



Article

Elementary Students' First Approach to Computational Thinking and Programming

Susanne Kjällander ^{1,*} , Linda Mannila ², Anna Åkerfeldt ³ and Fredrik Heintz ² ¹ Department of Child and Youth Studies, Stockholm University, 10691 Stockholm, Sweden² Department of Computer and Information Science, Linköping University, 58183 Linköping, Sweden; linda.mannila@liu.se (L.M.); fredrik.heintz@liu.se (F.H.)³ Department of Mathematics and Science Education, Stockholm University, 10691 Stockholm, Sweden; anna.akerfeldt@mnd.su.se

* Correspondence: susanne.kjallander@buv.su.se; Tel.: +46-736-789-958

Abstract: Digital competence and programming are actively highlighted areas in education worldwide. They are becoming part of curricula all over the world, including the Swedish elementary school curriculum. Children are expected to develop *computational thinking* through programming activities, mainly in mathematics—which are supposed to be based on both proven experience and scientific grounds. Both are lacking in the lower grades of elementary school. This article gives unique insight into pupils' learning during the first programming lessons based on a group of Swedish pupils' experiences when entering school. The goal of the article is to inform education policy and practice. The large interdisciplinary, longitudinal research project studies approximately 1500 students aged 6–16 and their teachers over three years, using video documentation, questionnaires, and focus group interviews. This article reports on empirical data collected during the first year in one class with 30 pupils aged 6–7 years. The social semiotic, multimodal theoretical framework “*Design for Learning*” is used to investigate potential *signs of learning* in pupils' multimodal *representations* when they, for example, use block programming in the primary and *secondary transformation unit*. We show that young pupils have positive attitudes to programming and high self-efficacy, and that pupils' signs of learning in programming are multimodal and often visible in social interactions.

Keywords: K-12 education; computational thinking; programming; design; multimodality; learning



Citation: Kjällander, S.; Mannila, L.; Åkerfeldt, A.; Heintz, F. Elementary Students' First Approach to Computational Thinking and Programming. *Educ. Sci.* **2021**, *11*, 80. <https://doi.org/10.3390/educsci11020080>

Academic Editors: João Piedade and Nuno Dorotea

Received: 31 December 2020

Accepted: 6 February 2021

Published: 19 February 2021

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



Copyright: © 2021 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

“All children should have the opportunity to develop an understanding that people are initiating the actions of computers, machines and robots”. This was said by one of the preschool teachers in a focus group interview in one of our earlier research projects [1]. It suggests “computational thinking” as described by Jeanette Wing: “Computational thinking is a fundamental skill for everyone, not just for computer scientists. To reading, writing, and arithmetic, we should add computational thinking to every child’s analytical ability” [2]. Digital competence and programming are actively highlighted areas in education worldwide. Today, researchers, policy makers, and educators agree on computational thinking (CT) and programming as important skills. The CSTA and ISTE (<https://www.iste.org/explore/computational-thinking-all> accessed on 30 December 2020) proposed a definition of CT suitable for use in K-12 education, identifying nine essential concepts: data collection, data analysis, data representation, problem decomposition, abstraction, algorithms, automation, parallelization and simulation.” [3]. Many governments, researchers, and educators agree that this is important, but one should also ask how and what pupils learn when they engage in programming activities. Since 2006, digital competence has been one of the European Commission’s eight key competencies, but a national digitalization strategy for the Swedish school system was not developed until 2017. All adults and children in education should achieve “adequate digital competence” and

knowledge development and equity should be strengthened [4]. In 2017, the Swedish government decided to introduce digital competence and programming in primary school [5]. A consequence is that all math and technology teachers in grades 1–9 in Sweden are expected to integrate programming in their teaching. Furthermore, the Swedish school law requires that teaching is based on scientific evidence and proven experience—something that so far is largely missing. Programming has mainly been studied in higher education, because it has been taught at that level for decades. Now we want to take a step towards understanding programming as part of younger children’s learning. An aim is therefore to inform education policy and practice in countries where programming is part of the curriculum for younger students (i.e., year 1–3 in elementary school, children aged 6–10). This article focuses on young pupils’ CT and learning framed by programming activities in the classroom.

1.1. Programming as a Key Competence—Also for Young Pupils

STEAM, i.e., Science, Technology, Engineering, Arts, and Mathematics, has, according to Cervera et al. [6], become a source of innovation for teaching and learning. Often, in K-12 education, digital tools and systems are implemented and used to enhance interest and learning in these five subjects. European Schoolnet [7] (p. 4) does not lean on empirical research and does not concern effects on learning, but on reports on how programming is used in education in Europe. Yet programming is described as a key competence: coding is becoming a key competence which will have to be acquired by all young students and increasingly by workers in a wide range of industries and professions. Logical reasoning is part of coding and represents one of the key skills which are part of what is now called “21st century skills”. In a meta study from Singapore, with data from 27 studies from different countries, programming was highlighted as something more than just coding, something that strives for computational thinking [2]: “Programming is more than just coding, for, it exposes students to computational thinking which involves problem-solving using computer science concepts like abstraction and decomposition.” ([8], p. i). According to the authors, most of the studies analyzed in their article are based on a view of knowledge and learning, where one can see that the participating children and young people create something when they program and that this consolidates what they have learned [8].

1.2. Physical and Visual Aspects of Programming—And How It Relates to Learning

A consistent feature in international research is that programming instructions and processes are visualized in different ways, starting with images or image flows on a screen to three-dimensional material, such as programmable bricks or different kinds of robots. Thus, a massive part of research on younger children and programming concerns the field of tangible programming, often with robots. Sapounidis and Demetriadis [9] compared programming using physical objects with visual programming in Greece. The authors showed how the youngest participants (5–8 year-olds) and girls were most positive and found it easier to use physical objects when working with programming. A randomized controlled trial (RCT) study by Kazakoff, Sullivan and Bers [10] examined younger pupils’ use of robots with both analogue, such as building blocks, and digital resources. The researchers found that the children in the robot programming group got significantly higher results in storytelling assignments. In another American study, young pupils used physical material to program a robot and the results show that children who participated in a week of intensive robot programming improved their ability to remember how different processes are related [10].

Bers [11] highlighted in particular how children’s work with robots involves much more than just building physical artifacts and suggested that “bringing robots to “life” involves computer programming. Thus, children learn to create computer programs—algorithms or sequences of instructions that allow robots to move and to sense and respond to their environment”. A robotics study [6] observed students who supported each other in programming activities and found that the use of robotics had positive contributions

to autonomous work. Teachers saw that students who were mentoring were motivated further when exploring through playing, investigating, observing, and cooperating. This kind of programming activities allow young students to engage in rudimentary skills, i.e., implementing an algorithm, debugging, and testing. Scratch was used as a tool for learning basic programming concepts in a cross-curriculum setting by Sáez López, González and Cano [12], and according to their research, pupils' use of the program made them understand computational concepts. The pupils engaged in active learning while programming in Scratch and they perceived the programming environment as both fun and motivating. On the other hand, programming does not have to be visual or physical to be used by younger pupils. A conference paper called "Teaching Programming to Young Learners Using Scala and Kojo" [13] presented how pupils in Sweden and India program. The results indicate that students experience text-based programming as something positive, since it teaches pupils what it really is like to program. The article also suggests that pupils learn abstract thinking by programming. To sum up, research suggests that younger children find it easier to learn programming by using visual or physical resources [14,15].

1.3. Learning CT and Programming—A Matter of Attitudes and Collaboration?

Robertson and Howells [14] showed that young pupils were motivated and enthusiastic for learning when they programmed. The students were determined that they wanted to achieve high results and showed abilities for both independent learning and group learning. The researchers argued that the students linked and used their learning in new situations, but it is not clear what these new situations were. The authors also noted that the students who created computer games expressed positive perceptions of game programming. Although the pupils were initially reluctant to collaborate in programming in school, Israel, Pearson, Tapia, Wherfel, and Reese [16] highlighted collaboration as an effective method for engaging young students in CT. Their research indicated that pupils became more self-reliant. Three modes of collaboration were found: teacher-prompted collaboration, organized collaboration, and student-initiated collaboration.

1.4. CT and Programming—How Early Can They Be Introduced?

Finally, since the area of CT and programming is a currently highlighted question worldwide, something should be mentioned about the suitability of introducing young pupils to CT and programming in education. A lot of countries introduce programming and CT in preschool. This was scientifically supported by Tran [17], for example, who argued that an early introduction to CT is important if pupils are to develop positive attitudes to the topic. The report "Computing Our Future" [7] reported on a study of younger children programming in Spain. The results show that children are prepared to program at a young age and that it can have effects on learning in other school subjects. The idea that young children are interested in learning programming and that they can learn to program was confirmed in Swedish research as well by, for example, Palmér [18]. Research by Irish and Kang [19] indicated that connecting programming experiences to other activities plays a decisive role in pupils' engagement and interest in programming in school, as well as for their learning.

1.5. The National Curriculum and Learning

In the Swedish national curriculum, adequate digital competence is already highlighted, focusing on children gaining an understanding of the digitalization in their everyday life [20]. Studies by the State Media Council [21] showed that almost all Swedish children use digital resources daily. Research by Martínez, Gómez, and Benotti [22] showed that even if younger children consume digital programs early, they are more seldom given the opportunity to develop understanding of how digital tools work. Programming in Swedish preschool can appear in different forms and often in other contexts than math and technology, such as play, music, digital storytelling, and aesthetic learning processes.

Younger children work collaboratively (often because the teacher arranges for pair or group work), while exploring and deconstructing problems and programming thus becomes a natural feature of cross-cutting activities in school, as shown in Kjällander and Ridder-sporre [23]. Research by Heintz and Mannila [24] discussed the possibility of scaling up teaching CT to all students, which is a reality in Sweden today. The national curriculum states the following: “Through teaching, pupils should be given the preconditions to develop their familiarity with basic mathematical concepts and methods, and their usefulness. In addition, through teaching pupils should be given opportunities to develop knowledge in using digital tools and programming to explore problems and mathematical concepts, make calculations and to present and interpret data.” ([20], p. 55.)

The core content in math that can be related to CT and programming, for grades 1–3 in primary school, is described in algebra as the following: “How unambiguous, step-by-step instructions can be constructed, described and followed as a basis for programming. The use of symbols in step-by-step instructions”. The core content in technology related to CT and programming for the corresponding grades is given as: “Working methods for developing technological solutions: Controlling objects by means of programming”.

1.6. Aim and Research Questions

To be able to teach programming on scientific grounds, research is needed to support teachers to didactically design their instruction in a way that supports their pupils’ learning of CT and programming. Since this is a new content area in the curriculum, it is difficult to know what kind of didactic design and teacher interventions are supportive or hindering. This article seeks to, from an interdisciplinary perspective, cast light on how pupils engage in CT and programming activities and how they design for their own learning in programming activities. Two research questions will guide the text:

- How do young pupils engage in computational thinking in a Learning Design Sequence?
- How can young pupils’ multimodal signs of learning be visualized in programming activities?

1.7. Theoretical Approach

The theoretical perspective used in this article is called “Design for Learning” [25–31]. The focus is on the representation of pupils’ learning and knowledge, rather than on the reception of information. The theoretical framework is social semiotics [26] and multimodality [25]. The perspective is based on a view of interaction, meaning-making and learning as sign-making multimodal activities, as visualized in the model Learning Design Sequence, Figure 1. This perspective allows for descriptions, interpretations, and analyses of detailed aspects of modes such as text, image, symbols, sound effects, gestures, speech, and colors. The perspective emphasizes communication in situated activities and focuses on a transformation process that children engage in when they learn. The empirical material is analyzed based on the Learning Design Sequence (LDS) model [27], a model used within multimodal design theoretical frameworks.

The paper focuses on how pupils take on the given programming task and the digital tools and interpret, discuss, process, collaborate, and create representations, or motivated signs, such as a movie, a game, a text, an image, or a recipe—i.e., their design in learning [30]. The theory is based on a perspective of learning known as multimodal sign-making procedures [25–31]. An LDS in primary school is framed by: learning resources, such as robots, tablets, and computers; purpose, such as curriculum and math books about programming; and institutional norms, such as that you are supposed to program your robot instead of playing with it during math class.

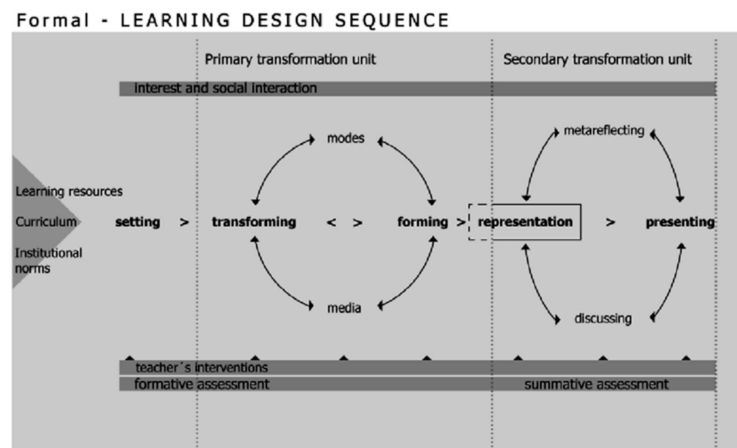


Figure 1. The analytical model Learning Design Sequence (LDS) [27].

2. Materials and Methods

This article can be described as a small-scale case study which is part of a large longitudinal project called *Programming Didactics. Learning and Assessing Programming in Primary Education* (Project number: Dnr MAW 2017.0096) focusing on programming didactics (Programming Didactics. Learning and Assessing Programming in Primary Education. The purpose of the project is to investigate how students learn programming and how these competencies can be assessed in order to provide scientifically grounded programming didactic for primary education. It is a longitudinal study of students programming in classroom settings and develops an assessment framework to be used in the longitudinal study for measuring students' programming competences. The study will allow us to better understand how students learn programming and what good teaching practices are.). The project is still ongoing and uses mixed methods, as data are collected through both quantitative methods, such as questionnaires, and qualitative methods, such as field work. Two schools with approximately 1500 students in total are studied for three years. Two classes in grade 1, two in grade 4, and two in grade 7 are followed for three years, and at the end of the project we will have collected empirical material from all compulsory school grades (1–9). The article at hand was designed as a case study illustrating empirical data in detail from one class (year 1) with 30 pupils, as an illustrative example.

The multimodal empirical examples presented in this article were based on video documentation of classroom interaction and on recorded focus group discussions with teachers. Examples were chosen to represent how programming and CT can become part of classroom teaching and what kind of programming affordances [32] that are used by young pupils in school. Even though teacher's didactic design is of great interest, this paper especially cast light on young pupils' explorative engagement in programming activities. A design theoretical analysis was made of the qualitative research material such as videos and interview recordings.

The analysis was divided into the same sections as the LDS model: the setting and the primary and secondary transformation units. The theoretical perspective allows for descriptions, interpretations and analyses of detailed aspects of modes [26,31] and therefore the analysis was underpinned by multimodal transcription charts, separating the dense and multimodal video material into different columns for different modes such as speech, gestures, sounds, and screen activity. The transcriptions were transformed into more readable excerpts and, if possible, illustrated by anonymous photos. The citations in the excerpts were translated as close to the Swedish meaning as possible, and might therefore stand out as "Swenglish".

The quantitative material comprised four questionnaires (the four questionnaires were aimed at different groups: teachers; students grade 1–3; students grade 4–6; and students grade 7–9) distributed online annually to all teachers and approximately 1500 pupils at the two participating schools. The design of the questionnaires was inspired by Leifheit

et al. [33], who piloted a questionnaire based on already existing scales for mathematics to gain insight into students' beliefs and attitudes towards programming, and thoroughly described in another article published from our research project by Mannilla et al. [34]. Self-efficacy and prior experience were most important to include in the questionnaires since the main aim of the project is to develop teaching methods and materials for assessment, but also questions about children's attitudes (if they think it is fun or hard to program) were asked and analyzed. The questionnaires were analyzed in Excel and the results are presented as percentages in tables.

The research project went through an ethical review. The project was approved by the Swedish Ethical Review Authority on the 22 November 2018 (Ethical approval code/registration number: Dnr 2018/485-31). All ethical guidelines were thoroughly followed. To mention a few examples relevant to this article, pupils' guardians received oral and written information and they signed informed consent. Pupils were asked for consent at the beginning of each observation, and their consent always outweighed that of the guardians. The researchers at every opportunity made sure that the children really want to be involved in the project. Images presented were deidentified and all personal data of the participants were protected from unauthorized access.

3. Results

In this part of the article, we will present the results using the Learning Design Sequence model's three parts—setting, primary transformation unit, and secondary transformation unit—along with the theoretical notions posted in the model.

3.1. Setting—Designs for Learning

This elementary school has about 1000 pupils aged of 6 to 16. It is at the digital cutting edge and uses a vast amount of digital learning resources, both hard- and software. The institutional norms at this specific school tell teachers to use digital resources as an integrated and natural part of teaching. Most teachers have adequate digital competence, but programming as a content area is new to most Swedish schoolteachers. In interviews, the teachers in this specific class claim that they feel uncomfortable with teaching programming, since this is the first time they will do it: school has gained a new purpose. The teachers had some in-service training in programming, but did not consider themselves as experienced. Fieldnotes from discussions with teachers during these lessons state that: "It is obvious that programming and computational thinking only is understood as something new and different to adults—teachers and researchers. To kids today, it is just one among many new assignments, no more mystic or exciting than addition or multiplication." We followed the very first programming lesson a class of 6–7-year-olds had in school. This was a math lesson planned to cover CT, as this is the topic of two pages in the math book.

3.2. Primary Transformation Unit—Learning, Transforming and Forming

At the beginning of the primary transformation unit, pupils interpret the task given in the setting and begin transforming information and based on their own interests, forming knowledge with different modes and media. Media is the meaning-making, actional, visual and linguistic resources [27,29,30] that teachers and pupils use in their *social interaction*, to communicate. Digital learning resources can offer potential for pupils to work with realistic multimodal simulations as mentioned by Schaffer [35]. In the following example, the teacher begins by creating a mind map, asking children to give suggestions on what programming is according to their own experience and interests.

The children propose several suggestions, that are visualized in Swedish on the White Board below in Figure 2:

- To control a character.
- A remote control.
- An app to run a bike.
- To check instructions and build Lego.

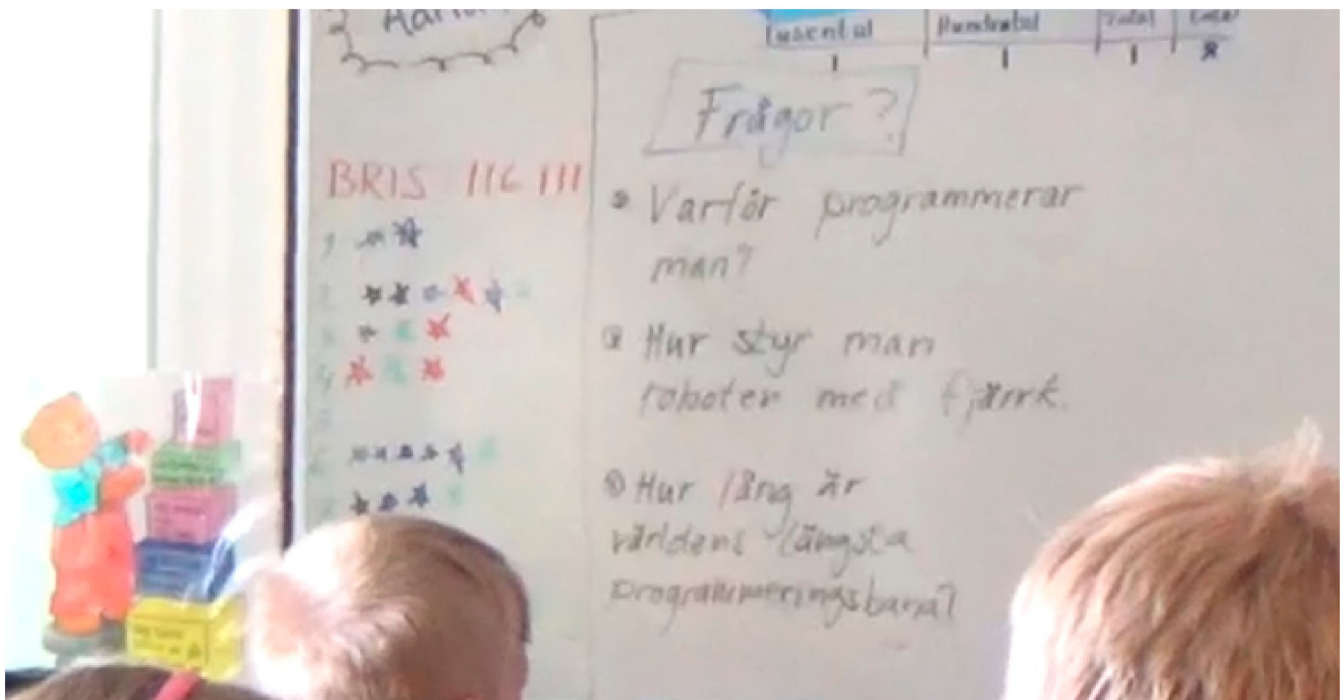


Figure 2. Pupils questions (in Swedish) at the whiteboard in the classroom.

A long discussion takes place, ending up in the following questions posed by the pupils:

- Why do you have to do programming?
- How do you control robots by remote control?
- How long is the world's longest programming track?

A common didactic design in the two project schools' lower grades (grade 1–3) is to show an episode from the TV program "Program mera" (In Swedish: "Programmerarna").) from the Swedish Broadcast Company, and so it is also shown in this empirical example. All pupils are intensively focused on the screen, expressing their interest and appreciation and discussing the content together afterwards.

After the TV program, the teacher intervenes by posing the question:

"What have you learned?"

The pupils answer:

- "Programming!"
- "How to program in different ways!"
- "With patterns!"
- "Instead of letters they read ones and zeros that becomes letters . . . "

Thereafter the teacher presents a ScratchJr assignment expressing the purpose:

"Here comes the guideline, and it is as follows: You shall make your character walk five steps forward and make a jump. And then, it is like this, that when you have made that code, you shall hand in your iPads, and I will see, we will see, if you have made that code."

The teacher informs the pupils about the basic concepts and constructs, such as that the flag-block indicates the beginning and how the repeat-block works. All pupils have one tablet each, but work in pairs, resulting in all children wanting to use their own tablet. The pairs work together but also, at almost all times, they interact with other pairs, asking questions, giving compliments, and commenting on each other's programs and the corresponding code. Such a group work is illustrated in the excerpt below and in Figure 3.



Figure 3. Pupil A and B programming in ScratchJr at a digital tablet.

“Two pupils—A and B—are programming their character according to the instructions. They have chosen a female character and the library as a scene. A: There! B: *“But. Now one should have gone five steps.”* A: *“What about number three?”* B: *“But, what are you supposed to do next? YES! You must have a flag! A flag!”* A pulls the flag icon down to the bar. B points at the screen. A: OK. (. . .) A scrolls between all commands/icons and she chooses the “jump” and thinks for a long time until she puts it into the right place. They make the character jump up into the air, and stay there, which they think looks funny.”

The girls use representational and communicative modes [31] such as movements, images, and colors [30]. Different modes in ScratchJr offer different potential options and impose different limitations for meaning making and interaction on the pupils [29]. The girls recognize and comprehend different affordances [32] that they explore and make use of. For example, the possibility to choose and design a character is explored by the girls, compared to other pupils in the classroom, who only use the pre-selected cat. They engage in what Cervera et al. [6] would call rudimentary skills i.e., implementing an algorithm, debugging, and testing. As with the pupils in Cervera et al.’s study [6], the pupils in this example seemed to be motivated while playing, investigating, observing, and cooperating, letting their own interest guide their design in learning. These actions are illustrated below in an excerpt and in Figure 4.

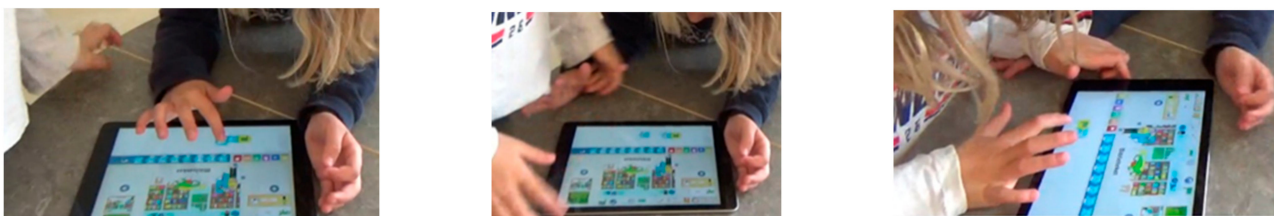


Figure 4. Pupil A takes the tablet from pupil B while programming blocks in ScratchJr.

“The girls A and B choose and design a new character, a male librarian, in the same scene: the library. They choose clothes, hair, and accessories when the teacher calls for them to hand in their tablets for a presentation on the interactive board. The girls want to quickly finish their code and B quickly puts a number of blocks together to code the five steps. A: *“No, you should not take five of those . . . ”* A pushes B’s hand away and then takes

the tablet away from her and writes the number 5 below the block to make the librarian take five steps and then she codes the character to jump both up and down.”

According to Lye and Koh [8], CT includes an understanding of and knowledge about computational notions such as code, sequence, and loop, something that is practiced in this example: pupils form and transform programming multimodally. Since visual programming, or block programming, is based on building blocks similar to puzzle pieces with predefined functionality and put together in a sequence, it is possible for young children to program. The flag was something expressed as important by the teacher in the setting/primary transformation unit, and when the pupils programmed, they could not get the program to work, therefore they form and transform, or, expressed with computational notions, deconstruct and decompose [27], trying to find the mistake. This means that they walk through the Learning Design Sequence back and forth, or in circular movements, through the first and second transformation units, meta-reflecting on their own learning and representations, trying to find mistakes and learn more. Signs of learning are expressed here, since their new knowledge was transformed and formed in a new situation [29,30] on the tablet instead of the white board. The flag is here expressed as an affordance, as described by Gibson [32], or a possible solution, to make the code work—and it works.

A gender difference is visible in this limited case study of a classroom in how girls tend to design their character and background for a long time, while the boys take on the assignment, the program, faster and then go on to the next assignment—something that is confirmed in interviews with the teacher. Some research showed no gender differences [36], but girls are often thought to like narrative coding more. The two girls in this example do the coding part quickly and then engage in the design. More research on gender and programming is needed, especially since researchers such as Heikkilä [36] argued that this discourse traditionally comes from the fact that men as a group stand for “the technical”, the harsh, the abstract, and the elusive. There is therefore a risk that girls feel less welcome, less included, and not as secure in programming. From a learning perspective, this means that some children might feel more included than others, and ultimately learn more. A lot of initiatives worldwide are currently actively working to fight against this, and hopefully the early introduction of CT and programming in education will decrease these inequalities. Research by Palmer [37] highlighted the importance of teachers’ expectations for students’ motivation and ability to learn. Her research illustrated that gender can affect pupils’ self-efficacy, such that it can become more acceptable for a girl to perform less well than a boy in male coded subjects such as programming.

The children in this example are asked to “show and tell” their programming activities, i.e., their code that is supposed to be “five steps and a jump”. Fourteen representations of codes are displayed on the smartboard and a majority of them are correct. Quite little attention is paid to the design; here it only matters if the pupils have managed to construct a program that runs or not. This focus on the result when running the code goes for both boys and girls (including the latter, who put a lot of effort in designing the characters and the environment during the lesson) as well as for the teacher.

3.3. Secondary Transformation Unit—Learning, Metareflection and Discussion

The teacher intervenes by praising all pupils’ representations, paying the program full attention. A didactic design of discussing and meta-reflection is at this point staged at the White Board in the classroom as shown in Figure 5. This is a kind of intervention (or assessment) act since grades are not used until in grade 6 in Sweden. The ones who are correct, such as pupil A and B in this example, are strongly encouraged, even if their code is much simpler than the code of other pupils, who have advanced to make their own code.

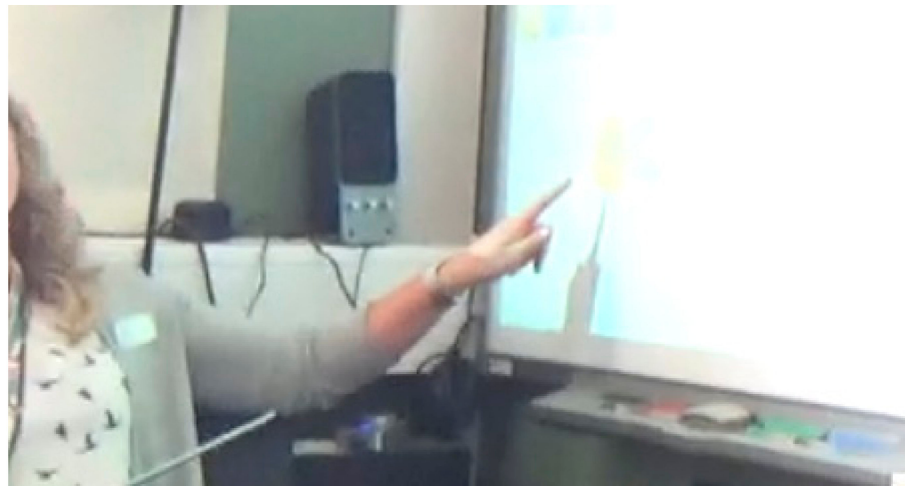


Figure 5. The teacher presents all pupils programs on the interactive whiteboard.

The teacher's intervention in the learning process with the pupils who have tried to construct a functioning program without help is pronounced in a positive voice:

"Five steps and then it jumps up into the air and stays there: good!"

The teacher's response to pupils who have not managed to make the program work is pronounced in a comforting voice:

"It doesn't matter!"

The teacher's response to pupils who have ignored the assignment and made their own code and thus not succeeded is pronounced in a challenging voice:

"You did a programming activity, absolutely, but it was not according to the instruction. You made your own guideline, your own code, which was 5 forward, 5 turn arounds, and then that is supposed to repeat over and over again, or it shall repeat five times. That was the instruction that you were given. But, you did not follow the one that the class was supposed to do. You made your own."

Making a representation is a matter of deliberate design [30]. The pupils' learning is related to what they represent multimodally. It does not focus on what is found inside the pupil's mind—the "internal signs"; instead, the "external signs" are viewed. In this example, their motivated signs can, for example, be their advancement in using the flag and their new knowledge to form, or code, a repetition instead of using the same command several times. These are examples of explicit signs of learning.

In programming, it is important to follow instructions, which is clear in the feedback from the teacher. It is a learning activity where pupils are supposed to try, fail, debug, and try again until they learn, supported by their peers and teacher. This is the first time this group of pupils program and the teacher is very supportive, giving feedback at the end of the lesson right before the break, which probably explains the lack of important discussion about what went wrong, how it could be corrected, and what should be tried next time.

3.4. Secondary Transformation Unit—Attitudes, Learning and Meta Reflection

The research project is both mixed discipline and mixed methods. Video ethnography is combined with both focus groups, field notes, and interviews, as well as questionnaires.

All pupils in the two project schools (about 1500 pupils) are asked to respond to an annual questionnaire. After these few lessons in CT/programming, children in this group, and some other 1st graders at the two project schools, were asked to respond to the questionnaire online by choosing the emoji corresponding to their own feelings, as shown in Figure 6. Emojis were used to express the response options, since children at this age might not be good readers. Endorsing a statement with an agreement response works out

fine in most questions. The statement “Programming is difficult”, however, may introduce bias, due to the response being contrary to the emotion of the prompt [38].

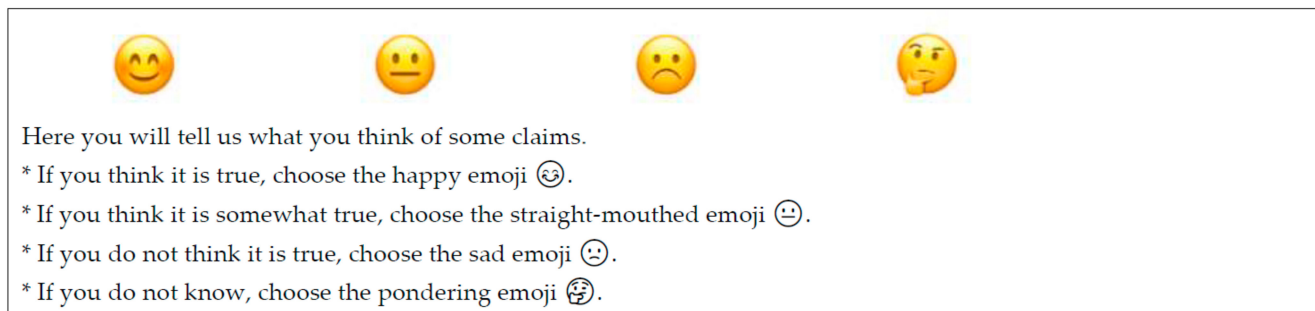


Figure 6. The response options in the self-efficacy questionnaire for younger pupils (grade 1–3).

The questionnaire included a few statements such as “Programming is fun”, “I usually program in my spare time”, and “Programming is difficult”. The first two of these are easy to answer with a happy emoji if the corresponding statement is true, but it can be confusing to respond to “programming is difficult” with a happy emoji [38].

The results for the three statements relevant for this article are shown in the Table 1:

Table 1. The results of relevant questions among younger pupils (grade 1–3) in the self-efficacy questionnaire.

	True	Somewhat true	Not at all true	Do not know
Programming is fun.	79.3 %	17.2 %	1.7 %	1.7 %
Programming is difficult	22.4 %	43.1 %	20.7 %	13.8 %
I would like to have more programming in school.	65.5 %	20.7 %	5.2 %	8.6 %
I am skilled at programming.	48.3 %	32.8 %	10.3 %	8.6 %

Earlier research [37] indicated that self-efficacy affects learning and that there can be gender differences in programming, as boys consider themselves as being more comfortable and knowledgeable in programming than girls do. In our very limited questionnaire, 54% of the boys, compared to 38% of the girls, considered themselves skilled at programming. This was, however, not compared to their actual practical programming skills.

Given the related research presented earlier, pupils’ attitudes and self-esteem in programming stands as a baseline for their possibility to think computationally and to learn. About half of the pupils in this study think that they are skilled in programming, while one third think they are rather skilled. Only a few pupils feel that they are unskilled. This can be related to four out of five pupils thinking that programming is fun and more than half of them urging for more programming in school. Earlier research [17] argued and showed that an early introduction to CT makes pupils develop positive attitudes to programming. These pupils most likely have prior experience of programming from preschool, since many Swedish preschools include programming [23]. Even if this is the first time they program and use ScratchJr in school, they may have done so previously. There is a possibility that programming lessons should be made somewhat more challenging for pupils to learn more, since only one out of five say that they think programming is hard (with the reminder that this is a limited case study, and that the questionnaire question might not be sufficiently reliably posed [38]).

4. Discussion

In this section, we discuss our research questions about how young pupils engage in CT and how their multimodal signs of learning are visualized in programming activities.

Programming has only been part of the national curriculum in Sweden since 2018 [20] and the teacher questionnaires in this study (but not presented in this article since focus is on pupils' learning) bear witness to a profession on shaky grounds; teachers do not feel that they have adequate digital competence, yet are supposed to teach children to become digitally competent. The empirical examples show that the teacher teaches what she is supposed to according to the national curriculum for grades 1–3 in math: step-by-step instructions and how they can be constructed and described as well as how they can be followed, which forms a basis for programming [2]. This empirical study's mixed quantitative and qualitative results illustrate how CT and programming is taught and learned for its own sake, which is opposed to earlier research [6] arguing that programming is used to motivate learning in STEAM subjects. It can appear in relation to, for example—as discussed by Kjällander and Riddersporre, 2019 [23]—play, storytelling, and aesthetic learning processes, but programming is the core learning goal. Earlier research showed that block programming and concrete resources (often tangible, such as for example robots) are used in education all around the world [10] when the youngest pupils learn to program, something that is also valid for this study where pupils' very first meeting with programming is studied and analyzed.

Out of CSTA and ISTE's nine essential concepts of CT, data representation, problem decomposition, abstraction, algorithms, automation, and simulation are illustrated in these empirical examples.

It is often claimed that children's attitudes towards programming are most positive among younger pupils (and boys). This is also the case in the questionnaire results from our two project schools, where a majority of children that have just begun to program (6–7 years old) claim that they think programming is fun. This is not uncommon, but rather confirms earlier research, for example by Robertson and Howells [14], who showed young pupils' motivation and enthusiasm for learning when programming. In this article, we only presented results from the first questionnaire distributed after grade 1, but they show a clear result: a vast majority of pupils in grade 1 enjoy programming, they want more programming in school and they consider themselves skilled in programming.

Signs of learning [30] are in the empirical examples expressed when new knowledge is used in new situations—pupils transform and form instructions they have learned and discussed in circle time to a programming language with visual programming blocks on a tablet screen. Their signs of learning are always multimodally represented [25] and are often seen in social interactions between pupils and between pupils and teachers. Signs of learning are visible in children's questions about their programmed daily-life surroundings in the introduction, in their answers as to what they think they have learned from the TV program, in their own program in ScratchJr and its constructs, in their speech including computational notions such as “repetition”, and in their choice of colors and shapes when they design their characters, as well as when they seem to act with self-esteem when they grab the tablet from each other to take over the programming activity.

Research [36,37] has sometimes claimed that there are gender differences between school students, but not among preschoolers. In the limited results from the questionnaires, we can see that boys tend to present themselves as skilled programmers more often than girls do. There is a weak tendency of difference in gender in the empirical example as well, where girls engage more in the design of the characters and background than boys, but we did not find if, and if so, how this affects their learning. Since earlier research highlighted the risk of programming being a male coded topic, it is important to discuss this at larger scale than in this limited study.

Research [37] highlighted the importance of teachers' expectations for students' motivation and ability to learn. In this example, the teacher expects all pupils to manage learning how to use ScratchJr and learning to program by creating code that makes a character take five steps and then jump. She praises pupils differently depending on their representations, not only depending on if they succeeded or not. In focus group discussions in this project, teachers argue that it is more important to follow instructions in

programming lessons than in many other subjects, since learning programming is closely linked to following instructions.

Research [16] showed that pupils initially (but not later) were reluctant to collaborate when programming in school, which is contrary to what we experienced in the classroom observations in our two project schools. Research [6] indicated positive results for students mentoring, while programming and the project were met with a positive reception. This is similar to the didactic design of our study where children help each other in a less structured, but still didactically designed method. In the TV program by the Swedish Broadcast Company that the pupils in the empirical examples watch regularly in math lessons, children are working in groups. The same goes for children in the majority of classes at all stages at both project schools that have been studied so far in our research project, during a period of almost 3 years. Programming is conducted in social interactive and collaborative processes. According to Israel et al.'s [16] three modes of collaboration, the pupils engage in teacher-prompted collaboration, something that seems to support pupils' learning. The two pupils that are exemplified in this article learn from each other, as they probably have different knowledge and skills in programming. Each lesson ends with a more organized mode of collaboration, where, for example, a "show-and-tell" didactic design is orchestrated, and opportunities are provided where pupils can learn from each other's mistakes and progress. In short, this means that they engage in rudimentary skills as described by Cervera et al. [6] when they can discuss and meta-reflect on their own transformative learning process, as well as learn from each other's representations of how they implemented an algorithm as well as debugged it and tested it.

5. Conclusions

The findings in this small-scale empirical case study show that children who are introduced to programming for the very first time in school have a positive attitude towards programming and they have high self-efficacy. Their knowledge is often represented and visible in social interaction with peers and teachers.

Author Contributions: Conceptualization, S.K., F.H., L.M. and A.Å.; methodology, S.K., F.H., L.M. and A.Å.; software, L.M.; validation, S.K., F.H., L.M. and A.Å.; formal analysis, S.K.; investigation, S.K., F.H., L.M. and A.Å.; resources, S.K., F.H., L.M. and A.Å.; data curation, S.K., F.H., L.M. and A.Å.; writing—original draft preparation, S.K.; writing—review and editing, S.K., F.H., L.M. and A.Å.; visualization, S.K., L.M.; supervision, F.H.; project administration, F.H.; funding acquisition, S.K., F.H., L.M. and A.Å. All authors have read and agreed to the published version of the manuscript.

Funding: Marcus and Amalia Wallenberg Foundation: Dnr MAW 2017.0096.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Informed consent was obtained from all subjects involved in the study.

Data Availability Statement: Not applicable.

Acknowledgments: We want to thank all participating students and teachers that welcomed us into their classrooms during these years.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Kjällander, S. *Plattan i Mattan: Digitala lärplattor Och Didaktisk Design i Förskolan*. Forskningsrapport. Uppsala Vård & Bildning. Available online: www.uppsala.se (accessed on 4 March 2016).
2. Wing, J. Computational thinking. *Commun. ACM* **2006**, *49*, 33–36. [CrossRef]
3. Mannila, L.; Dagiene, V.; Demo, B.; Grgurina, N.; Mirolo, C.; Rolandsson, L.; Settle, A. Computational Thinking in K-9 Education. In Proceedings of the 2014 Conference on Innovation & Technology in Computer Science Education (ITICSE '14), Uppsala, Sweden, 23–25 June 2014; pp. 1–29.
4. Regeringen. *Nationell Digitaliseringsstrategi för Skolväsendet; Bilaga Till Regeringsbeslut I*; Swedish Government: Stockholm, Sweden, 2017; Volume 1.

5. Heintz, F.; Mannila, L.; Nordén, L.; Parnes, P.; Björn, R. Introducing Programming and Digital Competence in Swedish K–9 Education. In *Informatics in Schools: Focus on Learning Programming, Proceedings of the Informatics in Schools: Focus on Learning Programming: 10th International Conference on Informatics in Schools: Situation Evolution and Perspective (ISSEP), Helsinki, Finland, 13–15 November 2017*; Lecture Notes in Computer Science #10696; Valentina, D., Arto, H., Eds.; Springer: Berlin/Heidelberg, Germany, 2017; pp. 117–128, ISBN 9783319714820, 9783319714837.
6. Cervera, N.; Diago, P.D.; Orcos, L.; Yáñez, D.F. The Acquisition of Computational Thinking through Mentoring: An Exploratory Study. *Educ. Sci.* **2020**, *10*, 202. [CrossRef]
7. European Schoolnet. Computing our Future—Computer Programming and Coding Priorities, School Curricula and Initiatives across Europe. Publisher: European Schoolnet Brussels: Belgium Published: October 2015. Available online: <http://www.eun.org> (accessed on 15 February 2021).
8. Lye, S.Y.; Koh, J.H.L. Review on teaching and learning of computational thinking through programming: What is next for K-12? *Comput. Hum. Behav.* **2014**, *41*, 51–61. [CrossRef]
9. Sapounidis, T.; Demetriadis, S. Tangible versus graphical user interfaces for robot programming: Exploring cross-age children’s preferences. *Pers. Ubiquitous Comput.* **2013**, *17*, 1775–1786. [CrossRef]
10. Kazakoff, E.R.; Sullivan, A.; Bers, M.U. The Effect of a Classroom-Based Intensive Robotics and Programming Workshop on Sequencing Ability in Early Childhood. *Early Child. Educ. J.* **2013**, *41*, 245–255. [CrossRef]
11. Bers, M.U. The TangibleK robotics program: Applied computational thinking for young children. *Early Child. Research Pract.* **2010**, *12*, 1524–1539.
12. Sáez-López, J.M.; Román-González, M.; Vázquez-Cano, E. Visual programming languages integrated across the curriculum in elementary school: A two year case study using “Scratch” in five schools. *Comput. Educ.* **2016**, *97*, 129–141. [CrossRef]
13. Regnell, B.; Pant, L.; Teaching Programming to Young Learners Using Scala and Kojo. LTHs 8:e Pedagogiska Inspirationskonferens. Available online: <http://lup.lub.lu.se/record/4780249> (accessed on 17 December 2014).
14. Robertson, J.; Howells, C. Computer game design: Opportunities for successful learning. *Comput. Educ.* **2008**, *50*, 559–578. [CrossRef]
15. Christensen, D.J.; Fogh, R.; Lund, H.H. Playte, a tangible interface for engaging human-robot interaction. In *Proceedings of the Robot and Human Interactive Communication, RO-MAN: The 23rd IEEE International Symposium, Edinburgh, UK, 25–29 August 2014*; pp. 56–62.
16. Israel, M.; Pearson, J.N.; Tapia, T.; Wherfel, Q.M.; Reese, G. Supporting all learners in school-wide computational thinking: A cross-case qualitative analysis. *Comput. Educ.* **2015**, *82*, 263–279. [CrossRef]
17. Tran, Y. Computational thinking equity in elementary classrooms: What third-grade students know and can do. *J. Educ. Comput. Res.* **2019**, *57*, 3–31. [CrossRef]
18. Palmér, H. Programming in preschool with a focus on learning mathematics. *Int. Res. Early Child. Educ.* **2017**, *8*, 75–87.
19. Irish, T.; Kang, N.-H. Connecting classroom science with everyday life: Teachers’ attempts and students’ insights. *Int. J. Sci. Math. Educ.* **2018**, *16*, 1227–1245. [CrossRef]
20. National Agency for Education. *Curriculum for the Compulsory School, Preschool Class and School-Age Educare*; (Revised 2018); Skolverket: Stockholm, Sweden, 2019; ISBN 978-913832734-0. Available online: <https://www.skolverket.se/publikationer?id=3984> (accessed on 15 February 2021).
21. State Media Council. Småungar Och Medier 2019. Publication date: 10 September 2019. Available online: <https://statensmedierad.se/publikationer/ungarochmedier/smaungarochmedier2019.3348.html> (accessed on 15 February 2021).
22. Martinez, C.; Gomez, M.J.; Benotti, L. A comparison of preschool and elementary school children learning computer science concepts through a multilanguage robot programming platform. In *Proceedings of the 2015 ACM Conference on Innovation and Technology in Computer Science Education, Vilnius, Lithuania, 6–8 July 2015*; pp. 159–164.
23. Kjällander, S.; Riddersporre, B. *Digitalisering i en Förskola på Vetenskaplig Grund*; Natur & Kultur: Stockholm, Sweden, 2019.
24. Heintz, F.; Mannila, L. Computational Thinking for All An Experience Report on Scaling up Teaching Computational Thinking to All Students in a Major City in Sweden. In *Proceedings of the 49th ACM Technical Symposium on Computer Science Education (SIGCSE), Association for Computing Machinery (ACM), Baltimore, MD, USA, 21–24 February 2018*; Association for Computing Machinery: New York, NY, USA; pp. 137–142, ISBN 978-1-4503-5103-4.
25. Kress, G. *Multimodality: A Social Semiotic Approach to Contemporary Communication*; Routledge: London, UK, 2010.
26. van Leeuwen, T. *Introducing Social Semiotics*; Routledge: London, UK, 2005.
27. Selander, S. Didaktisk Design. In *Didaktisk Design i Digital Miljö nya Möjligheter för Lärande*; Selander, S., Svärden-Åberg, E., Eds.; Liber: Stockholm, Sweden, 2009.
28. Jewitt, C. E. *The Routledge Handbook of Multimodal Analysis*; Routledge: New York, NY, USA, 2009.
29. Selander, S. *Didaktiken Efter Vygotskij: Design för Lärande*; Liber: Stockholm, Sweden, 2017.
30. Selander, S.; Kress, G. *Design för Lärande: Ett Multimodalt Perspektiv*; Norstedts Akademiska Förlag: Stockholm, Sweden, 2010.
31. Kress, G. What is mode? I: C. Jewitt (red.). In *The Routledge Handbook of Multimodal Analysis*; Routledge: London, UK; New York, NY, USA, 2009.
32. Gibson, J.J. *The Ecological Approach to Visual Perception*; Houghton Mifflin: Boston, MA, USA, 1979.
33. Leifheit, L.; Tsarava, K.; Moeller, K.; Ostermann, K.; Golle, J.; Trautwein, U.; Ninaus, M. Development of a questionnaire on self-concept, motivational beliefs, and attitude towards programming. In *Proceedings of the 14th Workshop in Primary and Secondary Computing Education, WiPSCÉ’19, Glasgow, Scotland, 23–25 October 2019*.

34. Mannila, L.; Heintz, F.; Kjällander, S.; Åkerfeldt, A. Programming in primary education: Towards a research based assessment framework. In Proceedings of the Woodstock '18: ACM Symposium on Neural Gaze Detection, Woodstock, NY, USA, 3–5 June 2018; ACM: New York, NY, USA, 2020. [\[CrossRef\]](#)
35. Shaffer, D.W. *How Computer Games Help Children Learn*; Palgrave Macmillan: New York, NY, USA, 2006.
36. Häikiilä, M. Vad händer när roboten får ögonfransar? Genusperspektiv på programmering i förskolan. In *Digitalisering i en Förskola på Vetenskaplig Grund*; Kjällander, S., Riddersporre, B., Eds.; Natur & Kultur: Stockholm, Sweden, 2019.
37. Palmer, A. *Hur Blir Man Matematisk?: Att Skapa Nya Relationer Till Matematik Och Genus i Arbetet Med Yngre Barn*; Liber: Stockholm, Sweden, 2011.
38. Krosnick, J.A. Survey research. *Annu. Rev. Psychol.* **1999**, *50*, 537–567. [\[CrossRef\]](#) [\[PubMed\]](#)