing, 18*, 57–78. https://doi.org/10.1016/j.entcom.2016.09.002.
Papavlasopoulou, S., Giannakos, M. N., & Jaccheri, L. (2017b). Reviewing the affordances of tangible programming languages: Implications for design and practice. *Paper presented at the global engineering education conference (EDUCON), 2017*. IEEE.
Papavlasopoulou, S., Sharma, K., Giannakos, M., & Jaccheri, L. (2017). Using eye-tracking to unveil differences between kids and teens in coding activities. *Paper presented at the proceedings of the 2017 conference on interaction design and children*.
Papavlasopoulou, S., Sharma, K., & Giannakos, M. N. (2018). How do you feel about learning to code? investigating the effect of children's attitudes towards coding using eye-tracking. *International Journal of Child-Computer Interaction, 17*, 50–60.
Papert, S. (1980). *Mindstorms: Children, computers, and powerful ideas*. Basic Books, Inc.
Papert, S. (1993). *The children's machine: Rethinking school in the age of the computer*. ERIC.
Papert, S. (1997). *Why school reform is impossible (with commentary on O'Shea's and Koschmann's reviews of" the children's machine*. JSTOR.
Parmaxi, A., & Zaphiris, P. (2015). Developing a framework for social technologies in learning via design-based research. *Educational Media International, 52*(1), 33–46.
Parmaxi, A., Zaphiris, P., & Ioannou, A. (2016). Enacting artifact-based activities for social technologies in language learning using a design-based research approach. *Computers in Human Behavior, 63*, 556–567.
Reeves, T. (2006). Design research from a technology perspective. *Educational design research* (pp. 64–78). Routledge.
Resnick, M. (2007). All I really need to know (about creative thinking) I learned (by studying how children learn) in kindergarten. *Paper presented at the Proceedings of the 6th ACM SIGCHI conference on Creativity & cognition*.
Robertson, J. (2012). Making games in the classroom: Benefits and gender concerns. *Computers & Education, 59*(2), 385–398. https://doi.org/10.1016/j.compedu.2011.12.020.
Robertson, J. (2013). The influence of a game-making project on male and female learners' attitudes to computing. *Computer Science Education, 23*(1), 58–83.
Robertson, J., & Howells, C. (2008). Computer game design: Opportunities for successful learning. *Computers & Education, 50*(2), 559–578.
Ruggiero, D., & Green, L. (2017). Problem solving through digital game design: A quantitative content analysis. *Computers in Human Behavior, 73*, 28–37.
Sáez-López, J.-M., Román-González, M., & Vázquez-Cano, E. (2016). Visual programming languages integrated across the curriculum in elementary school: A two year case study using "Scratch" in five schools. *Computers & Education, 97*, 129–141. https://doi.org/10.1016/j.compedu.2016.03.003.
Saldaña, J. (2015). *The coding manual for qualitative researchers*. Sage.
Schmitz, B., Klemke, R., Walhout, J., & Specht, M. (2015). Attuning a mobile simulation game for school children using a design-based research approach. *Computers & Education, 81*, 35–48.
Stewart-Gardiner, C., Carmichael, G., Latham, J., Lozano, N., & Greene, J. L. (2013). Influencing middle school girls to study computer science through educational computer games. *Journal of Computing Sciences in Colleges, 28*(6), 90–97.
Sung, H.-Y., & Hwang, G.-J. (2013). A collaborative game-based learning approach to improving students' learning performance in science courses. *Computers & Education, 63*, 43–51.
Teague, J. (2002). Women in computing: What brings them to it, what keeps them in it? *ACM SIGCSE Bulletin, 34*(2), 147–158.
Tuhkala, A., Wagner, M.-L., Nielsen, N., Iversen, O. S., & Kärkkäinen, T. (2018). Technology comprehension: Scaling making into a national discipline. *Paper presented at the proceedings of the conference on creativity and making in education*.
Turner, J. C., Christensen, A., Kackar-Cam, H. Z., Trucano, M., & Fulmer, S. M. (2014). Enhancing students' engagement: Report of a 3-year intervention with middle school teachers. *American Educational Research Journal, 51*(6), 1195–1226.
Wagner, A., Gray, J., Corley, J., & Wolber, D. (2013). Using app inventor in a K-12 summer camp. *Paper presented at the Proceeding of the 44th ACM technical symposium on Computer science education*.
Wang, F., & Hannafin, M. J. (2005). Design-based research and technology-enhanced learning environments. *Educational Technology Research & Development, 53*(4), 5–23.
Werner, L., Denner, J., Campe, S., & Kawamoto, D. C. (2012). The fairy performance assessment: Measuring computational thinking in middle school. *Paper presented at the proceedings of the 43rd ACM technical symposium on computer science education*.
Wilson, A., & Moffat, D. C. (2010). Evaluating Scratch to introduce younger school-children to programming. *Proceedings of the 22nd annual psychology of programming interest group*. Leganés, Spain: Universidad Carlos III de Madrid.
Wing, J. M. (2006). Computational thinking. *Communications of the ACM, 49*(3), 33–35.
Zhang, J. X., Liu, L., de Pablos, P. O., & She, J. (2014). The auxiliary role of information technology in teaching: Enhancing programming course using Alice. *International Journal of Engineering Education, 30*(3), 560–565.