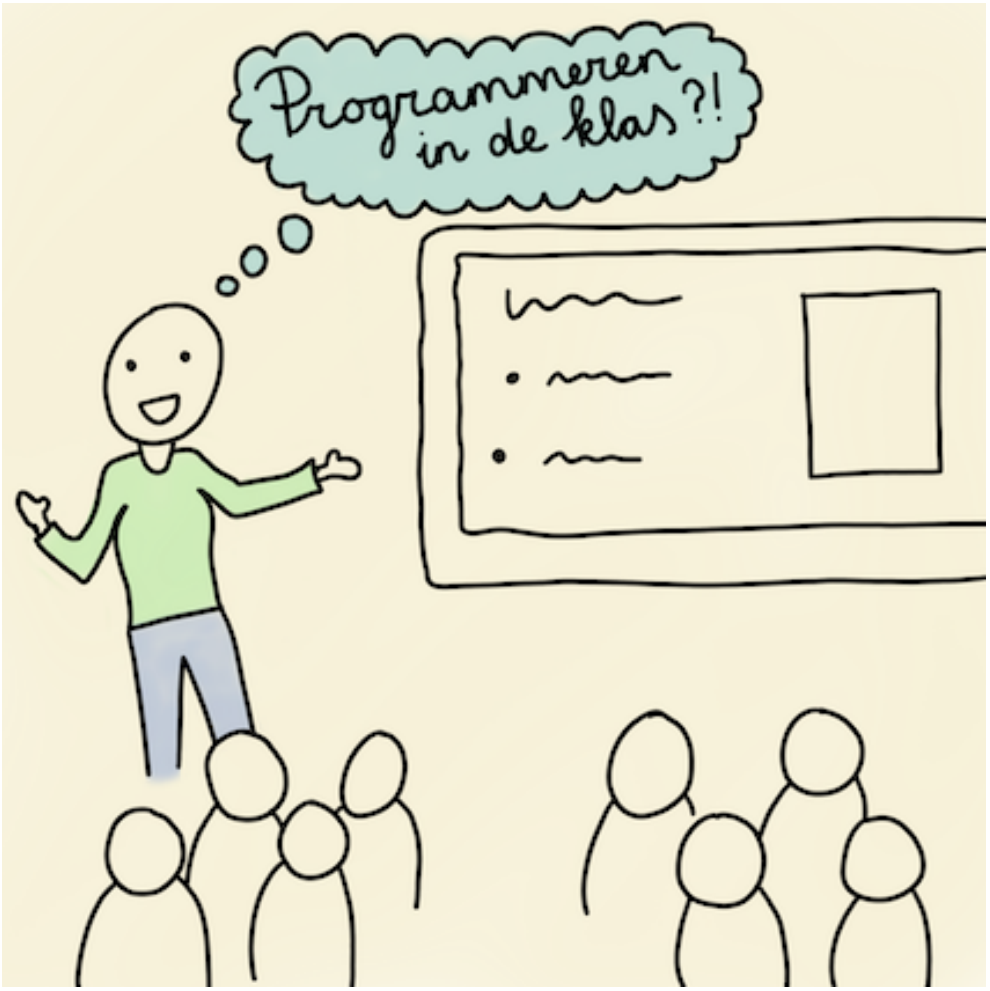
Self-efficacy of Dutch primary school teachers towards programming education



Shirley de Wit

Self-efficacy of Dutch primary school teachers towards programming education

THESIS

submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE

in

COMPUTER SCIENCE
&
SCIENCE COMMUNICATION

by

Shirley de Wit
born in Rotterdam, the Netherlands

Software Engineering Research Group
Faculty EEMCS,
Delft University of Technology
Delft, the Netherlands
www.ewi.tudelft.nl

Science Education and Communication
Faculty Applied Sciences,
Delft University of Technology
Delft, the Netherlands
www.sec.tudelft.nl

Self-efficacy of Dutch primary school teachers towards programming education

Author: Shirley de Wit
Email: S.deWit@tudelft.nl

Summary

Only 32% of the Dutch primary schools do something with programming, despite its benefits on students' skills, job opportunities, understanding of the world, and diversity. Primary school teachers not getting involved with programming education can be caused by a lack of belief in their teaching programming abilities. The belief in your own abilities towards a certain task is referred to as self-efficacy. We hypothesize that Dutch primary school teachers have a low self-efficacy towards programming education (partially) due to the teaching method used in programming education. Programming is often taught via discovery learning while Dutch primary school teachers use teacher-centered methods such as direct instruction more often. The usage of a certain teaching method can also influence the performance of the teacher. When teachers are able to use their existing knowledge and skills, then they have adaptive expertise. Combining the concepts of self-efficacy and adaptive expertise can give us insights into how teachers cope with the new situation of teaching programming.

This research aims at investigating the effect of teaching methods on the self-efficacy of Dutch primary school teachers towards programming education by answering the following question: *What is the effect of teaching methods on the self-efficacy of Dutch primary school teachers towards programming education?* Furthermore, we explore the role of adaptive expertise which is the ability to perform well in an unfamiliar situation by using existing knowledge and skills. This concept has the potential to give more insights into how self-efficacy changes.

The research question is answered through a mixed methods study consisting of a questionnaire, as part of a quantitative research, and an experiment, as part of a qualitative research.

The questionnaire includes questions about teaching methods, self-efficacy, and demographics. In total 259 teachers participated within the questionnaire. We found that both direct instruction and discovery learning are well known by Dutch primary school teachers. Direct instruction is used by almost all participating teachers in their

regular education. However, discovery learning is used by only 53% of the teachers in our sample. Furthermore, we found that gender, grade, programming experience, and programming teaching experience results in differences in teachers' self-efficacy towards programming education. Another result from the quantitative research is that teachers who expect to use discover learning in programming lessons have a different self-efficacy than teachers that do not expect this.

An experiment is set-up in which ten Dutch primary school teachers give four programming lessons in Scratch while following either a direct instruction or discovery learning approach. Data is gathered on self-efficacy and adaptive expertise via a questionnaire and a semi-structured interview. We found that getting involved in programming education, independently of the teaching method, can increase the self-efficacy of Dutch primary school teachers. However, the teachers in the discovery learning expressed more negative feelings towards the experience of teaching programming. The usage of discovery learning also seems to bring the risk of a decrease in self-efficacy. We think that adaptive expertise gives us more insights into this decrease. The teachers with a decreased self-efficacy showed higher levels of adaptive expertise. We think that this contributes to them having higher expectations of themselves. When these expectations are not met, a decrease in self-efficacy can occur.

The research we have done indicate that the self-efficacy towards programming education of Dutch primary school teachers can increase while using either direct instruction or discovery learning. The experiment indicates that the usage of discovery learning can cause more negative feelings as well as the risk of a decrease in self-efficacy.

Thesis Committee Computer Science:

Chair:	dr.ir. Willem-Paul Brinkman, Faculty EEMCS, TU Delft
University supervisor:	dr. ir. Felienne Hermans, Faculty EEMCS, TU Delft
Committee Member:	drs. Caroline Wehrmann, Faculty AS, TU Delft

Thesis Committee Science Communication:

Chair:	drs. Caroline Wehrmann, Faculty AS, TU Delft
Committee Member:	drs. Martin Bruggink, Faculty AS, TU Delft
Committee Member:	dr. ir. Felienne Hermans, Faculty EEMCS, TU Delft

Preface

I hope that this research will make the people involved in programming education aware of the importance of teaching methods. Especially people that, like me, have a background in computer science might find themselves taking the discovery learning approach for granted. During this thesis, I learned about the different teaching methods and how we can use them to our benefit.

Teachers that I talked to, both within and outside the context of this thesis, are often enthusiastic teachers that want to learn about programming and are open-minded to try out new things. But I also see those teachers struggling and trying to invent the same wheel. There seems to be a need for clear structured lessons. Lessons with a clear manual to reduce preparation time for a programming lesson. I hope that the manuals that I have created, even though there are only four of them, enable more teachers to get started with programming.

I hope to contribute my bit to making programming accessible for everyone. For me, programming is not a goal but a tool that you can use to solve problems. Also, during my studies in Delft, I never felt motivated to write a program just for the sake of it. But I am motivated by what we can achieve with it. How we can use a piece of software in therapies to virtually expose people to heights or social situations they are uncomfortable with. How we can use a piece of software to make medical visualisations that help doctors to make a better diagnosis. How we can use a piece of software to analyse data obtained during research such that we can learn more about, let's say, teachers' self-efficacy towards programming education.

During the experiment, I met teachers throughout the country. I would like to thank all of them for their participation, enthusiasm, and openness. Thanks, Felienne and Caroline for guiding me through the process of this thesis. Caroline, your critical view and asking pointed questions helped me to push my thinking process further. Felienne, besides your experience and enthusiasm I want to thank you for the opportunities you provide to students, like myself. It is a great experience to be part of a research group with such an open atmosphere. I am looking forward to be a part of this group for the coming years. Martin, thank you for providing an education perspective on this thesis. Willem-Paul, thanks for stepping in this committee such that I can actually graduate and giving tips I will benefit from during my PhD research. Maartje, thank you for sharing your expertise in statistics. I would also like to thank all the members of PERL, who are all more than happy to share their knowledge and give advice.

Especially Alaaeddin, it has been a great experience to work on a paper together. Last but definitely not least, I would like to thank family and friends that supported me during the process of my thesis. In particular, thank you Sander for your support. You are amazing.

Shirley de Wit
Delft, the Netherlands
October 24, 2019

Contents

Preface **iii**

Contents **v**

1 Introduction **1**
1.1 Problem statement and aim 1
1.2 Research questions 3
1.3 Approach and method 3
1.4 Relevance 4
1.5 Structure of the report 5

2 Context: Teaching **7**
2.1 Learning theories 7
2.2 Teaching methods 8
2.3 Teaching methods compared 10

3 Context: Programming education **13**
3.1 What is programming? 13
3.2 Scratch 14

4 Self-efficacy and adaptive expertise **17**
4.1 Self-efficacy 17
4.2 Adaptive expertise 18
4.3 Link between self-efficacy and adaptive expertise 21

5 Questionnaire: Teaching methods and self-efficacy **25**
5.1 Method 25
5.2 Participants 29
5.3 Results 29
5.4 Interpretation of results 35

6 Experiment: Teaching with direct instruction or discovery learning **37**
6.1 Method 37
6.2 Participants 39

6.3 Results 40
6.4 Interpretation of results 47

7 Conclusion **49**

8 Discussion **51**
8.1 Self-efficacy and adaptive expertise 51
8.2 Questionnaire 53
8.3 Experiment 56
8.4 Contributions 58
8.5 Future research 58
8.6 Working in the domains of Computer Science and Science Communication 59

Bibliography **61**

A Literature search **67**

B Experiment **69**
B.1 Method - Questionnaire 69
B.2 Results 75

C Questionnaire **85**
C.1 Educational material 85
C.2 Method 142

Chapter 1

Introduction

Across the entire world, there is an increased focus on the introduction of programming in primary schools [23, 27, 30, 39]. By programming we mean thinking about the steps necessary to solve a problem as well as writing those steps down in such a way that the computer can understand them. Learning to program stimulates students' critical thinking, problem solving skills, creativity, and ability to learn from mistakes [17, 37]. Knowledge of programming also benefits students in terms of job opportunities [37]. Moreover, having knowledge of programming contributes to an understanding of how the world works [37]. For example, social media such as Twitter or Instagram makes use of algorithms that determine what information is shown to you. Having some basic knowledge of programming helps you think about such algorithms. Teaching programming to primary school students involves girls and immigrants who are less likely to realise that they might like programming due to stereotyping [57]. In this way, programming education might increase diversity.

Despite the benefits, only 32% of the Dutch primary schools do something[1] with programming [31]. On 86% of those schools, programming is taught by the groups' teacher[2]. Reasons for schools to not be involved in programming include the lack of professional development and the need for more and better lesson content[3] [31]. For teachers to teach programming, they do not only need knowledge and skills but they also need a positive attitude towards the topic [39]. Unfortunately, programming is seen as too difficult to learn and teach [30]. In addition to this, over 80% of the Dutch primary school teachers are women [56], who have a more negative attitude towards programming than men [8].

1.1 Problem statement and aim

Only a relatively small part of the Dutch primary school teachers are involved in programming education. This might be explained by Dutch primary school teachers not believing in their teaching abilities when it comes to programming education, something which is found for other countries such as in [39]. Believing in your own abilities

[1] Examples being participation in events (like CodeWeek), (guest) lessons with or without computer, teaching programming as part of a project, etc.

[2] NL: groepsleerkracht

[3] NL: lesinhoud

towards a specific task is also known as *self-efficacy*. In other words, primary school teachers might have a low self-efficacy towards programming.

The self-efficacy of teachers affects the effort they invest in teaching, the goals they set, their level of aspiration, their openness to new ideas, and their willingness to experiment with new teaching methods [58]. A low self-efficacy of primary school teachers towards programming education implies that little effort is spent on this topic and that they are less open to it. Teachers' self-efficacy also relates to students' learning, self-efficacy beliefs, engagement, motivation and achievement [39, 58]. Research on teachers' self-efficacy in Computer Science and classroom practice is emerging [48] but currently still limited.

We hypothesize that Dutch primary school teachers have a low self-efficacy towards programming education which is (partially) caused by teachers not believing in their ability to use the instructional skills that are commonly used in programming education. We think that teachers do not believe in the presumed required instructional skill since teachers typically use different teaching methods in class than they (assume to) need to teach programming. Primary school teachers often teach via teacher-centered methods, such as direct instruction[4] [10, 49]. Lessons are goal-directed and the learning environment is structured by the teacher [4]. The teacher explains and demonstrates new knowledge and skills, checks for understanding, and guides students during practice before they work independently [24]. However, programming at primary schools is often taught via student-centered methods [26], such as discovery learning[5]. This is caused under the influence of Papert, founder of the idea that young children should program [57], who stressed the importance of discovery by having children work within an environment with no curricular objectives [44]. Discovery learning emphasises students learning concepts and principles through their own explorations and solving problems on their own [4]. The teacher mainly plays a role in creating situations where children can learn and develop. They do so by performing activities such as providing help and giving feedback [42]. Little research has been performed on how teaching methods affect teachers' self-efficacy in the field of programming education.

When teachers make use of a familiar teaching method, they can focus more on the programming context. Being able to use previous experience frees attention bandwidth and enables people to concentrate on other aspects of a new situation [52]. People who are able to perform well in an unfamiliar situation, by using existing knowledge and skills, have adaptive expertise [11]. Performing well might make it more likely that teachers experience their programming lesson as a success. Perceiving an experience as successful positively influences the self-efficacy [5]. Hence, we wonder whether adaptive expertise can give us insights into the development of teachers' self-efficacy.

This thesis focuses on the problem that primary school teachers have a low self-efficacy towards programming education which contributes to the little amount of programming education in Dutch primary schools. **The aim of this research is to investigate the effect of teaching methods on the self-efficacy of Dutch primary school teachers towards programming education**. Furthermore, we explore the concept of adaptive expertise within this context. With the research in this thesis, a better under-

[4]NL: directe instructie

[5]NL: ontdekkend leren

standing of primary school teachers' self-efficacy towards programming education is gained. This knowledge can be used to support primary school teachers better during professional development and by adapting teacher manuals for programming lessons.

1.2 Research questions

Based on the problem statement and aim we define the following research question:

RQ. *What is the effect of teaching methods on the self-efficacy of Dutch primary school teachers towards programming education?*

In order to answer this question, we first answer the following subquestions:

RQ1. How well known and used are direct instruction and discovery learning by Dutch primary school teachers?

RQ2. Which factors result in a difference in the self-efficacy of Dutch primary school teachers towards programming education?

RQ3. Which teaching practices result in a difference in the self-efficacy towards programming education of Dutch primary school teachers?

RQ4. How does the usage of a teaching method influence the self-efficacy of Dutch primary school teachers towards programming education?

In RQ2 we mention 'factors' by which we mean demographics and programming (teaching) experiences. The term 'teaching practices', which we use in RQ3, refers to which teaching methods teachers know, use and prefer in regular education and which method teachers expect to use when teaching programming.

This thesis is extended by research on adaptive expertise. The addition of this concept potentially gives us more insight into the (development of) teachers' self-efficacy. This results in our last subquestion:

RQ5. What is the relationship between adaptive expertise and self-efficacy of Dutch primary school teachers in the context of programming education?

1.3 Approach and method

The research questions will be answered through a mixed methods study consisting of a questionnaire and an experiment. The questionnaire is conducted within a quantitative research which allows us to obtain information from a large sample of teachers enabling us to draw conclusions about teachers' self-efficacy towards programming education that are generalisable. On the other hand, the experiment is conducted within a qualitative research which gives us insights into how teachers' self-efficacy changes depending on the teaching method used in programming lessons.

The quantitative research, in the form of a questionnaire, is conducted among Dutch primary school teachers to answer RQ1, RQ2, and RQ3. The questionnaire includes questions about knowledge, usage, preference, and expectations with regards

to teaching methods. Furthermore, the questionnaire measures the self-efficacy of the participants based on STEBI-NL [61]. The questionnaire is distributed via social media, an education fair, and mailings.

The qualitative research is used to answer RQ4 and RQ5. An experiment is set-up in which primary school teachers give four programming lessons while following either a direct instruction or discovery learning approach. Data is gathered on their self-efficacy before and after the experiment in the form of a questionnaire, based on STEBI-NL [61], and a semi-structured interview, based on OSTES [58]. The post-measurement is extended with questions related to adaptive expertise based on the Adaptive Expertise Inventory [11].

More details about the method are given in the remainder of this thesis, more specifically in Sections 5.1 and 6.1. A schematic overview of the thesis and its methods can be found in Table 1.1.

Table 1.1: Overview of thesis and methods

RQ	Approach	Method
1	Quantitative	Questionnaire
2	Quantitative	Questionnaire
3	Quantitative	Questionnaire
4	Qualitative	Experiment
5	Qualitative	Experiment

1.4 Relevance

1.4.1 Scientific relevance

There are many open questions relating to programming education. Little of the existing work focuses on primary schools and even less is known about how in-service teachers cope with programming education. We do know that self-efficacy is studied by many in the educational context [6, 28, 39, 45, 50, 58, 62] and that it is a predictor of teacher behaviour [59]. This thesis gives an insight in the self-efficacy of Dutch primary school teachers and its relation with teaching methods. Furthermore, to the best of our knowledge, there is no consensus in the current literature on how adaptive expertise and self-efficacy relate. With this thesis, we explore this relationship.

1.4.2 Societal relevance

By gaining an understanding of teachers' self-efficacy, we learn how we can support teachers in giving programming lessons. Our findings can be applied to teacher preparation programs, teachers' professional development as well as programming material. Supporting the teachers such that their self-efficacy towards programming education can increase, makes it more likely that more Dutch primary school teachers will teach programming. Their students will benefit from this because of programming stimulates students' skills, job opportunities, understanding of how the world works, and diversity.

As part of the qualitative study, we will create four programming lessons, in both a direct instruction and discovery learning format. The lessons include an extensive

teacher manual. The lessons will be available online for free, enabling teachers to (re-)use the material.

1.5 Structure of the report

The remainder of this thesis is structured as follows. First background information is given about teaching as well as programming education in Chapter 2 and 3 respectively. The obtained knowledge helps in understanding the context in which this thesis is executed. Thereafter self-efficacy, adaptive expertise, and their (theoretical) relation are discussed in Chapter 4. Based on this chapter, we have chosen the instruments that are used to collect our data. The quantitative study is described in Chapter 5 and provides answers to RQ1, RQ2, and RQ3. Subsequently, RQ4 and RQ5 are researched by the qualitative study as can be found in Chapter 6. After the conclusion in Chapter 7, a discussion on the work and future research follows in Chapter 8.

Chapter 2

Context: Teaching

This chapter discusses different learning theories and teaching methods are discussed. Literature is found by recommendations of several experts in the field, snowball effect as well as searches on related terms. The definitions and components of teaching methods as described in this chapter are used in both quantitative and qualitative research.

Based on this information design decisions for both the questionnaire and the experiment are made such as how the teaching methods are defined and applied.

2.1 Learning theories

Evidence about learning, from a pedagogical point of view, is guided by contrasting and disputed theories from psychology, sociology, education and policy studies [13]. Political ideologies and social values create preferences towards a type of theory. Theorists and developers appending different views tend to not engage but rather ignore each other [13]. The lack of integrated reviews and methodologies hinder in seeing the stories in each others work [21]. Moreover, this results in many different terms and concepts used in research involving learning [13]. To put these different terms and concepts into perspective, different learning theories are described below. Besides the three main learning theories (behaviorism, cognitivism, and constructivism) [15], constructionism is discussed because of its influence on programming education.

2.1.1 Behaviorism

Behaviorism focuses on observable behaviors and how learning is the consequence of external events, such as reinforcement, conditioning, rewards, and punishments [4].

The educational implications of behaviorism include structuring instruction around the presentation of the target stimulus and the provision of opportunities for the learners to practice making the proper response. It is the job of the teacher to determine the cues and desired responses as well as training the students to connect cues and responses [15].

2.1.2 Cognitivism

Cognitivism looks inside the 'black box' of the mind to determine mental activity and structures as if we were dealing with a computer [34]. Because of the empha-

sis on mental structures, cognitive theories are usually considered more appropriate for explaining complex forms of learning such as reasoning, problem solving and information-processing [15]. Cognitive theories emphasize making knowledge meaningful and helping learners organize and relate new information to existing knowledge in memory.

According to cognitivism, a teacher must understand that individual learners can have different prior knowledge and they should determine the most effective way to connect new information with the existing knowledge. Cognitive strategies include the usage of analogies, metaphors, framing, outlining, and concept mapping [15].

2.1.3 Constructivism

Constructivism is considered to be a branch of cognitivism but distinguishes itself by stating that learning is creating meaning from experience as opposed to something you acquire [15].

Instructional methods and strategies assist learners in actively exploring complex topics while learning objectives are not pre-specified nor is instruction pre-designed. The teacher instructs the student on how to construct meaning, as well as how to effectively monitor, evaluate, and update those constructions. Furthermore, the teacher aligns and designs experiences for the learner so that authentic, relevant contexts can be experienced [15].

2.1.4 Constructionism

Similar to constructivism, constructionism states that meaning is created [33]. The meaning we make is affected by our social interpretation. Papert focuses on the variance of individual and the environment and states that learning occurs through interaction and reflection. Learners can create meaning by building artifacts.

Constructionism does not concern learning basic skills but about attitudes, open-mindedness and willingness to try things out [22, 29].

2.2 Teaching methods

Under the influence of the different learning theories, a variety of teaching methods has been developed. Teaching methods differ, among others, in the level and way of guidance offered to students. The amount of instructional guidance can range from no instructor guidance to complete instructor guidance with a continuum of instructional support between these two [25]. Discovery learning and direct instruction are on the extremes of the continuum and are the focus of this thesis.

2.2.1 Direct instruction

Direct instruction is a teacher-centered way of teaching. There is a clear learning outcome and a structured learning environment, both following from the influences of Behaviorism [4]. The Social Cognitive Theory from Bandura [7] influenced direct instruction leading to include a more precise presentation and demonstration of the desired learning behaviors [4]. With direct instruction, lessons provide students with

explanations of concepts and procedures and the lessons support their learning strategy [35]. Direct instruction is often used by primary school teachers [10, 49].

Hollingsworth and Ybarra apply direct instruction in their Explicit Direct Instruction (EDI) model [24]. According to EDI, a lesson consists of several components, being;

Learning objectives A statement, which must be clearly communicated to the students, describing what students will be able to do by the end of the lesson.

Activate prior knowledge Purposefully moving something connected to the new lesson from students' long-term memories into their working memories so they can build upon existing knowledge.

Concept development Teaching the concepts contained in the learning objective.

Skill development Teaching the steps or processes used to execute the skills in the learning objective.

Lesson importance Teaching why the content in the lesson is important to learn.

Guided practice Working on problems with students, step-by-step, while checking that they execute each step correctly.

Lesson closure Checking whether students learned concepts and skills in the learning objective by having them make some exercises.

Independent practice Letting the students practice what they were taught.

In addition to the listed lesson components, EDI provides delivery strategies that the teacher should follow that include checking for students' understanding, explaining, modeling, and demonstrating.

2.2.2 Discovery learning

Student-centered discovery learning is mostly influenced by Constructivist theories [4]. The teacher has a facilitating and guiding role and the teacher helps students constructing their own understandings and ideas. Within discovery learning, the teacher is the motivator to the learning process and not the source [42].

Discovery learning comes in many forms and with many names, there is a wide range of instructional conditions which are referred to as discovery learning [3]. Alfieri et al. [3] looked at a variety of literature related to discovery learning in order to find a definition of discovery learning. Based on the literature, they concluded that it is undetermined what exactly contributes to learning when referred to as discovery learning. Nevertheless, all the different discovery learning methods do expect the learner to discover the target information within the boundaries of given tasks and materials.

In elementary and middle schools, discovery learning is often used to teach programming in the form of tutorial-focused experiences as well as open exploration [26]. Tutorial-focused experiences aim at engaging learners to code themselves and often provide a structured hierarchy [32] while open exploration maximizes the freedom to explore. This form of learning is sometimes referred to as pure discovery learning [40].

Discovery learning is sometimes used by primary school teachers [49]. Teacher training is needed for teachers to work with discovery learning [51].

The learning by exploration is encouraged in visual programming environments designed for young people, such as Scratch [38, 41]. The developers of Scratch, MIT Media Lab[1], suggest to include the following structure in a one-hour programming workshop [53]:

Imagine Gather as a group to introduce the theme and spark ideas.

Create Help participants as they create story projects, working at their own pace.

Share At the end of the session, gather together to share and reflect.

2.3 Teaching methods compared

As mentioned before, elementary and middle school students often learn Computer Science by discovery learning [26]. However, the effectiveness of (pure) discovery learning is questioned [35, 40, 36].

Kirschner et al. [35] reviewed various studies and concluded that teaching methods that use minimal guidance, such as discovery learning, are less effective and less efficient than approaches that provide more guidance for novices to intermediate learners. They favor direct instructional guidance in which the teacher explains concepts, procedures and learning strategies. Critics of direct guidance argue that learners must be engaged in the processes of learning and that the passive role of the learner in direct guidance contexts may inhibit beneficial learner engagement [25].

Klahr and Nigam [36] compared direct instruction and open-ended discovery learning in the context of teaching scientific reasoning to primary school students. The students needed to reason about how two variables affected the distance a ball rolled after leaving a downhill ramp. All children received an explanation about the ramp itself. In the direct instruction condition, an instructor demonstrated several experiments and asked questions about the thoughts of the children while the children in the discovery learning condition designed their own experiment. The study shows that more students learned from direct instruction than from open-ended discovery learning.

Next to students' learning, the amount of instructional guidance can affect students' self-efficacy as well. Hushman and Marley [25] compared direct instruction (consisting of a lecture and examples), guided instruction (consisting of examples and student-generated explanations) and minimal instruction (consisting of student directed discovery). All three types of instruction had a positive effect on students' self-efficacy. However, guided instruction had a greater impact than direct or minimal instruction.

We think that students' learning and self-efficacy affect the self-efficacy of teachers. When students show good performance and belief in their own abilities, we believe that a teacher is more likely to perceive a situation as successful which positively affects teachers' self-efficacy. Since students' learning benefits more from direct instruction, we think that the usage of direct instruction will more likely result in a positive experience and therefore have a bigger effect on teachers' self-efficacy.

[1] `https://www.media.mit.edu/projects/scratch/`

However, it should be noted that teachers that already have a high self-efficacy are more interested in learning about student-centered methods while low self-efficacy teachers are more focused on teacher-centered methods [54]. For this reason, we wonder whether the effect of a teaching method differs depending on the initial level of self-efficacy.

Chapter 3

Context: Programming education

Within this chapter, some background information is given about programming education. Furthermore, the programming language Scratch is discussed. Scratch is the programming language used within the experiment as described in Chapter 6.

3.1 What is programming?

Programming involves, but is not limited to, the writing of software. For a computer to understand what to do, a programmer needs to formulate tasks in a, for a computer, logical way via a programming language. The programmer needs to comply with the rules and commands of the programming language. Many different programming languages exist and are designed for different purposes.

The first programming language which is specifically designed for children is the Logo programming language [64]. This language has a major influence on both the development of other programming languages aimed at children as well as that the teaching methods used since the language is designed from a constructionism perspective. After the introduction of Logo in the 70's programming became popular at schools which resulted that in the 1980s many schools had programming classes once a week [30]. In the mid-1990's schools these classes disappeared. Programming was labeled as too difficult to teach and learn, and there were questions about the purpose. There was a lack of subject-matter integration as well as a lack of qualified instructors [30]. Meanwhile, there is a retained attention on programming education. In the Netherlands, this is partly caused by SLO[1] defining 21st-century skills which include computational thinking. Programming is one of the ways of implementing computational thinking in class.

Research on programming education at primary schools is upcoming but still limited [23]. Research that does focus on programming education within the primary school setting include topics such as programming concepts and misconceptions[2, 19, 55], subject integration [23], attitudes towards programming [1, 8], and teacher training [9, 23].

[1] 'Stichting Leerplanontwikkeling' focuses on the development of curricula in Dutch primary and secondary education

3.2 Scratch

Some of the popular languages include Java, C, and Python [46]. They are all textual languages, meaning they all are used by typing commands. However, block-based languages are powerful when learning to program [2]. In these languages, the programmer can use predefined blocks and put them together to form a program. Block-based languages that are aimed at novice programmers include Alice, Blockly, App Inventor and Scratch [2] Scratch is used within the qualitative research as described in Chapter6. Scratch[2] is developed by MIT Media Lab[3], and used by many [2, 43, 38]. A screenshot of the Scratch environment is shown in Figure 3.1.

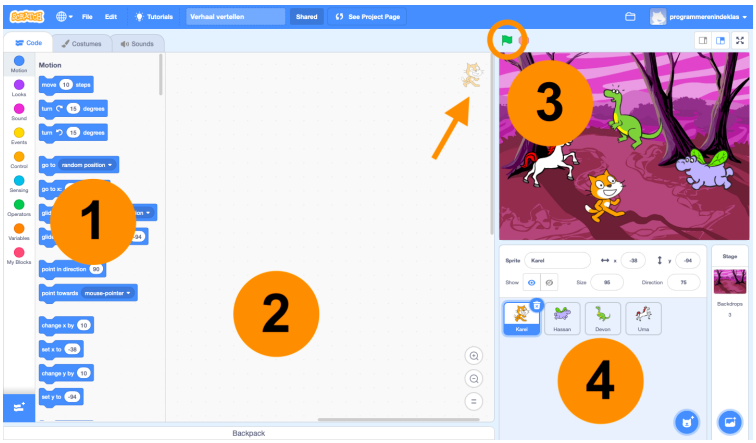


Figure 3.1: *Screenshot of the Scratch environment with 1) toolbox 2) programming area 3) result 4) sprites*

The blocks that can be used are found in the toolbox (1) where different colors indicate different categories of blocks. In total there are 119 Scratch blocks when no extensions are used[65]. By dragging blocks into this gray area (2), you can build your program. Blocks should be connected to each other to be executed sequentially. A sequence of connected blocks is referred to as a script. The arrow in Figure 3.1 indicates which character, referred to as sprite in Scratch, you are programming. The visual output of your program can be found in the right corner (3). In this area you can also find the green flag, which is the button regularly used to start your program. On the bottom right (4) the different sprites that are in your program are shown.

Although programming languages can differ in many ways, some also show similarities. Below four different programming concepts are discussed. These concepts can be found in both textual and block-based languages. For each of the concepts an example program in Scratch is provided. Note that these concepts are not required to be present in all programming languages, but that is outside the scope of this thesis.

Sequence Blocks are executed from top to bottom. It is possible to have multiple scripts running concurrent. For example in Figure 3.2 we see two scripts. When

[2] https://scratch.mit.edu
[3] https://www.media.mit.edu/projects/scratch/

we press the green flag, they both start running. The sprite which is programmed will say ‘Hello‘ for two seconds after which it will say ‘Nice to see you here‘. Meanwhile, the sprite will take ten steps, wait one second and take another ten steps. So the sprite will both talk and walk when pressing the green flag.

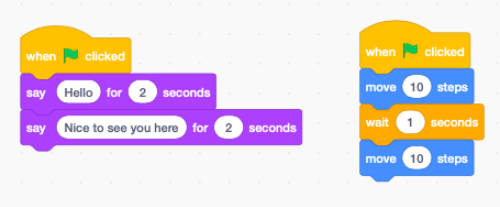


Figure 3.2: *Scratch program that makes the sprite talk and walk*

Variable With the use of a variable a value can be saved in the program. A variable in Scratch can hold either a single number or text. A special type of variable in Scratch are lists. In a list multiple numbers or texts can be added. For instance, if we are building a game we want to save the amount of points the player has currently scored. At the beginning of our program, as can be seen in Figure 3.3, we save the value of zero in our variable named ‘points’. Every time the player clicks at the sprite, the value of one is added to the current value of ‘points’. In Figure 3.4 a program is shown which uses a list. We can add different kinds of food to a list and let our sprite say the first thing that is on or list.

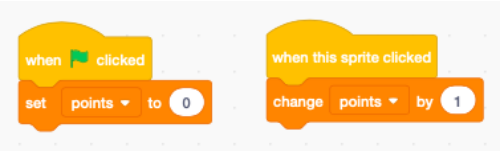


Figure 3.3: *Scratch program that sets and changes the variable ‘points’*

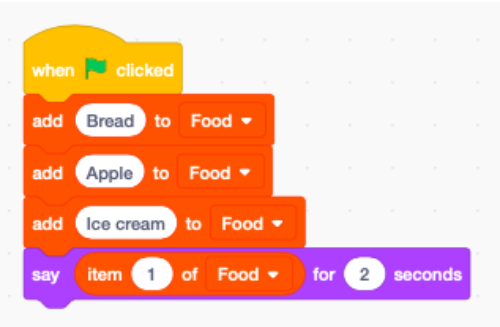


Figure 3.4: *Scratch program that saves text in a list and let the sprite talk*

Loop Repetition can be created in Scratch by the usage of repeat and forever blocks. A program that uses both these blocks is shown in Figure 3.5. In this program

the sprite says ‘Hello’ two times for two seconds (so four seconds in total) after which the sprite says ‘Bye’ for another two seconds. Meanwhile, the sprite is turning fifteen degrees after every second. The process of spinning goes on forever unless the program is stopped manually.

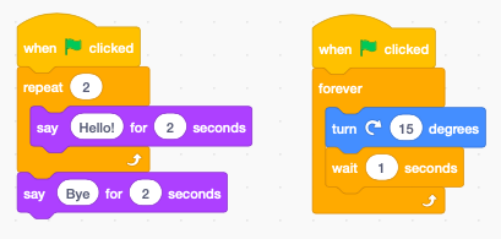


Figure 3.5: *Scratch program in which the sprite talks a bit and spins forever*

Conditions A program can adapt its behaviour depending on the situation it is in. For instance, we can adapt the text a sprite is saying depending on the time of the day. In Figure 3.6 a program is shown in which the sprite says ‘Have a nice day!’ until five p.m. and otherwise says ‘Have a nice evening!’.

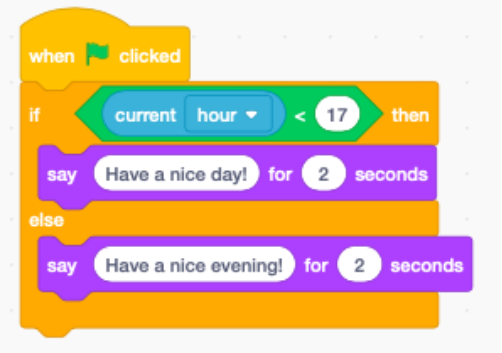


Figure 3.6: *Scratch program in which the sprite says something depending on the time of the day*

Chapter 4

Self-efficacy and adaptive expertise

In this chapter, an overview of both the concept of self-efficacy and adaptive expertise is given. Furthermore, the relation between the two is explored. Related literature is found by searching via search engines as well as via snowballing. A more detailed overview of the literature search is given in Appendix A. The knowledge and insights gained within this chapter contributes to the design decisions made regarding the method and the interpretation of the results in both the qualitative and quantitative research.

4.1 Self-efficacy

Self-efficacy is the strength of someone's belief in their own ability to master or complete a task [5] in a specific domain- and context [62]. When having a higher level of self-efficacy, one will make more effort and carry through longer with a specific task [5]. Bandura [5] distinguished efficacy expectations from outcome expectancy. Efficacy expectation relates to the belief that one can execute behavior, while outcome expectancy relates to whether one believes that a given behavior will lead to certain outcomes.

Ones level of self-efficacy is affected by the following sources [5];

Mastery experience A successful experience can increase ones self-efficacy while repeated failure has the opposite effect. The more the circumstances vary in which success is experienced, the more likely it is that an increase in self-efficacy occurs. Mastery experience is considered to be the most influential source of ones self-efficacy.

Vicarious experience Seeing others, also referred to as models, perform tasks successfully can create the expectation within observers that they should be able to do so as well. The strength of this effect depends on the models' characteristics, the similarity between model and observers, the difficulty of the tasks, the situation in which the task is performed, and the diversity of modeled skills.

Verbal persuasion Verbal persuasion, also referred to as social persuasion [6], can impact self-efficacy but works best in combination with one of the sources above. The more believable the persuade (think of perceived credibility, thrust worthiness, and expertise) the more likely self-efficacy is to change.

Emotional arousal The physiological state of someone influences their anxiety and stress level towards a certain task, which both negatively influence ones self-efficacy.

It should be noted that although most research concerning teachers' self-efficacy is built upon the work of Bandura [5]. The first measures of teacher efficacy were grounded in Rotter's social learning theory. In this perspective, teacher efficacy focused on whether teachers believe that their abilities have a bigger impact on students learning than the environment [58].

Self-efficacy is used by many in an educational context [6, 28, 39, 45, 50, 58, 62]. Teaching self-efficacy is associated with student outcomes [58], behavior in the classroom [59], implementation of instructional change [59] and teachers' personality and beliefs [28]. Jamil et al. [28] found that teachers who have more constructivist beliefs about how children learn tend to have higher self-efficacy. One of the reasons for this could be that these teachers see children as partners in the creation of knowledge and are therefore less likely to consider children's difficulties as their own personal failures [28]. Teachers' self-efficacy may vary from subject to subject [62], so a teacher with a high self-efficacy for mathematics or linguistics might not have the same self-efficacy for teaching programming. When learning a new teaching instruction or strategy, mastery experiences through follow-up coaching have the strongest effect on teacher self-efficacy beliefs [59].

The most popular instrument to measure teachers' self-efficacy is the teacher efficacy scale by Gibson and Dembo [58]. Within this instrument, both outcome expectancy and personal teaching efficacy are measured. However, determining the level of specificity has been a problem for this, and other, efficacy instruments [58]. Therefore researchers have adjusted the teacher efficacy scale, including Riggs and Enochs [47]. They developed the Science Teaching Efficacy Belief Instrument (STEBI) which includes two factors being personal science teaching efficacy (PSTE) and science teaching outcome expectancy (STOE). The STEBI is used across several studies [58, 62]. Velthuis et al. [61] translated the STEBI to Dutch resulting in the STEBI-NL. They analysed the instrument and concluded that the PSTE factor showed no major problems in reliability. However, the STOE statements showed low reliability scores. Building upon existing instruments, Tschannen-Moran et al. [58] developed (and validated) the Ohio State Teacher Efficacy Scale (OSTE). This instrument measures teacher efficacy using three factors being instructional strategies, classroom management, and student engagement. The developers of the instrument state that the instrument 'has a unified and stable factor structure and assesses a broad range of capabilities that teachers consider important to good teaching'.

We combined the work of Bandura [5] and Tschannen-Moran et al. [58] which resulted in the model as depicted in Figure 4.1. The four terms on the outside are the four sources affecting self-efficacy according to [5]. The three components within the circle compose teacher efficacy according to [58].

4.2 Adaptive expertise

Adaptive expertise is the ability to perform well in a non-standard situation [11] in which the task, method or desired result is not known in advance [14]. Hatano & Ina-

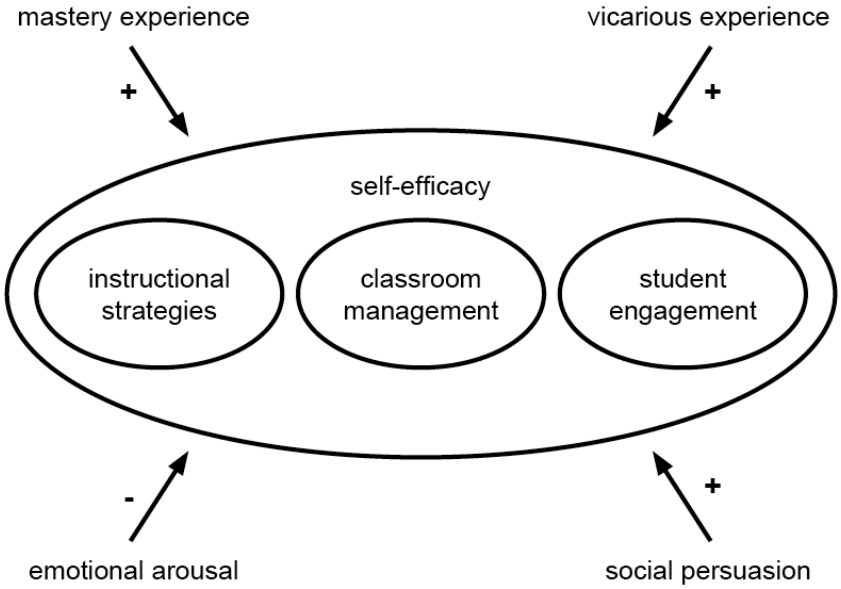


Figure 4.1: *Self-efficacy in the educational context for which we combined the outside factors affecting self-efficacy [5] and within the circle components that compose teacher efficacy [58]*

gaki [20] distinguished routine expertise from adaptive expertise, which is followed by many others including [12, 16, 18]. Routine experts are outstanding in terms of speed, accuracy, and automatising of performance. On the other hand, they lack flexibility and adaptability to new problems [20]. In contrast to routine expertise, individuals with adaptive expertise possess the knowledge of why and under which conditions certain methods have to be used or new methods have to be devised [12]. They look for opportunities for new learning in their field of expertise, successfully monitor their understanding and conceive knowledge as something dynamic rather than something static [63]. An outcome of having adaptive expertise is the creation of new knowledge or methods. [11]. According to the literature, adaptive expertise has three components: (1) domain-specific skills, (2) meta-cognitive skills, and (3) innovative skills [11]. Combining these components with the work of Carbonell et al. [12] suggest that domain-specific skills are also required for routine experts. They also suggest that metacognitive skills are better developed for adaptive experts. Furthermore, they suggest innovation skills include flexibility, ability to innovate, continuous learning, seeking out challenges, and creativity. Hatano & Inagaki [20] describe three factors contributing to the development of adaptive expertise. The first factor is the variation in the situation since a varying situation motives someone to adjust their skills. The effect of this adjustment on the outcome can be observed and thereby contribute to ones adaptive expertise. Secondly, when a situation is not important people are more likely to vary their approaches and examine their effects. The third factor relates to the culture in which the situation occurs. In some cultures understanding is more valued than the outcome which results in people being encouraged in active experimentation.

Schwartz et al. [52] state that adaptive experts score high in both innovation and

efficiency, which are two dimensions of their learning and performance 'space'. People who are high on efficiency can rapidly retrieve and accurately apply appropriate knowledge and skills to solve a problem. Within this space, there is a hypothetical optimal adaptability corridor, or OAC, for the development of adaptive expertise. Its function is to help ensure that innovation and efficiency develop together [52] since this yields in better adaptability in the short run and better efficiency in the long run.

An overview of adaptive expertise is shown in Figure 4.2, in which we combine the three factors that affect adaptive expertise [20] with the components of adaptive expertise [11].

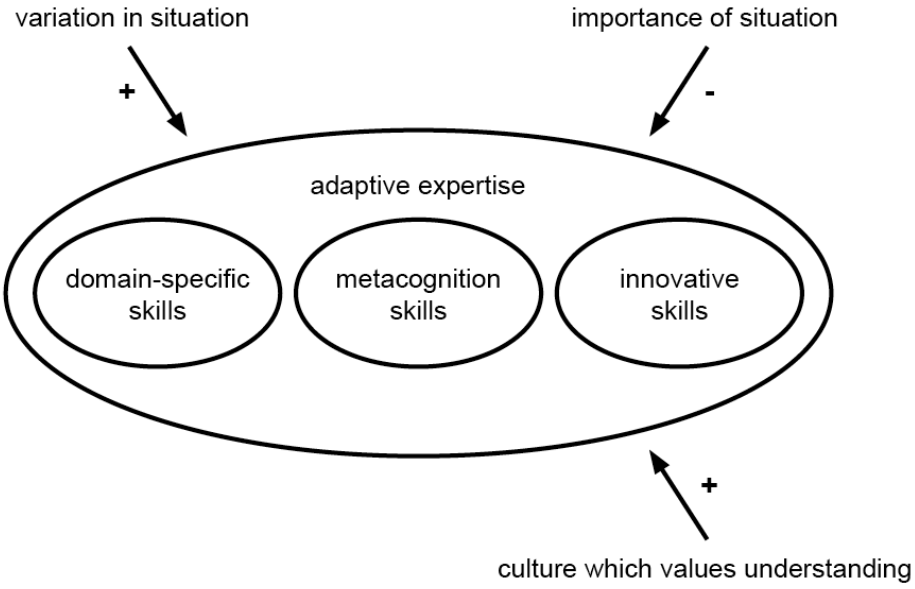


Figure 4.2: *Adaptive expertise for which we combined the outside factors affecting adaptive expertise [20] and within the circle components that compose adaptive expertise [11]*

Some of the research on adaptive expertise is conducted in the educational context [12]. Research on adaptive expertise is important for the educational field as it will provide further insights on how to design learning environments and tasks that support the growth of students' adaptive expertise [12]. By having a higher level of adaptive expertise, students are better prepared to function in complex environments [16].

Different instructional methods were analysed for their impact on adaptive expertise. The common trend in these methods is that students are stimulated to explore and discover [12]. The different instructional methods demonstrate that adaptive expertise benefits from training situations in which participants are responsible for building their knowledge, as in unguided exploration or other forms of active learning styles. In such learning formats, there are numerous possibilities for making mistakes, which further benefits adaptive expertise when a link is made between the mistakes and the to-be-learned knowledge. Establishing this link leads to a deeper understanding of the domain, resulting in a knowledge representation beneficial for adaptive expertise [12].

A significant challenge to researchers interested in adaptive expertise is establishing valid and reliable ways of capturing and representing what people know, how they apply their skills, and how their performance varies over time and across problems [63]. The usage of mixed methods can help in understanding both the performance and perception of adaptive expertise [16]. Ferguson et al. [16] introduced an Adaptive Expertise survey in which they theorised four sub-scales addressing innovative skills, domain skills, metacognition, as well as self-efficacy and resilience. Carbonell et al. [11] included only two of these factors, being domain-specific and innovative skills, in their Adaptive Expertise Inventory. The reliability of the Adaptive Expertise Inventory ranges from acceptable to good.

4.3 Link between self-efficacy and adaptive expertise

Some literature discusses both self-efficacy and adaptive expertise. Carbonell et al. [12] concluded, based on a literature review, that there seems to be a moderately positive correlation between self-efficacy and adaptive expertise. Walker et al. [63] found a positive link between advanced students' innovation, which relates to adaptive expertise, and confidence, which relates to self-efficacy, but no such links were found for beginning students. Mannila et al. [39] measured both teachers' self-efficacy and the competence 'I could adapt my ways of working based on new digital tools'. Respondents with a low self-efficacy rated themselves low on this particular competence.

There seems to be a relation between self-efficacy and adaptive expertise, however, to the best of our knowledge there is no consensus on how these two interact. Some of the literature indicates that self-efficacy is a part of adaptive expertise [16, 63], while other studies suggest that self-efficacy is needed in order to develop adaptive expertise [60].

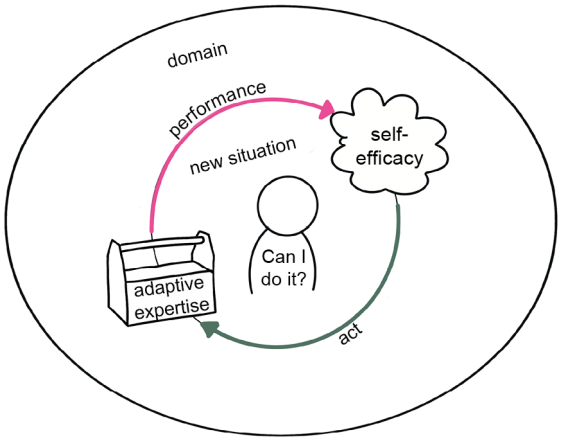


Figure 4.3: *Self-efficacy and adaptive expertise in a new situation*

We think that self-efficacy and adaptive expertise reinforce each other. When a new situation occurs, as displayed in Figure 4.3, you will ask yourself whether you can act in the new situation. When you believe that your abilities are sufficient, meaning a

higher self-efficacy, you will be more likely to act in this new situation. By interacting with this new situation, you will learn about whether your approach failed or succeeded and thus this experience will contribute to the development of adaptive expertise. The better your adaptive expertise, the more likely it is that you will have the abilities needed to perform successfully. The successful performance will in its turn increase self-efficacy.

We also believe that, next to the direct relation, the concepts are linked by the factors in the environment. The links we see are illustrated in Figure 4.4. Being exposed to different variations of a situation affects adaptive expertise but also self-efficacy. Experiencing variations of situations strengthens the effect of mastery experiences and vicarious experiences. These experiences, in turn, affect the self-efficacy. When a task is not perceived as important, it will stimulate risk taking behaviour and therefor stimulate the development of adaptive expertise. When a task situation is important, it can increase the tension a person experiences (emotional arousal) and thereby decrease the self-efficacy. The persuasion someone receives influences the self-efficacy and, depending on the emphasis of the persuasion, influences whether understanding or outcome is valued.

4.3.1 Self-efficacy, adaptive expertise, and teaching methods

Since this thesis aims at understanding the effect of teaching methods on self-efficacy, we discuss the role of teaching methods within Figure 4.4. Furthermore, we describe the factors we measure within both the quantitative and qualitative research.

The teaching method used seems to have a direct influence on the self-efficacy of teachers since different teaching methods imply different instructional strategies. When not being familiar with a teaching method, the teacher is less likely to be confident with the corresponding instructional strategies. We think this since the teacher has no experience on which self-efficacy can be build. Moreover, teachers might also have more negative feelings (emotional arousal) towards a teaching method they are unfamiliar with. Furthermore, the teaching method used can influence how the teachers experience success, as discussed in Section 2.3.

Aspects measured in quantitative research

Within the quantitative research, in the form of a questionnaire, we focus on the self-efficacy of Dutch primary school teachers, as depicted in Figure 4.5. We use the STEBI-NL [61] to measure teachers' self-efficacy. When we look at the STOE statements from the STEBI-NL we see that they are about the relationship between student performance/understanding and 1) the teachers' effort, 2) the programming curriculum, 3) the competence of the teacher, 4) the teacher in general. One of the statements is about students' interest and competence of the teacher. The PSTE statements are mainly about the beliefs that teachers have towards their own performance and instructional skills and knowledge. Next to the STEBI-NL, we ask teachers about their current programming experiences to get an insight into their (mastery or vicarious) experiences.

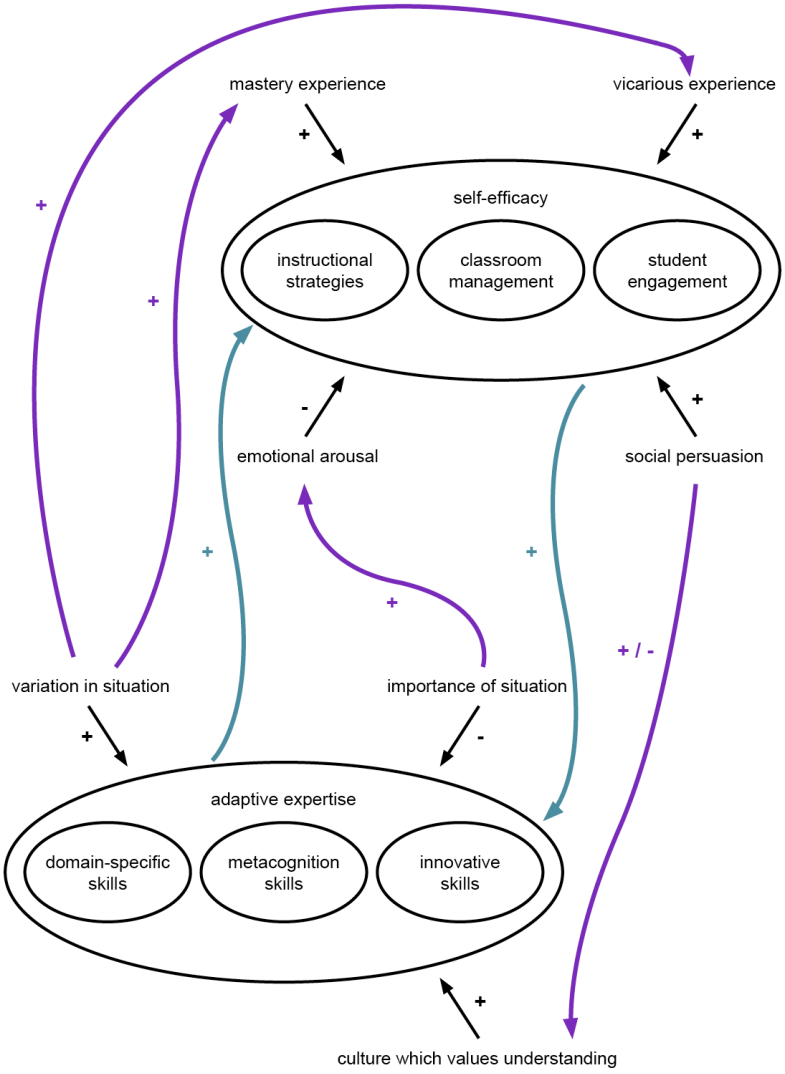


Figure 4.4: *The concept of self-efficacy and adaptive expertise and the interaction between them and related factors. The black arrows indicate a direct influence on the concepts. The green arrows indicate a direct relation between the concept. The purple arrows indicate the influence on one related factor on another.*

Aspects measured in qualitative research

Within the qualitative research, in the form of an experiment, we again use the STEBI-NL [61] but expend the research with questions about adaptive expertise. We do so by including a translation of the Adaptive Expertise Inventory [11]. When we reviewed the statements of the Adaptive Expertise Inventory we identify statements about innovative skills and domain-specific skills. It seems that the ability to use your adaptive expertise is included in the domain skill statements. Furthermore, we let teachers reflect on their adaptive expertise with the usage of the innovation-efficacy space [52].

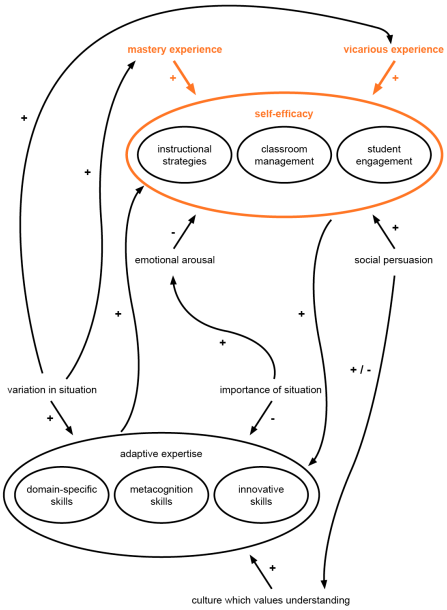


Figure 4.5: *The orange elements within this model are measured in the quantitative research (questionnaire)*

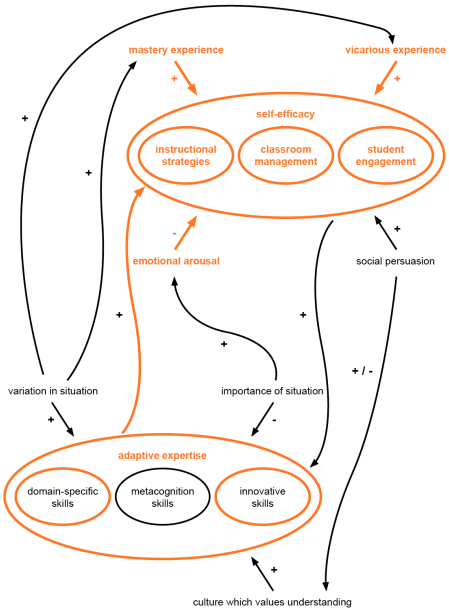


Figure 4.6: *The orange elements within this model are measured in the qualitative research (experiment)*

Chapter 5

Questionnaire: Teaching methods and self-efficacy

The results within this chapter help us answering the main research question *What is the effect of teaching methods on the self-efficacy of Dutch primary school teachers towards programming education?*

We think that Dutch primary school teachers are more familiar with direct instruction than with discovery learning. Furthermore, we expect that the self-efficacy of Dutch primary school teachers towards programming education while differ depending on factors such as gender and experience with programming education. Next to these factors, we hypothesize that the self-efficacy of teachers differs when teachers have different teaching practices such as usage and preference of teaching methods. These assumptions are studied by answering the following research questions;

RQ1. How well known and used are direct instruction and discovery learning by Dutch primary school teachers?

RQ2. Which factors result in a difference in the self-efficacy of Dutch primary school teachers towards programming education?

RQ3. Which teaching practices result in a difference in the self-efficacy towards programming education of Dutch primary school teachers?

5.1 Method

Data is collected via an online questionnaire which consists of 17 questions. The first four questions are about teaching preferences which do not specify to programming education. These questions are followed by 12 questions that are about programming education. The last of these 12 questions include all of the 24 adapted STEBI-NL statements. The last five questions ask for participants' demographics. How the data answers the RQ's is described in Section 5.1.2, Section 5.1.3 and Section 5.1.4. The full questionnaire can be found in Appendix B.

The questions about teaching methods include three teaching methods being direct instruction, inquiry learning, and discovery learning. Only the answers regarding direct instruction and discovery learning are analysed since these are the focus of the thesis.

The questionnaire is distributed online via social media and via a mailing list with over 6,000 email addresses from Dutch primary schools[1]. Flyers of the research were also displayed at the Nationale Onderwijs Tentoonstelling (NOT)[2], which is a Dutch education fair with around 36,000 visitors. Eventually, data of 259 participants is analysed.

5.1.1 Preprocessing

Before the analysis, the data is preprocessed to improve the quality of the data. The following changes are made:

Age is extracted from an open question and is modified such that only the numeric values here contained. For example, some participants wrote '32 jaar' which is converted to '32'. Furthermore, age is categorised into the following categories: under 25, 25-34, 35-44, 45-34, over 55, unknown.

Grades are extracted from an open question and are manually parsed into specific grades. For example, some participants indicated that they teach 'groep 3-4-5' which corresponds to the individual grades 3, 4, and 5. Furthermore, the grades are categorised in low, middle and high grades. This division comes from the Dutch primary school system. Low grades include grades 1 and 2, middle include grades 3, 4 and 5, and high grades include grades 6, 7 and 8. When going through the answers we came across a participant who indicated teaching the first-year high school students. This data entry has been removed.

Work experience includes a category 'other'. When evaluating these answers we found that all but one of the participants indicating 'other' had many years of teaching experience. These participants are re-classified in the 'over 10 years experience' category.

School type include the categories 'public schools' and 'religion schools'. However, they differ mainly in the source of norms and values. Therefore these two categories are merged into the 'public' category.

Programming experience teachers have is classified in 'textual languages', 'hardware', 'Scratch', and 'other'.

Programming teaching experience is a multiple choice question where two of the participants selected more than one answer. One of the participants indicated both to have no experience with teaching programming as well as having observed a (guest) lesson. Only the second answer is preserved. The other participant selected an answer indicating that (s)he taught programming lessons as well as the answer about observing a (guest)lesson. Since mastery experience has a bigger influence on ones self-efficacy than vicarious experience, the answer related to observing a lesson has been removed.

[1] Mail addresses collected via `https://instellingsinformatie.duo.nl/public/websitecfi/`
[2] https://www.not-online.nl

Preference teaching method includes a category 'other' which many have used to indicate that they prefer a combination of teaching methods. When there was an explicit mention of the words 'combination' or 'mix' the 'other' category is interpreted as a preference for all three methods. Some teachers also responded with explicit mentioning of one or more teaching methods in the 'other' category which is reclassified to the specific method(s).

Expected teaching method for programming lesson shows a similar pattern as the category 'preference teaching method'. Therefore the same strategy is applied here.

To deal with sparse data while retaining the maximum amount of responses, different subsets are created. The inclusion of participants is based on the following criteria;

Criterion 1 Respondents answered whether they know and use both direct instruction and discovery learning

Criterion 2 Respondents answered all statements related to self-efficacy

The demographics of participants are analysed by including all respondents that meet either one of the described criteria. Data collected by the respondents fulfilling the first criterion is used to answer RQ1, while RQ2 is answered by data collected from respondents meeting criterion 2. For RQ3 data is used from respondents that meet both criterion. An overview of the datasets, size, and usage is found in Table 5.1

Table 5.1: Overview of the different datasets

Dataset	RQ	Size	Criteria	Paragraph
Participants	-	259	1 or 2	5.2
Teaching methods	1	238	1	5.3.1
Self-efficacy	2	208	2	5.3.2
Both	3	187	1 and 2	5.3.3

5.1.2 Familiarity with teaching methods

The familiarity of teachers with different teaching methods is measured by asking whether teachers know and/or use the method. To make sure the participants understand the methods as we mean them, definitions of the methods are provided within this questionnaire. Whether factors, such as demographics and experiences, relate to teachers knowing and/or using certain teaching methods is analysed with statistical tests. Frequencies per teaching method and factor are calculated and analysed one by one. In other words, we measure whether there is a relation between an individual factor and either using or knowing a single teaching method. In most situations, the chi-square test of independence is used. However, when over 20% of frequencies are smaller than 5, the Fisher's exact test is used. Another situation where an alternative test is used is when dealing with factors that are not mutually exclusive which means that one teacher can contribute to multiple frequencies. For example, some participants indicated to teach multiple grades. For these demographics, a simple logistic regression is used instead.

5.1.3 Self-efficacy

The self-efficacy is measured by adapting the questionnaire from Velthuis et al. [61], which is a Dutch translation of the STEBI. The instrument consists of 24 positively phrased statements which participants need to rate on a 5-point Likert scale. So participants need to indicate for each statement whether they 1) strongly disagree, 2) disagree, 3) neither agree nor disagree, 4) agree, or 5) strongly agree. The first 12 statements are about someone's science teaching outcome expectancy (STOE), while the other statements measure personal science teaching self-efficacy (PSTE). Both the STOE and PSTE scores are calculated by summing the scores of the related statements. Statements within the STEBI-NL are about science teaching. In this thesis, the statements are rephrased by replacing 'science' with 'programming'. This instrument is chosen because it is commonly used [58, 62] and the Dutch translation of the PSTE is already validated [61].

Before we analyse the data, we evaluate the adapted STEBI-NL instrument. To see whether the items are internally consistent, the Cronbach's alpha is measured. A high value for the Cronbach's alpha, up to 1, indicates that the items are closely related to each other. The Cronbach's alpha for all statements is 0.86 for the STOE statements, 0.63 and 0.96 for the PSTE statements. Thus the statements as a whole have a good internal consistency, the STOE statements have a questionable consistency while the PSTE statements have an excellent consistency[3]. This implies that the PSTE statements measure the same concepts while this is questionable for the STOE statements.

We also did a factor analysis, which is described in B.2.3.

Based on both the Chronbach's alpha and factor analysis we decided that the STOE factor, as measured with the adapted STEBI-NL, does not result in valid data and is therefore excluded from further analysis.

We test whether the PSTE scores differ depending on several factors, being demographics and programming (teaching) experience. To test whether PSTE scores differ for the mutually exclusive factors, one-way ANOVA is used except for age for which a t-test is sufficient. The relation between grades as well as school type and PSTE is analysed with the usage of one-way ANOVA by considering one grade or school type at a time. To interpret the findings better, the PSTE score is divided into low, middle and high PSTE. The thresholds for low, middle and high classification are determined by the 25th and 75th quartiles of all PSTE scores.

5.1.4 Teaching practices and self-efficacy

Participants are asked about their teaching practices which include the teaching method they know, use and prefer in their regular education as well as which method they expect to use when teaching programming. The difference in PSTE score depending on a single teaching practice is analysed by the usage of one-way ANOVA.

5.1.5 Pilot

A paper version of the questionnaire is conducted during a professional development training for primary school teachers. The pilot is filled out by ten participants (nine

[3] https://spsshandboek.nl/cronbachs-alpha/

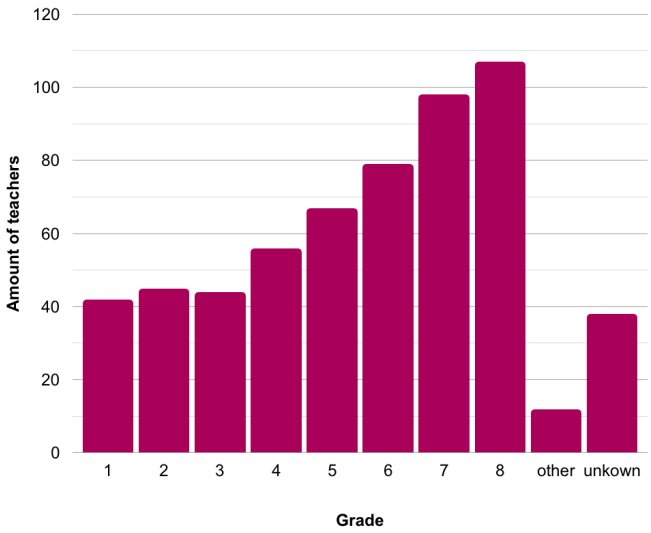


Figure 5.1: *Grades participants teach*

females, one male). While answering the multiple choice questions a relative high number of participants crossed more than one answer for at least one question. Therefor respondents can pick more than one answer in the online questionnaire as well. One of the participants mentioned not to be able to fill in the self-efficacy statements. Because the other nine were able to score the adjusted STEBI-NL statements and because the statements originated from a (partially) validated instrument nothing is changed to this part of the questionnaire. No questions or further comments were made.

5.2 Participants

The total amount of participants is 259, which includes 152 female teachers, 69 male teachers and 38 unknown. The age of the participants ranges from 21 to 64 with a mean of 39.8 and median of 37. The majority of the participants teach higher grades (147 or 57%), as can also be seen in Figure 5.1. Some respondents indicated to not teach at this moment or to teach gifted children[4]. Both respondents are included in the'other' category. Most of the participants teach a single or two different grades (144 or 56%).

A relatively high part of the participants are experienced teachers, with 76 (or 32%) having more than 10 years of teaching experience. The school type represented the most in the sample are public schools (199 or 77%).

5.3 Results

5.3.1 Familiarity with programming

The majority of the participants have experience in programming (170 or 66%) as depicted in Figure 5.3. Of those participants 138 or 85% report having experience in

[4]NL: plusklassen

Scratch. Also 171, or 66%, participants have experience in teaching programming. The amount of programming lessons they taught can be seen in Figure 5.3. Of these 171 teachers, 154 teachers indicated to have programming experience. Of the participants, 72% (or 187) has experience with either programming or teaching programming.

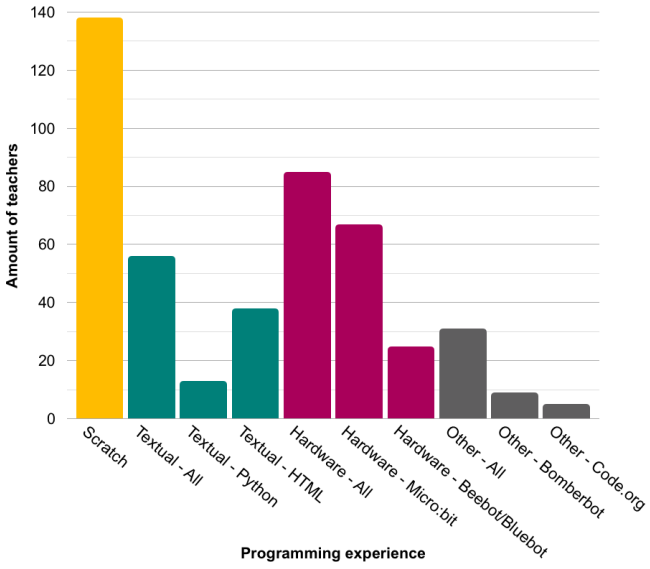


Figure 5.2: *Experience in teaching programming*

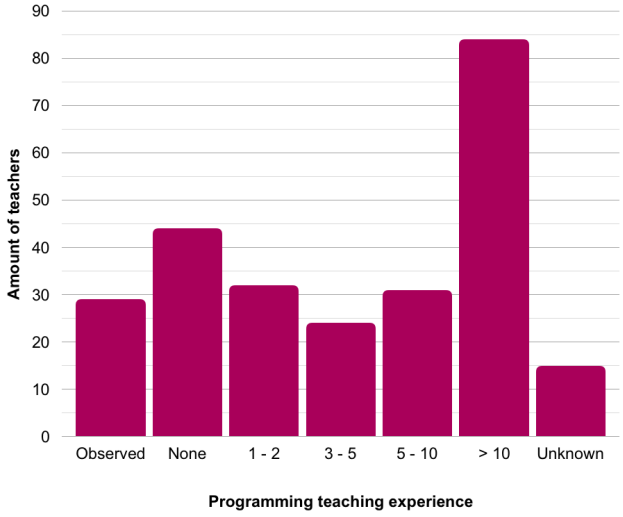


Figure 5.3: *Experience in teaching programming*

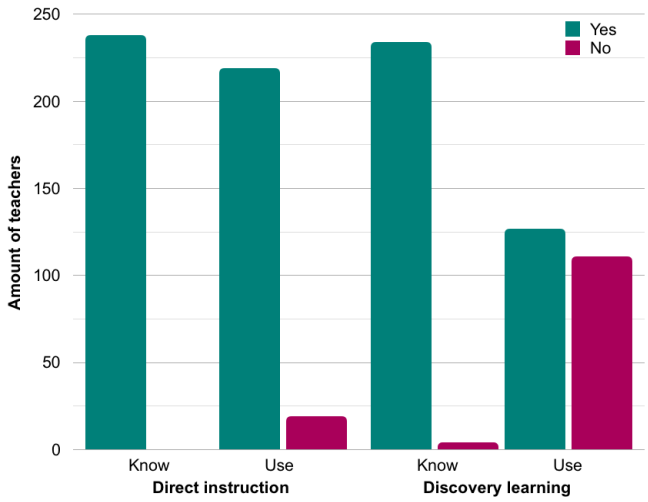


Figure 5.4: *Familiarity with direct instruction and discovery learning*

5.3.2 Familiarity with direct instruction and discovery learning

Participants were asked whether they use or know direct instruction and discovery learning in class with the explicit notion that this question is not related to programming education.

All participants indicated that they know direct instruction and a big part of them (219 or 92%) use direct instruction. Discovery learning is known by 98% of the participants but only 53% (or 127) of the participants use discovery learning. These findings are shown in Figure 5.4.

Whether the teaching methods are known or used is set against the several factors. An overview of all frequencies is displayed in Tables B.2.2 and B.2.2. Since almost all participants know direct instruction and discovery learning we found it of no interest to further analyse this. The relation between the usage of teaching methods and demographics as well as programming (teaching) experience is analysed.

A significant relation between the usage of direct instruction and school type is found with $p=0.017$. Relatively more participants working at a Jenaplan and Montessori school indicated to not use direct instruction.

The usage of discovery learning relates significantly to the amount of programming lessons taught ($p=0.016$). Of the teachers that gave over 10 programming lessons, 71% indicated to use discovery learning.

Analyzing the usage of the teaching methods and the grades taught resulted in the lower grades being a predictor of both the usage of direct instruction, with $p=0.02$, and discovery learning, with $p<0.001$. More teachers teaching lower grades indicated to use discovery learning.

Based on this data we can answer RQ1 by stating that both direct instruction and discovery learning are well known by primary school teachers. However, while direct instruction is used by many teachers, discovery learning is not.

5.3.3 Self-efficacy

Table B.2.3 shows low, middle or high scores of either STOE or PSTE against the demographics and programming (teaching) experience. As described in Section 5.1.3 using STOE does not results in valid data. Therefore only the results of PSTE are further analysed.

PSTE scores range from 12 to 60 with a mean of 37.68 and median of 40. The distribution of the PSTE scores can be found in Figure 5.5.

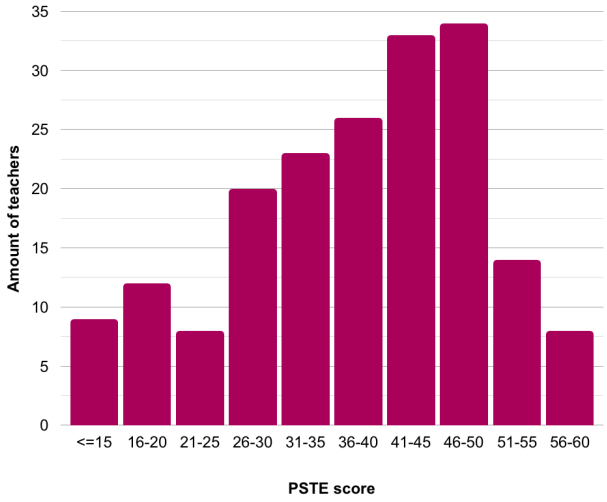


Figure 5.5: *Distribution of PSTE*

There is a significant difference between the gender labels and the PSTE score with p=0.0058. Male teachers show higher levels of PSTE than females; 34% of the males have a high PSTE against only 20% of the females. The mean PSTE of males is 41.5 (median=43.5) while the mean PSTE of females is 35.92 (median=37).

The grades 5, 6, 7 and 8 resulted, individually, in significant differences for the PSTE scores. When looking at low, middle and high grades, the high grades result in a significant difference in PSTE scores with $p<0.01$. In Table 5.2 the PSTE scores depending on the grades taught can be found.

Table 5.2: The PSTE scores depending on teaching low, middle or high grades

	Mean	**Median**	**Standard deviation**
Low grades	38.76	41	11.47
Middle grades	37.91	38.5	11.75
High grades	39.56	42.5	11.21

There is a significant difference between the PSTE scores of teachers with and without experience in programming ($p<0.001$). We found that 98% of high PSTE teachers have programming experience, while 70% of the middle and 43% low PSTE teachers do. The mean PSTE for teachers with programming experience is 41.73, without experience it is 28.35. A boxplot of both groups is shown in Figure 5.6. Next

Table 5.3: *PSTE scores based on teaching programming experience*

Programming teaching experience	Mean	Median	Range	Standard deviation
Yes (total)	41.30	44	12 - 60	10.01
1-2	29.37	31.5	12 - 44	8.88
3-5	38.71	39	23 - 51	7.46
5-10	40.48	40	25 - 55	6.93
>10	47.08	47	25 - 60	6.91
No (total)	28.41	29	12 - 48	10.29
Observed	29.25	31	12 - 45	9.46
No	27.97	27.5	12 - 48	10.80

to having programming experience, it seems that the type of experience results in different PSTE scores. Having experience with Scratch resulted in the lowest PSTE of 42.86 while having experience in textual languages resulted in a mean of 47.23.

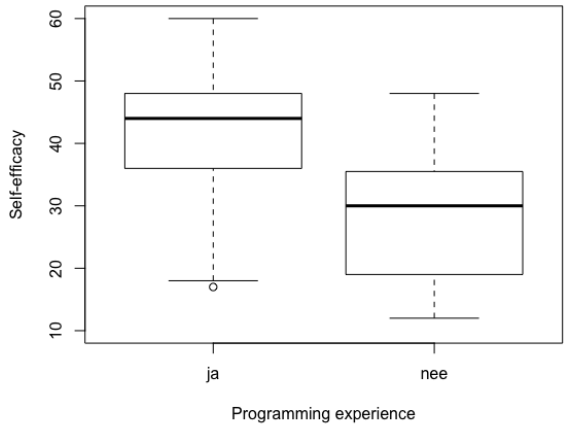


Figure 5.6: *Programming experience against self-efficacy*

Not only whether a teacher has taught programming lessons shows significant differences ($p<0.001$) but also the amount of lessons taught result in significant differences ($p<0.001$). The mean, median, range, as well as standard deviation are shown in Table 5.3.

Based on this data we can answer RQ2 by stating that gender, grade, experience in programming and experience in teaching programming result in a different self-efficacy towards programming education of Dutch primary school teachers.

5.3.4 Teaching practices and self-efficacy

Since we found in Section 5.3.2 that all participants know direct instruction and almost all know discovery learning we do not look into the relation between knowing a

teaching method and PSTE scores.

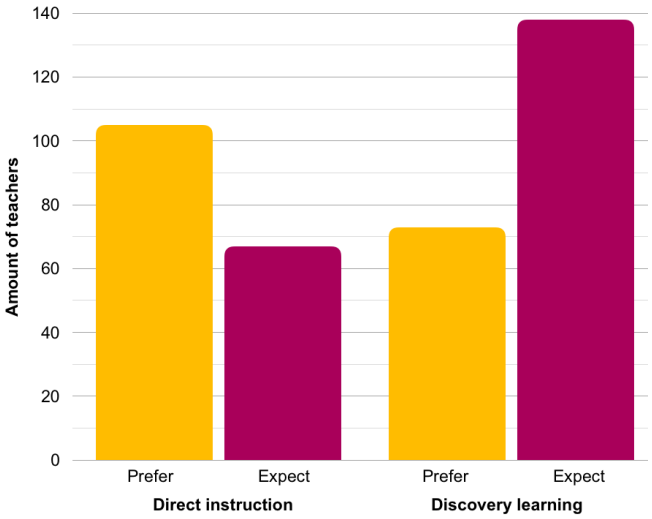


Figure 5.7: *Amount of teachers that expect to use teaching method in regular education and which teaching method they expect to use during a programming lesson*

Using a teaching method or having a preference for either one of the teaching methods does not result in significant different PSTE scores. However, we did find that more teachers prefer direct instruction than they prefer discovery learning in their regular education, as can be seen in Figure 5.7.

On the other hand, more teachers expect to use discovery learning than direct instruction during a programming lesson. Teachers that expect to use discovery learning during a programming lesson have different PSTE scores than teachers that do not expect to use discovery learning, with $p<0.01$, as can be seen in Figure 5.8. Of the teachers with a high PSTE, 89% expects to work with discovery learning when giving programming lessons. A similar pattern was not found for direct instruction.

Furthermore, we see some influences which are not significant, but still noteworthy. The majority of teachers with a high PSTE (60%) use discovery learning while this is 55% for middle and 48% for low PSTE teachers. Teachers with high PSTE scores seem to prefer discovery learning more than other teachers (high 40%, 25% middle and 34% low).

Based on this data we can answer RQ3 by stating that teachers that expect to use discovery learning during programming lessons have different PSTE scores than teachers that do not.

5.3.5 Adaptive expertise

Within the questionnaire, nothing is measured related to the concept of adaptive expertise.

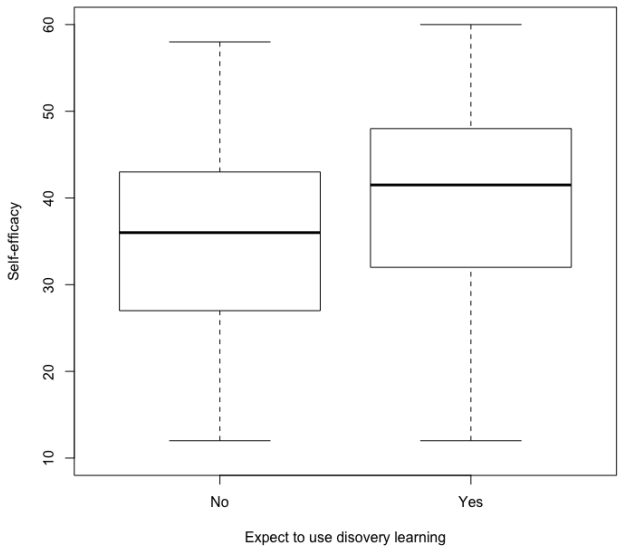


Figure 5.8: *Boxplot showing difference in PSTE scores depending on whether teachers expect to use discovery learning*

5.4 Interpretation of results

Within Section 5.3.2 we found that using direct instruction relates with school type, using discovery learning relates with the amount of programming lessons taught and the usage of both teaching methods relates to teaching lower grades. The results relating to the school types can be explained by Jenaplan and Montessori schools having a more student-centered vision influencing the methods they use. This finding is relevant when selecting participants for the experiment. Finding a relation between the amount of programming lessons taught and the usage of discovery learning in general subjects can be interpreted in multiple ways. It can be that teachers use discovery learning because they give programming lessons. However, it could also be that teachers are comfortable with giving programming lessons because they are acquainted with discovery learning. The findings relating to teaching methods and grades can be explained by lower grades focusing more on learning by playing while from grade 3 upwards there is a more explicit focus on learning.

We compared the PSTE scores we found in Section 5.3.3 with a study from Velthuis et al. [62] in which they looked at the self-efficacy of pre-service teachers towards science teaching, which is known to be low. We found that the PSTE scores for programming education are slightly lower than the PSTE scores for science teaching.

In Section 5.3.3 we also compared several factors with teachers' PSTE scores and found a difference in PSTE scores and gender, grade, experience in programming, and experience in teaching programming. Female teachers show lower PSTE scores which might be because of them holding stereotypes of programming being not for them. Higher grades teachers having higher PSTE scores can be caused by the perception of teachers. Lower and middle grade teachers might think programming does

not concern them. Teachers with programming experience have higher self-efficacy levels than teachers that do not. Interesting to notice is that Scratch shows the lowest PSTE scores compared to the other defined categories. This might be because of those teachers obtain their programming experience within the educational context, while a teacher with Python experience maybe had a programming job previously or practices programming as a hobby. Teachers that observed a lesson have a slightly higher PSTE score than the ones that do not have any programming teaching experience which is in line with vicarious experiences influencing self-efficacy, as discussed in Chapter 4. The higher PSTE scores when more lessons are taught is in line with the master experience affecting self-efficacy, which is also discussed in Chapter 4. Since Velthuis et al. [62] also looked at PSTE and experience in (science) teaching, we can compare our results with this study. We see that the PSTE scores of teachers that do not have any experience are lower for programming than for science teaching (36.81 against 28.41). Teachers with some programming teaching experience have lower PSTE scores than pre-service teachers with some science teaching experience. However, teachers who give programming lessons often (over ten) have higher PSTE scores than teachers who give science lessons less often (47.08 against 44.34).

In Section 5.1.4, we found that teachers that expect to use discovery learning during programming lessons have different, higher, PSTE scores. This might be explained by teachers with a higher self-efficacy being more interested in student-centered methods as discussed in Section 2.3.

Chapter 6

Experiment: Teaching with direct instruction or discovery learning

In the previous chapter, we gained some insights into the self-efficacy of Dutch primary school teachers. However, to learn more about the effect a teaching method might have on teachers' self-efficacy we conduct an experiment. Within this experiment, Dutch primary school teachers teach programming via either direct instruction or discovery learning. The data collected during the experiment is used to answers the following research questions:

RQ4. How does the usage of a teaching method influence the self-efficacy of Dutch primary school teachers towards programming education?

RQ5. What is the relationship between adaptive expertise and self-efficacy of Dutch primary school teachers in the context of programming education?

6.1 Method

6.1.1 Research design

In order to measure the effect of a teaching method and adaptive expertise on teachers' self-efficacy an experimental research is set-up, as depicted in Figure 6.1. Ten teachers from six different schools participated in this study. The intervention consists of two conditions, being direct instruction and discovery learning. The teachers are semi-randomly divided into one of the two conditions. Semi-randomly in the sense that if multiple teachers participated from the same school they are all in the same condition. This because teachers receiving different treatment formats within the same school would introduce a potential cross-contamination of treatment effects [59]. We also make sure that both conditions included teachers from at least two different schools. The experiment takes place at the teachers' own school and class.

Before being exposed to either the direct instruction or discovery learning intervention, a questionnaire and interview are conducted. These measurements are referred to as the pre-questionnaire and pre-interview. The pre-questionnaire consists of the STEBI-NL statements [61] which were specified for teaching programming in Scratch. The pre-interview is used to get insights into the teaching methods used during regular lessons, programming (education) experiences as well as teachers' self-efficacy.

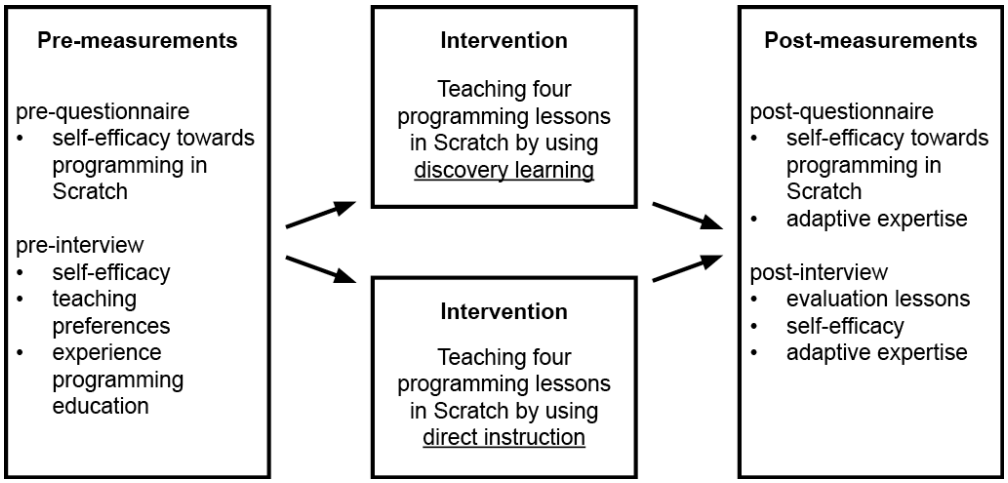


Figure 6.1: *Research design*

Questions about self-efficacy, within the pre-interview, relate to either instructional strategies, classroom management or student engagement as suggested by [58].

The interviews are followed by a preparation session, which takes at most one hour. In this session the intervention condition is revealed, an introduction to Scratch is given and the four lessons are discussed. The teachers receive manuals for all four lessons which include the structure of the lesson as well as a step-by-step manual to create the end program. The complete manuals can be found in Appendix C.1.

After the intervention, the post-questionnaire and post-interview are conducted. The post-questionnaire included the same modified STEBI-NL statements as the pre-questionnaire. Furthermore, it is extended by the Adaptive Expertise Inventory [11] which is translated to Dutch and modified to the educational context. Note that this inventory measures the adaptive expertise of teachers in general and not specific to the programming education context. The post-interview consists of an evaluation of the lessons, questions on self-efficacy as well as questions on adaptive expertise. The self-efficacy questions are similar to the pre-questions and based on [58]. The questions related to adaptive expertise are inspired by the work of [11, 16, 52]. The interview is piloted with a Computer Science teacher at the university which resulted in some questions being rephrased.

Both the interview and questionnaires can be found in Appendix C.2.

6.1.2 Lesson material

The intervention consisted of four lessons in Scratch. All four lessons are based on the Massive Open Online Course (MOOC) 'Scratch: Programming for teachers'[1]. The focus of the lessons can be found in Table 6.1. The lessons contained the same end goal and subjects for both conditions.

[1] `https://www.edx.org/course/programmeren-voor-leerkrachten-met-scratch`

Table 6.1: *Overview of the lessons*

Lesson	Topic	Concepts
1	Story telling	Sequence
2	Funny stories	Lists
3	Drawing a Spirograph	Variables and loops
4	Mathematical tables	Variables and loops

For all four lessons two different manuals are written, one following the EDI model and one following the suggested Scratch elements as described in Section 2.2.1 and Section 2.2.2 respectively.

The teacher manuals can be found in Appendix C.1.

6.1.3 Data analysis

The pre-questionnaires as well as the post-questionnaires are digitalized. For the adopted STEBI-NL statements, which are specified for teaching programming in Scratch, the scores are calculated by summing the scores from the individual statements. Since we learned in Section 5.1.3 that the STOE does not result in valid data, only the PSTE statements are considered. For the Adaptive Expertise Inventory, the ninth statement has been omitted as has been done by [11]. The first five statements are used to measure the domain-specific skill, while the remaining statements relate to the innovative skill. The average is calculated for all statements as well as the individual skills.

All interviews are audio-recorded and transcribed. The interviews conducted before the interventions are transcribed manually by going over the audio twice. The interviews conducted after the interventions are transcribed by an automatic transcription service (Amberscript [2]), after which the automatically generated transcription is enhanced manually.

Based on the questions asked, a rough coding schema is created for the pre-interview. The associated transcriptions are coded with this coding schema. Phrases assigned to the same code are clustered. Each cluster is evaluated to improve the coding schema. The improved coding schema is used to, again, go over all pre-interview transcriptions. For the post-interview transcriptions a similar process was followed. The only difference is that some codes are re-used from the pre-coding schema in order to compare pre and post answers, this includes the codes feeling towards programming, instructional strategies, classroom management, and student engagement. With the final coding schemes all interviews were coded and put into a spreadsheet after which the data is analysed and compared. The coding schemes can be found in Appendix C.2.1.

6.2 Participants

Teachers who participated in the quantitative study were asked whether they wanted to participate in the experiment. If so they could fill in their mailing address within the questionnaire. From these teachers, only teachers from public schools teaching the

[2] `amberscript.com`, GDPR compliant

upper grades are contacted. When a participant registers for the experiment it is suggested that colleagues can participate as well. An overview of the participants is given in Table 6.2. The Social Economic Status (SES) of the schools is calculated by the 'statusscores' ranking[3] where the ranking is split proportionately into five categories being very low, low, average, high and very high.

Table 6.2: *Overview of the participants, *discovery learning condition*

Teacher	School	Gender	Age	Teaching Experience (in years)	Grade	SES School
1 *	a	Female	50	5 - 10	7	very low
2 *	a	Female	35	>10	6	very low
3 *	a	Female	52	2 - 5	8	very low
4 *	c	Female	47	0 - 2	7	very low
5 *	d	Female	37	>10	7	average
6	b	Male	39	>10	6	very low
7 *	e	Female	20	0 - 2	8	average
8	e	Male	24	2 - 5	6 - 7	average
9	e	Female	28	2 - 5	8	average
10	f	Male	49	>10	7	very high

Some of the teachers have no to little programming experience while others are more experienced, as shown in Table 6.3. Teacher 8, for example, mentioned seeing programming as one black box while teacher 10 started this year as a dedicated ICT teacher and is following a study in Computer Science. Besides teacher 7, all the teachers encountered programming education in some form. Within Table 6.3, the 'Outside classroom' indicates that teachers participated in programming education activities without children, for example a workshop or training. 'Observed' implies that a teacher observed a programming lesson. More specifically, both teachers 8 and 9 had children in their classes that followed a special program that included programming. Those children 'taught' a programming lesson which the teachers observed. The label 'Teaching' indicates that teachers taught a programming lesson at least once in class by making use of existing materials, while the label 'Own material' indicates that a teacher made educational materials at least once. Teachers 1 to 5 are in the discovery learning condition, while teachers 6 to 10 are in the direct instruction condition.

6.3 Results

6.3.1 Self-efficacy

Pre-measurements

The pre-interviews give us insights in which teaching methods the participating teachers use during regular education, what teaching methods they expect to use for Scratch

[3] https://www.scp.nl/Onderzoek/Statusscores

Table 6.3: *Experience and remarks, *discovery learning condition*

Teacher	Experience programming	Amount of lessons	Remark
1 *	Teaching	4	Supported teacher 2 during her second lesson
2 *	Outside classroom	4	Received extra help during second lesson
3 *	Outside classroom	4	-
4 *	Outside classroom	5	Did the first lesson twice
5 *	Teaching	3	Combined lessons 3 and 4 in one lesson
6	Teaching	4	-
7	None	4	Is an intern Pre-questionnaire before preparation, pre-interview after first lesson
8	Observed	4	-
9	Observed	0	Observed the lessons given by participant 7
10	Own material	16	Taught same lessons to 4 different groups Dedicated ICT teacher since beginning of school year

lessons and how they feel about giving programming lessons. The findings related to these topics are summarized in Table 6.4.

All of the teachers indicated to use (a form of) direct instruction. Four teachers indicated to use discovery learning during their lessons. Three teachers indicated to use discovery learning irregularly (sometimes). The other teachers mentioned to make little use of discovery learning.

When explaining new content in regular subjects, such as mathematics and linguistics, all teachers mentioned the usage of strategies related to teacher-centered approaches. Two teachers mentioned using student-centered approaches as well by stating 'The other time you let them experience, search for themselves and discover for themselves.'[4].

When talking about giving programming lessons, half of the participants mentioned to expect teacher-centered approaches. For example, teacher 8 said 'I would at least start with a general introduction [...] then modeling something which they imitate.'[5] On the other hand, teacher 5 stated 'And then at first I just let children do something by trial-and-error. I give them a manual and just let them start. Let them just do something.'[6]

For the instructional strategies, the teachers doubted whether some instructional strategies, such as the usage of erase boards[7], could be used in programming lessons. They were also not always sure about having enough knowledge and how they could deal with the different levels[8].

[4] 'De andere keer laat je ze het ervaren, zelf opzoeken, zelf uitzoeken.'

[5] 'Ik zou in ieder geval beginnen met een algemene inleiding. [...] En dan iets voordoen, dat gaan ze nadoen.'

[6] 'En dan laat ik ook de kinderen eerst eigenlijk door trail-and-error gewoon ook. Ik geef ze een handleiding eigenlijk en begin maar. Doe maar iets.'

[7] NL: wisbordjes

[8] NL: niveauverschillen

Table 6.4: *Before interview, *discovery learning condition*

Teacher	Uses direct instruction	Uses discovery learning	Approach introducing new content in general	Expected approach Scratch lesson	Feeling towards teaching programming
1 *	Yes	Sometimes	Mix	Student	Fun Excited/tense
2 *	Yes	Little	Teacher	-	Fun Excited/tense
3 *	Yes	Yes	Teacher	Teacher	Fun Excited/tense
4 *	Yes	Little	Teacher	Teacher	Fun
5 *	Yes	Yes	Mix	Student	Fun
6	Yes	Yes	Teacher	Teacher	Fun Excited/tense
7	Yes	Sometimes	Teacher	-	Fun
8	Yes	Little	Teacher	Teacher	Fun Excited/tense
9	Yes	Sometimes	Teacher	Teacher	Fun Excited/tense
10	Yes	Yes	Teacher	Mix	Fun

With regards to classroom management, half of the teachers expected no difference between regular and programming lessons (teachers 1, 3, 7, 8, 9). The other half expected a difference in one aspect. Two teachers (2, 6) mentioned that they expected more questions being asked while the others mentioned the use of computers in class (4) and not working in subgroups (5, 10).

There were only little doubts with regards to student engagement. Teacher 8 mentioned that motivating children for programming is only possible to some extent. Teacher 2 had some concerns about the motivation of the children as well. She mentioned that it is important for students to notice something is working and therefore the assignments, at least in the beginning, should be small.

Post-measurements

Seven out of ten teachers had an improved PSTE score after the intervention, as shown in Table 6.5. Three teachers showed a decrease in their PSTE. The decrease for teacher 1 can be explained by the fact that she overestimated her programming abilities, as she stated 'Beforehand I thought I would be better at it than I turned out to be'[9]. Teacher 4 showed a major decrease in her PSTE and responded to the question on whether her self-efficacy improved with 'Yes I know it for sure however it does not feel like that. Crazy huh.'[10] She described that she experienced the programming lessons as chaotic and that the children were not able to experience success[11]. Participant 7 showed a

[9] 'Ik had me vooraf wel beter ingeschat dan dat het nu blijkt te zijn'

[10] 'Ja dat weet ik wel zeker maar het voelt niet zo. Gek he.'

[11] NL: succeservaring

decrease in PSTE as well but the decrease is minimal and no direct cause has been identified. Participant 8 showed the biggest increase although he described himself as a teacher with low self-efficacy because he stated not having enough basic knowledge.

Table 6.5: *After intervention, *discovery learning condition*

Teacher	**PSTE before**	**PSTE after**	**Difference**	**Description self-efficacy**	**Description change self-efficacy**
1 *	42.5	38	- 4.5	Low	Grown
2 *	33	40.36	+ 7.36	Middle	Grown
3 *	30	33	+ 3	Low	Grown
4 *	46	26	- 20	Low	Grown but does not feel like it
5 *	32.5	43	+ 9.5	Low	Grown
6	36	44	+ 8	Middle	Grown
7	51	49	- 2	Middle	Grown
8	26.18	38	+ 11.82	Low	Grown
9	37	45.5	+ 8.5	-	-
10	42.5	47	+ 4.5	Middle to High	Grown

All of the teachers experienced (at least a part of) teaching programming as difficult, as seen in Table 6.6. This was mostly due to a lack of knowledge of Scratch. One of the teachers (10) also mentioned that it is difficult to work with the different levels in a class. Teacher 3, however, mentioned that it was less difficult than expected; 'I thought programming was for mathkids and mathematicians and that it was really a far-from-my-bed show. That is just not for me at all. [...] so in the beginning I really thought "is this it?", that is really simple.'[12]

All the teachers mentioned they experienced programming as fun, at least to some extent. Teachers in both conditions experienced moments of insecurity. Teacher 7 mentioned to feel insecure about not being able to answer all questions. Teacher 1 mentioned that not having enough knowledge made her less secure. However, the discovery learning condition teachers also expressed feeling frustrated and disappointed in statements such as 'It is just demotivating if you do not achieve your learning goals'[13] and 'Then I am not able to help them directly. I find that very unfortunate.'[14] The direct instruction teachers did not mention this feeling but talked about feeling excited/tense.

Most of the time the teachers within both conditions were able to apply (different) instructional strategies. These strategies are not necessarily linked to a specific teaching method. Teacher 8 mentioned to be able to ask questions but assigned this ability to an external factor: the manual. Teacher 6 stated 'Eight out of ten times I had the feeling that I did not know it either'[15] about answering the questions of students.

[12] 'want programmeren was voor mij echt voor wiskids en voor wiskundigen en dat was voor mij echt een ver-van-mijn-bed show. Dat is gewoon helemaal niks voor mij. [...] dus ik dacht in het begin dacht ik echt "is dit alles?", dat is echt simpel.'

[13] 'Want het is gewoon demotiverend als je je leerdoelen niet hebt gehaald.'

[14] 'Dan kan ik niet direct helpen zeg maar. Dat vind ik wel heel jammer.'

[15] 'Bij acht van de tien keer had ik zelf het gevoel dat ik het ook niet helemaal wist.

Table 6.6: *After intervention, *discovery learning condition*

Teacher	Experience	Feeling towards teaching programming	Approach new programming lesson	Expects to teach programming more often
1 *	Difficult	Fun Insecure Frustration Disappointment	Mix	Yes
2 *	Difficult	Fun Insecure Disappointment	Teacher	Maybe
3 *	Difficult / Easy	Fun Disappointment	Student	Yes
4 *	Difficult	Fun Insecure Disappointment	Mix	Yes
5 *	Difficult	Fun	Mix	Yes
6	Difficult	Fun Excited/tense Insecure	Teacher	Yes
7	Difficult	Fun Excited/tense Insecure	Mix	Yes
8	Difficult	Fun Excited/tense Insecure	Teacher	Maybe
9	Difficult	-	Mix	-
10	Difficult	Fun	Teacher	Yes

When asking teacher 2 whether asking questions helped the children the teacher answered 'The children that understood it a little were helped I believe. Because when I asked something then they replied like "oh yeah, and then I need to do this and this". And the children that already had a hard time following along were not helped by that I think.'[16] Teachers 2 and 4 within the discovery learning condition mentioned that they did not need to challenge the children because the students already challenged themselves within the programming environment. Teacher 5 stated that she as a teacher did not necessarily add something to students' learning; 'But by doing it themselves they learn the most. It is not my merit if they can do it or not.'[17]. Teachers mentioned that you can use discovery learning for covering your lack of knowledge; 'I can easily hide behind it; if I do not know something I can say "discover it yourself".'[18] With

[16] 'De kinderen die het een beetje snapte die hielpen denk ik wel want dan dan vroeg iets en dan "oh ja dan moet ik dat en dat". En de kinderen die er al weinig van snapte die denk ik dat het niet heel veel geholpen heeft.'

[17] 'Maar door het zelf te doen leren ze het meest. Het is niet mijn verdienste of ze het wel of niet goed kunnen.'

[18] 'Ik kan er ook heel makkelijk achter verschuilen; wat ik niet weet dat ik dan kan zeggen van "on-

regards to classroom management, half of the teachers (1, 3, 6, 7, 9) did not report any differences. Teacher 2 mentioned that she was shorthanded since she received more questions than she could answer by herself. Teacher 4 mentioned that she normally controls the group compositions but did not do this in the programming lesson. Teacher 5 normally divides the class into two but during the programming lessons she did not. Teacher 8 mentioned that he normally taught in two different levels. Teacher 10 compared the intervention with the programming lessons he normally gave and explained that he normally gave shorter instructions.

All the teachers who taught programming lessons felt that they were able to engage most of their students. Some however assigned this ability (partially) to external factors. Teachers 4, 5, 7 and 9 mentioned that the creativity of the children was stimulated by the lessons and Scratch itself.

The teachers that used direct instruction mentioned several advantages of direct instruction, being clear structure and goals, suitability for weaker students, building basis knowledge, and as a teacher it is easier to prepare for the lessons. The disadvantages they mentioned relate to experiencing difficulty in working with different levels, pushing students in a certain direction, and when you teach programming more often via direct instruction it can become boring.

The teachers within the direct instruction condition also mentioned some advantages of using discovery learning; being able to challenge students, it can be useful[19] and educational[20] and it can be a way of working for the more interested students. The downside they saw for using discovery learning is that children do not always reach their potential because they do not try out some blocks, the risk of copying instead of thinking, you have to keep a close eye on the students that they all understand the material, it is more difficult for weaker students, it is a lot to take in for the students to both use a new way of learning an a new context, students are afraid of trying resulting in many questions which makes it more difficult to teach, students do not have sufficient basic knowledge of Scratch to apply discovery learning, and it is more difficult for teachers because you have to have more knowledge to ask the right questions.

The advantages of discovery learning according to teachers in the discovery learning condition relate to teachers obtaining skills, that children start trying out stuff themselves, they learn more and what they learn stick better, promotes self-confidence, children need to take initiative, they have to think in a certain way, when they find something the euphoria is bigger. One of the teachers also mentioned 'Now I can see where they get stuck and I can help them from that point onward'[21]. The disadvantage described by these teachers included that students can get out of control[22], some children need to be helped on their way in order to achieve success and one of the teachers (2) mentioned to find it difficult to not be able to tell and explain.

The advantages the discovery learning conditioned teachers saw for direct instruction include providing structure and safety to the children, achieving results sooner, it is more suited for certain students, students have to learn to listen and can learn from each other questions, being able to make the student succeed. The disadvantages are

derzoek zelf maar".'

[19] NL: handig

[20] NL: leerzaam

[21] 'Nu kan ik zien waar ze blijven hangen en kan ik ze daarvandaan helpen.'

[22] NL: stuurloos

that some students will only repeat what you did and will not go beyond, some students lose interest, it goes too slow, less programming time, patronizing. One of the teachers from both conditions mentioned 'I think that at least for this group they benefited less. I think that they progressed more now.'[23].

The approach a teacher would use in follow up programming lessons seems to depend both on personal preference and experience. When asking teacher 4 how she would feel to be in the direct instruction group she responded with 'After doing it like this, I would not do it like that'[24]. Even after she stated earlier in the interview 'It is just too free, too loose for me as well. And thereby for the children. They need some sort of light structure'[25].

Based on these results we can answer RQ4 by stating that both the usage of direct instruction and discovery learning can have a positive effect on teachers' self-efficacy. Teachers in the discovery learning condition expressed more negative feelings towards the experience of teaching programming. Furthermore, the usage of discovery learning seems to bring the risk of a decrease in self-efficacy towards programming education in Scratch.

6.3.2 Adaptive expertise

Table 6.7: *Measurements adaptive expertise*

Teacher	Adaptive Expertise Inventory	Domain-specific skill	Innovative skill	Out of comfortzone	Perceive self as innovative	Focus preparation
1	4.5	4.4	4.33	Yes	Yes	Discover
2	3.89	4	3.83	Yes	No	Mix
3	4.3	4.4	4.33	Yes	Yes	Reproduce
4	4.8	5	4.67	Yes	A bit	Mix
5	4.5	4.8	4.33	No	A bit	Reproduce
6	4	4	4	Yes	No	Reproduce
7	4.1	4.2	4.17	No	-	Reproduce
8	4.5	4.8	4.33	Yes	No	Mix
9	4.2	3.8	4.67	-	Yes	-
10	4.3	4.4	4.33	No	A bit	Mix

During the post-questionnaire and post-interview, several aspects related to adaptive expertise are measured. The adaptive expertise score and differences in PSTE scores are results from the post-questionnaire. The other three columns in Table 6.7 originate from the post-interview. Based on the post-questionnaire no direct relation between self-efficacy and adaptive expertise is found.

[23] 'Ik denk dat dat voor deze groep in ieder geval minder had gebracht. Ik denk dat ze nu verder zijn'.

[24] 'Nu ik dit heb gedaan zou ik dit niet meer zo doen.'

[25] 'Het is gewoon te vrij, te los voor mij dan ook. En daardoor ook voor de kinderen. Zij hebben ook een soort lichte structuur nodig.'

One of the teachers mentioned that as a teacher you are required to be flexible in order to answer questions from students. All the teachers described knowledge to be something dynamic.

There seems to be a relation between instructional strategy and domain-specific skills since teachers bring up the (lack of) domain-specific skills during the questions about instructional strategies, as example 'then you notice that you do not have enough knowledge about programming' [26] Carbonell et al. [12] state that an initial level of efficiency in the domain is needed before unfamiliar tasks can be introduced. Thereby the lack of domain-specific skills may restrain teachers to use their adaptive expertise.

Furthermore, during the interview several things were mentioned that relate to factors influencing adaptive expertise. A teacher mentioned that every year is different which contributes to a varying situation. Although it seems that teachers would like to focus on skills and processes, they are assessed on how children do on specific subjects. This implies that the outcome is more valued than the process. Having to reach certain goals also influenced the importance of the situation. Furthermore, risk taking is influenced by whether a teacher feels safe in their own classroom.

Based on these results we can answer RQ5 by stating that there seems to be a relation between instructional strategy (part of teachers' self-efficacy) and domain-specific skills (part of adaptive expertise). No further direct relations between adaptive expertise and self-efficacy were found.

6.4 Interpretation of results

6.4.1 Self-efficacy

During the pre-interviews teachers mentioned having doubts about whether some instructional strategies can be used. This can be due to their unfamiliarity with programming and how they perceive programming. We hypothesize that most teachers see programming education as something different than for instance math or linguistics which can result in more doubts about how to teach it.

In line with this, in the post-interview teacher 3 expressed that she experienced programming less difficult than expected. Even though she did interact with programming a bit outside of the classroom. Thinking that programming is only for the smart kids seems to contribute to her having the lowest PSTE begin score.

Within the discovery learning condition teachers expressed more negative feelings than the direct instruction teachers. We think that since the teachers in this experiment were almost all intrinsically motivated, the negative feelings did not have a visible effect on teachers' self-efficacy. However, we believe that less motivated teachers might experience a decrease in self-efficacy because of this.

Teachers mentioned that the programming environment, Scratch, provided challenges for the students but also that the teachers themselves experienced not having enough basic knowledge of Scratch. We think that if we reduce the number of blocks visible within Scratch, teachers will feel more secure about their basic knowledge and would feel that they contribute more to the student engagement. Furthermore, we think

[26] 'dan merk je dus dat je zelf te kort kennis hebt over programmeren.'

that the amount of options contributes to teachers using discovery learning as cover for not knowing, which does not seem to have a positive effect on the instructional strategy efficacy. In this way, we think that a modified Scratch environment would help in developing self-efficacy.

6.4.2 Adaptive expertise

During the post-interview, teacher 4 mentioned that she knows her self-efficacy has increased but that it did not feel that way. Apparently, the belief in her ability consists of a rational and irrational part. We think that adaptive expertise can help in understanding this finding. Teacher 4 gained domain-specific skills and knowledge and scored high on the Adaptive Expertise Inventory. While knowing she has the tools needed to perform, she felt her performance was disappointing. This resulted in a decrease in self-efficacy. Although to a lesser extent, a similar pattern is seen with teacher 1. Based on these findings, we hypothesize that teachers who are adaptive experts can have higher expectations of themselves. In the case of disappointing experiences, self-efficacy will decrease. In this experiment, we saw this effect only within the discovery learning condition.

The lack of basic knowledge can, next to an effect on self-efficacy, have a negative influence on their ability to use adaptive expertise. It could be that when the domain-specific skill is better teachers can use their adaptive expertise better. This is in line with [12] who stated that novelty in the form of unfamiliar tasks can be introduced after an initial level of efficiency in the domain is achieved. [12].

One of the differences we noted is that teachers in the discovery learning condition talked more about skills while direct instruction teachers talked more about end-results. Within our model, presented in Section 4.3, we see that when understanding is valued it has a positive effect on adaptive expertise. We interpret talking about skills rather than about the end-result as an indicator to value understanding more than the outcome. Thereby teaching programming via discovery learning could influence the adaptive expertise of the teacher.

Chapter 7

Conclusion

In this thesis, we focused on the problem that primary school teachers have a low self-efficacy towards programming education which contributes to the little amount of programming education in Dutch primary schools. The aim was to investigate the effect of teaching methods on the self-efficacy of Dutch primary school teachers towards programming by answering the following research question:

RQ. *What is the effect of teaching methods on the self-efficacy of Dutch primary school teachers towards programming education?*.

This research question is answered by conducting a questionnaire as well as an experiment. The questionnaire included questions about teachers' self-efficacy as well as their teaching preferences. Within the experiment ten teachers gave four programming lessons by using either direct instruction or discovery learning. Before the intervention, a pre-questionnaire and pre-interview were conducted about teachers' self-efficacy. After the intervention, the post-questionnaire and post-interview were conducted which includes questions about teachers' self-efficacy as well as their level of adaptive expertise.

Below the subquestions and their answers are discussed.

RQ1. *How well known and used are direct instruction and discovery learning by Dutch primary school teachers?*
We found that both direct instruction and discovery learning are well known by Dutch primary school teachers. However, while direct instruction is used by many teachers in their regular education, discovery learning is used by 53% of the teachers in our sample.

However, we believe that discovery learning is even used less by the general population of Dutch primary school teachers. This is motivated by only 32% of the schools being involved in programming education, which makes it unlikely that 66% of the Dutch primary school teachers have programming teaching experiences. Since we found a positive relation between the usage of discovery learning and programming teaching experience, a sample with less experienced teachers might result in a lower percentage of teachers that use discovery learning.

RQ2. *Which factors result in a difference in the self-efficacy of Dutch primary school teachers towards programming education?*
We found that gender, grade, experience in programming and experience in teaching programming result in a different self-efficacy towards programming education of Dutch primary school teachers.

Our finding that females have lower self-efficacy scores is important since 80% of the primary school teachers is female [56]. Both the programming experience and teaching programming experience influence self-efficacy, which is in line with the theory of [5].

RQ3. *Which teaching practices result in a difference in the self-efficacy towards programming education of Dutch primary school teachers?*
The teaching method teachers expect to use during a programming lesson influences their self-efficacy. Teachers with higher levels of self-efficacy expect to use discovery learning more than teachers with a lower level of self-efficacy.

RQ4. *How does the usage of a teaching method influence the self-efficacy of Dutch primary school teachers towards programming education?*

Getting involved in programming education, independently of the teaching method, can increase the self-efficacy of Dutch primary school teachers. However, teachers that used discovery learning expressed more negative feelings towards teaching programming. Furthermore, the usage of discovery learning seems to results in a risk of a decrease in self-efficacy.

We think that direct instruction creates a safe environment which reduces the risk of a decrease in self-efficacy. Within this experiment, the effect of negative feelings on self-efficacy was not measured, but this could be due to this sample of teachers being (intrinsically) motivated teachers. We think that these negative feelings would impact a less motivated or insecure sample of teachers negatively.

RQ5. *What is the relationship between adaptive expertise and self-efficacy of Dutch primary school teachers in the context of programming education?*

There seems to be a relation between instructional strategy (part of teachers' self-efficacy) and domain-specific skills (part of adaptive expertise). No further direct relations were found.

However, we do hypothesize that teachers who are adaptive experts can have higher expectations of themselves. In the case of disappointing experiences the self-efficacy will decrease. In this experiment we saw this effect for two teachers within the discovery learning condition.

Based on the research we have done, we can say that the self-efficacy towards programming education of Dutch primary school teachers can increase while using either direct instruction or discovery learning. The usage of discovery learning can cause more negative feelings as well as the risk of a decrease in self-efficacy.

Chapter 8

Discussion

Within this thesis, we focused on the relation between the self-efficacy of Dutch primary school teachers and teaching methods. Furthermore, we explored the role of adaptive expertise in this context. In this chapter we discuss the thesis and its limitations.

8.1 Self-efficacy and adaptive expertise

8.1.1 Existing instruments

Dutch translation of Science Teaching Efficacy Belief Instrument (STEBI-NL)

In both the quantitative and qualitative we have used the STEBI-NL [61] and adapted the statements to the context of programming and programming in Scratch. The STEBI-NL consists of two factors, the science teaching outcome expectancy (STOE) and personal science teaching efficacy (PSTE). Although we only used PSTE in our analysis, we have some suggestions for both factors.

The first STOE statement we want to discuss is the one that states that a student performs better than others[1]. According to literature [61], however, the statement should be about performing better than usual. Therefore we would reconsider this translation.

Another STEBI-NL statement that could be rephrased is the one that suggests that good programming education has little effect on students with low motivation[2]. We think that this statement does not measure a positive outcome expectancy. We suggest to change it such that it states that good programming education can affect the performance of students even for the ones with less motivation.

Due to our modification, one of the STOE statements[3] became ambiguous since 'programming program' was supposed to refer to 'programming curriculum' but could be interpreted as 'programming tool'. When re-using this translation, the statement

[1] 'Als een leerling beter presteert dan anderen tijdens een programmeerles, dan kan dat komen doordat de leerkracht zich extra heeft ingespannen voor deze leerling'

[2] 'Goed programmeeronderwijs heeft weinig effect op de prestaties van leerlingen met een slechte motivatie'

[3] 'Als leerlingen goed presteren op het gebied van programmeren ligt dat zeer waarschijnlijk aan het aangeboden programmeerprogramma op de school'

should be rephrased such that it becomes clear that the statement is about the curriculum.

One of the PSTE statements implies the usage of a student-centered teaching methods, being: 'I can guide student in such a way that they can find the answers on their own questions when working on programming assignments'[4]. We suggest to either rephrase or remove this statement to prevent an implication towards a certain teaching method.

Ohio State teacher efficacy scale (OSTES)

During the interview, within the qualitative research, questions were based on the OSTES[58]. The questions related to efficacy towards instructional strategies, classroom management, and student engagement. We experienced that the level of specificity differs for these three components. Answers related to instructional strategies were more often specific to programming education. The other two components, classroom management and student engagement, seem to be subject independent. Therefore we question the reliability of the instructional strategy part of the OSTES when not specifying a specific subject or domain.

Adaptive Expertise Inventory

The post-questionnaire within the qualitative research included a translation of the Adaptive Expertise Inventory [11]. This instrument aims to measure domain-specific skills and innovative skills, or at least how teachers perceive themselves in relation to those two components. We do not completely understand how the statements are divided among the skills. In particular, we question the categorisation of the statement about concerning yourself with the latest development within your field of expertise[5]. This statement is categorized in the domain-specific skill while we could also imagine it belonging to the innovative skill.

Another thing we would like to note is that the inventory measures adaptive expertise on a general level. However, the statements could be rephrased to subject-specific questions. This might have added value since the level of adaptive expertise someone has can differ per domain.

8.1.2 Link between self-efficacy and adaptive expertise

Within Section 4.3 we discussed how self-efficacy and adaptive expertise might relate. We mentioned that the teaching method used might have different effects depending on the initial level of self-efficacy. During the qualitative research, we saw a pattern for two teachers within the discovery learning condition. Namely, that having a higher level of adaptive expertise might result in higher expectations. When these expectations are not met, a decrease in self-efficacy can occur. We also indicated that domain-specific skills and instructional strategies might relate to each other. These

[4] 'Ik kan leerlingen zodanig begeleiden bij programmeeropdrachten, dat zij zelf antwoorden kunnen vinden op hun eigen vragen'

[5] 'Tijdens mijn loopbaan als leerkracht heb ik mij bezig gehouden met de nieuwste ontwikkelingen in het onderwijsveld'

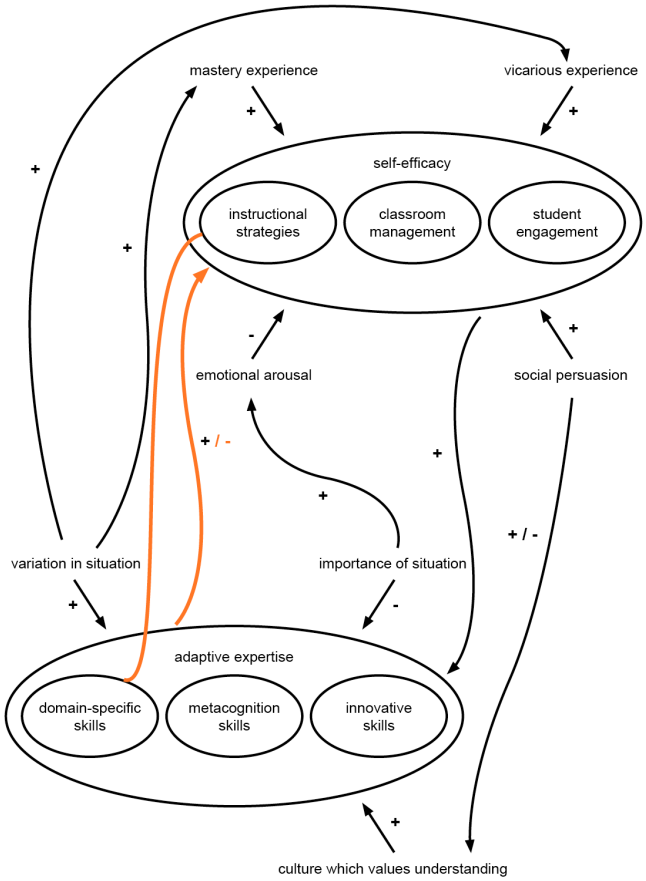


Figure 8.1: *The orange elements indicate modification we made in the model based on our findings*

two finding make us add a relation and adjust the influence of a hypothesized relations in our model, as depicted in orange in Figure 8.1.

We think that the setup of this thesis was not optimal for researching a link between self-efficacy and adaptive expertise. Within Section 8.2.2 and Section 8.3.2 we make some suggestions on how to more explicitly research the possible links between the two concepts and related factors.

8.2 Questionnaire

8.2.1 Sample

Although we did gain insights with our sample of teachers, we believe the sample is biased. First of all, less than 60% of the respondents were female while more than 80% of the Dutch primary school teachers are [56]. Furthermore, 72% of the teachers within the sample have either programming experience or programming teaching experience. It is likely that the actual percentage of experienced teachers is lower since only 32% of Dutch primary schools say to do something with programming [31]. Having more

male teachers as well as more experienced teachers in our sample makes us believe that the self-efficacy found might be lower when having a more representative sample. We think this because both gender and experience result in different levels of self-efficacy towards programming education. Furthermore, we found that teaching programming and the usage of discovery learning relate. Therefore, we think that having a more representative sample, the percentage of teachers that use discovery learning might be lower.

A more representative sample might be acquired by trying to frame the mails and flyer differently. Another way of getting a more representative sample could be by collaborating with a partner that, for instance, give professional development to teachers that go beyond the scope of programming. On the other hand, with the current lack of teachers within Dutch primary schools, we have to be realistic and say that it will be difficult to find teachers that are not involved in programming education and still are willing to make time for a questionnaire on this topic.

8.2.2 Method

Next to direct instruction and discovery learning, we included inquiry learning in the questionnaire. It might be that including this third teaching method affected the answers given by participants. In one way, the addition of inquiry learning might have helped the teachers in understanding our definition of discovery learning. The teachers could read both definitions and see the difference between the two. This is relevant since inquiry and discovery learning are sometimes used interchangeably. On the other hand, it can also confuse the participants or influence their answers.

Given that not all questions from the questionnaire are used in our analysis, we could have made the questionnaire shorter and thereby we could have reduced the risk of respondent fatigue. Furthermore, a shorter questionnaire would mean that the completion time of the questionnaire is lower. This might lower the effort it takes to participate and result in more respondents.

One of the questions, question 8, included both the mastery experience and vicarious experience of the participants. It would be better to split this question into two such that mastery and vicarious experience could be measured separately. No questions were asked about social persuasion and emotional arousal, which are both factors that affect the self-efficacy. Including questions about those two factors might give a more complete image of the self-efficacy of Dutch primary school teachers towards programming education.

We suggest to change the question about which teaching method teachers expect to use within a programming lesson, question 11 in the questionnaire. This question is answered by both teachers with and without experience. It is likely, but not certain, that the experienced teachers answer this question with the methods they are actually using. We would add a new multiple-choice question to ask teachers with programming teaching experience which teaching method they use in their programming lesson. Furthermore, a question could be added that asks which teaching method teachers find fitting for programming education. Adding an open question about this topic, such as why teachers use or expect a certain method, could also give more insights in this matter.

Within the questionnaire, nothing is asked about adaptive expertise and its relation with self-efficacy. At the time that the questionnaire was conducted, adaptive expertise was not part of the thesis yet. We would suggest to include a measure on adaptive expertise, such as the Adaptive Expertise Inventory. We also suggest to add questions about the importance of programming within the curriculum as well as about whether teachers value understanding or outcome. These elements all follow from the model as depicted in, among others, Figure 8.1. The variation within a situation is also part of the model but it is hard to measure within a questionnaire.

8.2.3 Relevance of results

With our questionnaire, we found that discovery learning is less often used compared to direct instruction. This finding strengthens the relevance of this thesis since we hypothesized that the difference in teaching method used in schools and the teaching methods used in programming education contributes to a low self-efficacy towards programming education. This hypothesis assumes that discovery learning is used less by Dutch primary school teachers which we can support with our data.

The factors that result in a difference in self-efficacy help us in generalising the results. As disused in Section 5.4, more than 80% of the Dutch primary school teachers is female [56]. Combining this with our sample consisting of 59% female teachers and the findings of female teachers having lower self-efficacy scores suggest that the self-efficacy of all Dutch primary school teachers is lower than we have measured. Furthermore, only 32% of the Dutch primary schools do something with programming [31]. On 86% of those schools, programming is taught by the groups' teacher and on those schools not all grades are said to be involved in programming education. Since 66% of the teachers in our sample said to have programming teaching experience and given that this factor results in a higher self-efficacy we also suggest that based on this data the self-efficacy might be lower than we have measured.

The difference in self-efficacy and the expectation of using discovery learning within programming lessons can be possibly explained by something we discussed in Section 2.3. Namely that teachers with a high self-efficacy are more interested in learning about student-centered methods [54].

8.2.4 Reliability, validity and ethics

To make sure that the data is reliable, the internal consistency of the self-efficacy scores was tested with the usage of Cronbach's alpha. This measurement showed that the PSTE was consistent but the OSTE was not. Therefore, the data collected with the OSTE statements were not analysed in this thesis. It was not possible to re-test since it is questionable whether the measurements are consistent across time and due to practical reasons. It could not be controlled whether the participants gained new teaching experience between the test and a re-test. It would also not be practical since the survey is filled out anonymously and voluntarily. Furthermore, data collection and analysis has been performed by one person. Therefore there are no inter-rate reliability issues between different people. The data used from the questionnaire is collected with the use of closed questions which leaves little space for subjectively. The reliability of the

data is strengthened due to knowing what the biases in the data is. Moreover, despite the bias there was a variety in demographics and experience.

One of the validity issues is that it is not known what the optimal level of specificity is for measuring self-efficacy [58]. Thus, should the items relate to teaching, programming, a specific programming language, or a programming concept. We adopted the level of specificity of the questionnaire to 'teaching programming'. This decision is made since the aim of the survey was to gain an insight into programming education.

The participants in the questionnaire participated voluntarily and did not receive any rewards for participating. The questionnaire was hosted via Collector, of which it was confirmed that the service is GDPR compliant. In order to analyse the data, the data was saved on a local, password-protected, machine.

8.3 Experiment

8.3.1 Sample

Ten different teachers participated in the experiment. We did gain insights from these teachers but bigger scale research should be conducted in order to test whether our findings are generalisable.

Moreover, we did select on teachers teaching higher grades at public schools but we could have selected on more criteria. We would suggest to be stricter in selecting the teachers and only include teachers without experience with programming or programming education. When having enough candidates to participate in the experiment, we would also select based on teachers' self-efficacy and maybe even the teaching methods they use.

8.3.2 Method

Self-efficacy

The questionnaire conducted before the interviews made use of the adopted STEBI-NL. This instrument is discussed in Section 8.1.1.

Within the interview we focused on three efficacy components, being instructional strategy, classroom management as well as student engagement. We saw that teachers were sometimes insecure about which instructional strategies they could use beforehand. Moreover, after the intervention, the instructional strategy was linked mostly to required knowledge. We therefore think that the instructional strategy efficacy is domain-specific and tells us more about the self-efficacy of the teachers. The classroom management and student engagement efficacy do not tell us much about teachers' self-efficacy towards programming but more about the efficacy as a teacher in general. Therefore we argue to focus on the instructional strategies more within the interview.

Within our experiment, the teachers gave four programming lessons. It would be interesting to research how teachers' self-efficacy changes over a longer period of time by extending the number of lessons. We would suggest to measure the self-efficacy multiple times when extending the amount of lessons.

Since teachers' beliefs on children and how they learn are associated with their self-efficacy [28], we could have measured these beliefs upfront as Jamil et al. did [28].

Another suggestion is to do the same experiment with another programming language or a modification of Scratch. Teachers experience the Scratch environment to be overwhelming. One of the teachers stated 'Ik merk dat er zo veel mogelijk is met Scratch, echt zo ontzettend veel dat je vooral als beginner of vooral start met "Welke bomen in dit bos heb ik nodig om dit te kunnen doen"'. Reducing the amount of blocks in Scratch gives more control to the teacher and reduces the amount of knowledge a teacher needs.

Adaptive expertise

Within the post-questionnaire, the translated and modified Adaptive Expertise Inventory is used. This instrument is discussed in Section 8.1.1. The measurements we did on the concept of adaptive expertise were self-reported. The participants of the experiment shared their view of their own adaptive expertise with use. We did observe a part of the lessons but we did not have enough knowledge about teaching to observe whether teachers make use of their adaptive expertise. A protocol for such observations would be helpful. We also suggest to measure adaptive expertise in the pre-questionnaire to see whether self-efficacy affects adaptive expertise. We could also include questions about importance of the situation to see whether this indeed affects the emotional arousal of teachers.

8.3.3 Relevance of results

The results show that getting involved in programming education, independently of the teaching method, can increase the self-efficacy of primary school teachers. We also found that discovery learning can have a negative impact on the self-efficacy. To avoid the risk of a decrease in self-efficacy we suggest to adjust existing programming material and tools such that they can be used with the direct instruction method. This is especially relevant for programming education since the current material is often based on discovery learning.

8.3.4 Reliability, validity and ethics

In the experiment re-testing is not possible since the participants underwent an intervention aimed at changing the main construct measured.

The data collection and analysis has been performed by one person. Therefore there are no inter-rate reliability issues. However, this data is sensitive to subjectively. The validity of the results can be improved by having another researcher going over the same data and compare the coding results.

While the questionnaire in the experiment related to 'teaching in Scratch'. while the experiment aimed to measure the effect of the intervention. Since the intervention only related to a single aspect of programming, being Scratch, the items were adapted to this level.

The participants in the questionnaire and experiment participated voluntarily. The participants in the survey did not receive any rewards. The experiment participants

did receive a small thank you gift, but this was not known in advance and only given after all data was collected. The participants from the experiment signed an informed consent form.

Online services used during the experiment include Amberscript for automatic transcription and Surfdrive to backup some of the work. These services are both GDPR compliant.

Within the transcription, names are omitted. Sometimes an interview was interrupted by a student or another teacher. These interruptions are not included in the transcription since they were not about the research.

8.4 Contributions

8.4.1 Scientific contributions

To our knowledge, little to no research has been performed on how teaching methods affect self-efficacy of teachers towards programming education. With this thesis, we saw that the usage of both direct instruction and discovery learning can increase teachers' self-efficacy towards programming education. However, using discovery learning seems to have the risk of a decrease in self-efficacy. Moreover, this thesis indicates that the Scratch environment seems to have too many options for teachers to use in it their classroom. This finding results in a requirement when researching new educational environments. Furthermore, the adoption of the STEBI-NL for the programming context and the translation of the Adaptive Expertise Inventory can be used by other researchers. The data collected during this thesis can be used to write one or multiple scientific papers.

8.4.2 Practical contributions

The work resulted in four manuals that can be re-used by teachers. The work shows programming does not necessarily have to be taught with discovery learning. Developers of professional development as well as programming educational material should consider that discovery learning is not something that all teachers are comfortable with. When the training or material they develop uses discovery learning they should include a module about how to use this teaching method.

8.5 Future research

During this research, we taught of several research topics that might be interesting to pursue in the area of programming education. The ones we are most interested in are listed below.

Teacher-student interaction Measuring the self-efficacy of not only the teacher but also of the students would give us more insight into the interaction between student and teacher. Moreover, as described in Section 2.3, the performance and self-efficacy of students' might have a direct influence on teachers' self-efficacy. Furthermore, it would be interesting to explore what expectations teachers have students and what role stereotyping might play in this.

Strategy when not knowing the answer Many teachers mentioned that when students ask a question on which they do not know the answer, they search the answer online. However, it is quite difficult to find solutions online when getting stuck with Scratch.

Influence of fun During the interviews teachers mentioned the aspect of programming being fun. It sometimes seems that 'fun' is a requirement when teaching programming. How do this impact the way teachers see programming and how they teach the subject.

Misconception among teachers Teachers mentioned that it is helpful to first make programming assignments yourself such that you know what children will encounter. This makes us wonder whether teachers and students face the same problems when learning to program.

Differences in level One of the most mentioned challenges during the interviews was working with the different levels among students. How can teachers support children that have troubling learning to program and challenge students who understand programming quickly given the fact that the programming knowledge of teachers is limited.

8.6 Working in the domains of Computer Science and Science Communication

Before describing what combining the two domains brought to this thesis, I will describe what Computer Science and Science Communication mean for me and what each of the domains contributed to this work. Since this is my personal perspective and reflection I wrote this section in the 'I' / 'me' form.

For me the core of computer science is about developing software we can use to solve problems as well as developing software in such a way that it is efficient and maintainable. When thinking about solving problems one might think about big societal problems, but these problems can be much closer to home. Being quizzed by a machine when learning for a test or creating an overview of our monthly expenses are both examples of how to use software to our benefits. Being able to think of such solutions and creating them is relevant for everyone in our society. Therefore, I believe, that it is important to learn everyone a bit about programming and this made me interested in the field of Computer Science Education. Next to my interest in Computer Science Education, my background in computer science allowed me to focus on the aspects of teaching and learning which were unknown to me. Moreover, computer science includes research on how user use a certain system. Educational systems consist of the technical part, being the programming environment, and also users can use it. How a teachers uses a system in class relates directly to teaching methods.

For me, science communication is about building a bridge between experts and non-experts. These non-experts can be stakeholders in industry or government but also the general public. One of the aspects of science communication focuses on is professional development. Within professional development, adaptive expertise is one of the key concepts.

Working in different domains is challenging and pushes you to look beyond the limitations of a field. It allows you to see links you normally might not see. An example is that during the research I noted that descriptions teachers gave about programming and direct instruction teachers mentioned in both cases that is is about taking steps. The fun thing about working in different domains is that there is a lot to learn and also that there are many opportunities to obtain new knowledge. Different fields focus on different methods and are critical in different ways, this reinforces each other.

While working on the problem of little programming being taught at primary schools, I came across an aspect for which I believe it is interesting to look into further from a Science Communication perspective. First, the perceptions teachers have towards programming education. Some teachers seem to think of programming as a scary thing, something they can not do. Science communication skills and knowledge can be applied to analyse this problem and work on a communication strategy accordingly. It seems that the innovators and early adaptors are involved, but how to involve the rest?

Bibliography

[1] Efthimia Aivaloglou and Felienne Hermans. “Early Programming Education and Career Orientation: The Effects of Gender, Self-Efficacy, Motivation and Stereotypes”. In: *Proceedings of the 50th ACM Technical Symposium on Computer Science Education*. ACM. 2019, pp. 679–685.

[2] Efthimia Aivaloglou and Felienne Hermans. “How kids code and how we know: An exploratory study on the Scratch repository”. In: *Proceedings of the 2016 ACM Conference on International Computing Education Research*. ACM. 2016, pp. 53–61.

[3] Louis Alfieri et al. “Does discovery-based instruction enhance learning?” In: *Journal of educational psychology* 103.1 (2011), p. 1.

[4] Richard Arends and Sharon Castle. *Learning to teach*. Vol. 2. McGraw-Hill New York, 1991.

[5] Albert Bandura. “Self-efficacy: toward a unifying theory of behavioral change.” In: *Psychological review* 84.2 (1977), p. 191.

[6] Albert Bandura. “Self-efficacy. In. VS Ramachaudran”. In: *Encyclopedia of human behavior* 4.4 (1994), pp. 71–81.

[7] Albert Bandura and Richard H Walters. *Social learning theory*. Vol. 1. Prentice-hall Englewood Cliffs, NJ, 1977.

[8] Mustafa Baser. “Attitude, gender and achievement in computer programming.” In: *Online Submission* 14.2 (2013), pp. 248–255.

[9] Nathan Bean et al. “Starting from scratch: Developing a pre-service teacher training program in computational thinking”. In: *2015 IEEE Frontiers in Education Conference (FIE)*. IEEE. 2015, pp. 1–8.

[10] Conny Boendermaker et al. “” Ik gebruik die taaldingetjes nu ook in andere lessen”. In: *Tijdschrift voor Lerarenopleiders* 38 (2017), p. 4.

[11] Katerina Bohle Carbonell et al. “Measuring adaptive expertise: development and validation of an instrument”. In: *European Journal of Work and Organizational Psychology* 25.2 (2016), pp. 167–180.

[12] Katerina Bohle Carbonell et al. “How experts deal with novel situations: A review of adaptive expertise”. In: *Educational Research Review* 12 (2014), pp. 14–29.

[13] Frank Coffield et al. *Learning styles and pedagogy in post-16 learning: A systematic and critical review*. 2004.

[14] Per-Erik Ellström. "Integrating learning and work: Problems and prospects". In: *Human resource development quarterly* 12.4 (2001), pp. 421–435.

[15] Peggy A Ertmer and Timothy J Newby. "Behaviorism, cognitivism, constructivism: Comparing critical features from an instructional design perspective". In: *Performance improvement quarterly* 6.4 (1993), pp. 50–72.

[16] Janna H. Ferguson et al. "Adaptive Expertise: The Development of a Measurement Instrument". In: https://peer.asee.org/29752. ASEE Conferences, 2018.

[17] Varvara Garneli, Michail N Giannakos, and Konstantinos Chorianopoulos. "Computing education in K-12 schools: A review of the literature". In: *2015 IEEE Global Engineering Education Conference (EDUCON)*. IEEE. 2015, pp. 543–551.

[18] Jodi S Goodman and James O'Brien. "Teaching and learning using evidence-based principles". In: *The Oxford handbook of evidence-based management* (2012), pp. 309–336.

[19] Shuchi Grover and Satabdi Basu. "Measuring student learning in introductory block-based programming: Examining misconceptions of loops, variables, and boolean logic". In: *Proceedings of the 2017 ACM SIGCSE technical symposium on computer science education*. ACM. 2017, pp. 267–272.

[20] Giyoo Hatano and Kayoko Inagaki. "Two courses of expertise". In: *RESEARCH AND CLINICAL CENTER FOR CHILD DEVELOPMENT Annual Report* 6 (1984), pp. 27–36.

[21] John Hattie. "Influences on student learning". In: *Inaugural lecture given on August* 2 (1999), p. 1999.

[22] Terry Haydn. "How do you get pre-service teachers to become 'good at ICT'in their subject teaching? The views of expert practitioners". In: *Technology, Pedagogy and Education* 23.4 (2014), pp. 455–469.

[23] Fredrik Heintz, Linda Mannila, and Tommy Färnqvist. "A review of models for introducing computational thinking, computer science and computing in K-12 education". In: *2016 IEEE Frontiers in Education conference (FIE)*. IEEE. 2016, pp. 1–9.

[24] John R Hollingsworth and Silvia E Ybarra. *Explicit direct instruction (EDI): The power of the well-crafted, well-taught lesson*. Corwin Press, 2017.

[25] Carolyn J Hushman and Scott C Marley. "Guided instruction improves elementary student learning and self-efficacy in science". In: *The Journal of Educational Research* 108.5 (2015), pp. 371–381.

[26] Maya Israel et al. "Describing elementary students' interactions in K-5 puzzle-based computer science environments using the collaborative computing observation instrument (c-coi)". In: *Proceedings of the 2017 ACM Conference on International Computing Education Research*. ACM. 2017, pp. 110–117.

[27] Maya Israel et al. “Empowering K–12 students with disabilities to learn computational thinking and computer programming”. In: *TEACHING Exceptional Children* 48.1 (2015), pp. 45–53.

[28] Faiza M Jamil, Jason T Downer, and Robert C Pianta. “Association of preservice teachers’ performance, personality, and beliefs with teacher self-efficacy at program completion”. In: *Teacher Education Quarterly* 39.4 (2012), pp. 119–138.

[29] Anthony Jones. “Teacher Education and Constructionism when Teaching with Digital Technologies”. In: (2015).

[30] Yasmin B Kafai and Quinn Burke. “Computer programming goes back to school”. In: *Phi Delta Kappan* 95.1 (2013), pp. 61–65.

[31] Kennisnet. *Programmeren nog een moeilijk verhaal voor scholen*. 2015. URL: https://www.kennisnet.nl/fileadmin/kennisnet/digitale_vaardigheden/programmeren_maken/bijlagen/Programmeren_nog_een_moeilijk_verhaal_voor_scholen.pdf (visited on 10/10/2019).

[32] Ada S Kim and Andrew J Ko. “A pedagogical analysis of online coding tutorials”. In: *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education*. ACM. 2017, pp. 321–326.

[33] Beaumie Kim. “Social constructivism”. In: *Emerging perspectives on learning, teaching, and technology* 1.1 (2001), p. 16.

[34] Paul A Kirschner, Rob L Martens, and Jan-Willem Strijbos. “CSCL in higher education?” In: *What we know about CSCL*. Springer, 2004, pp. 3–30.

[35] Paul A Kirschner, John Sweller, and Richard E Clark. “Why minimal guidance during instruction does not work: An analysis of the failure of constructivist, discovery, problem-based, experiential, and inquiry-based teaching”. In: *Educational psychologist* 41.2 (2006), pp. 75–86.

[36] David Klahr and Milena Nigam. “The equivalence of learning paths in early science instruction: Effects of direct instruction and discovery learning”. In: *Psychological science* 15.10 (2004), pp. 661–667.

[37] Michiel Lucassen. *Leren programmeren: waarom eigenlijk?* 2017. URL: https://www.vernieuwenderwijs.nl/leren-programmeren-waarom-eigenlijk (visited on 06/25/2019).

[38] John Maloney et al. “The scratch programming language and environment”. In: *ACM Transactions on Computing Education (TOCE)* 10.4 (2010), p. 16.

[39] Linda Mannila, Lars-Åke Nordén, and Arnold Pears. “Digital competence, teacher self-efficacy and training needs”. In: *Proceedings of the 2018 ACM Conference on International Computing Education Research*. ACM. 2018, pp. 78–85.

[40] Richard E Mayer. “Should there be a three-strikes rule against pure discovery learning?” In: *American psychologist* 59.1 (2004), p. 14.

[41] Orni Meerbaum-Salant, Michal Armoni, and Mordechai Ben-Ari. “Habits of programming in scratch”. In: *Proceedings of the 16th annual joint conference on Innovation and technology in computer science education*. ACM. 2011, pp. 168–172.

[42] Leonel Caseiro Morgado. "Framework for computer programming in preschool and kindergarten". PhD thesis. Universidade de Tras-os-Montes e Alto Douro, 2005. URL: http://hdl.handle.net/10348/5344.

[43] Mark Noone and Aidan Mooney. "Visual and textual programming languages: a systematic review of the literature". In: *Journal of Computers in Education* 5.2 (2018), pp. 149–174.

[44] Seymour Papert. *Mindstorms: Children, computers, and powerful ideas*. Basic Books, Inc., 1980.

[45] Paul R Pintrich et al. "Reliability and predictive validity of the Motivated Strategies for Learning Questionnaire (MSLQ)". In: *Educational and psychological measurement* 53.3 (1993), pp. 801–813.

[46] Ben Putano. *A Look At 5 of the Most Popular Programming Languages of 2019*. 2019. URL: https://stackify.com/popular-programming-languages-2018/ (visited on 10/11/2019).

[47] Iris M Riggs and Larry G Enochs. "Toward the development of an elementary teacher's science teaching efficacy belief instrument". In: *Science Education* 74.6 (1990), pp. 625–637.

[48] Michele Roberts, Kiki Prottsman, and Jeff Gray. "Priming the Pump: Reflections on Training K-5 Teachers in Computer Science". In: *Proceedings of the 49th ACM Technical Symposium on Computer Science Education*. ACM. 2018, pp. 723–728.

[49] Erik Roelofs and Jacqueline Visser. "Leeromgevingen volgens ouders en leraren: voorkeuren en realisatie". In: *Pedagogische Studiën* 78.3 (2001), pp. 151–168.

[50] Marc Roosenbrand and N Mohammadi Fard. "Self-efficacy beliefs van leraren". MA thesis. 2013.

[51] Algemene Vereniging Schoolleiders. *Werken met onderzoekend en ontwerpend leren vereist scholing leerkrachten*. 2017. URL: https://www.avs.nl/sites/default/files/KaderDigitaal-5-januari-2017a.pdf (visited on 10/11/2019).

[52] DL Schwartz, JD Bransford, and D Sears. "Innovation and efficiency in learning and transfer". In: *Transfer of learning from a modern multidisciplinary perspective* (2005), pp. 1–51.

[53] Scratch. *Educator Guide*. URL: https://resources.scratch.mit.edu/www/guides/en/EducatorGuidesAll.pdf (visited on 06/27/2019).

[54] Lyn Ely Swackhamer et al. "Increasing the self-efficacy of inservice teachers through content knowledge". In: *Teacher Education Quarterly* 36.2 (2009), pp. 63–78.

[55] Alaaeddin Swidan, Felienne Hermans, and Marileen Smit. "Programming misconceptions for school students". In: *Proceedings of the 2018 ACM Conference on International Computing Education Research*. ACM. 2018, pp. 151–159.

[56] Tanja Traag. "Leerkrachten in het basisonderwijs". In: (2018).

[57] Marjolein van Trigt. *Waarom je kinderen wel/niet moet leren programmeren*. 2016. URL: https://www.vn.nl/waarom-je-kinderen-welniet-moet-leren-programmeren (visited on 06/25/2019).

[58] Megan Tschannen-Moran and Anita Woolfolk Hoy. "Teacher efficacy: Capturing an elusive construct". In: *Teaching and teacher education* 17.7 (2001), pp. 783–805.

[59] Megan Tschannen-Moran and Peggy McMaster. "Sources of self-efficacy: Four professional development formats and their relationship to self-efficacy and implementation of a new teaching strategy". In: *The elementary school journal* 110.2 (2009), pp. 228–245.

[60] Linda Vanasupa, Jonathan Stolk, and Trevor Harding. "Application of self-determination and self-regulation theories to course design: Planting the seeds for adaptive expertise". In: *International Journal of Engineering Education* 26.4 (2010), p. 914.

[61] Chantal Velthuis. *Collaborative curriculum design to increase science teaching self-efficacy*. University of Twente, 2014.

[62] Chantal Velthuis, Petra Fisser, and Jules Pieters. "Teacher training and pre-service primary teachers' self-efficacy for science teaching". In: *Journal of science teacher education* 25.4 (2014), pp. 445–464.

[63] JM Walker et al. "Design scenarios as an assessment of adaptive expertise". In: *situations* 1.2 (2006).

[64] David Weintrop and Uri Wilensky. "Bringing blocks-based programming into high school computer science classrooms". In: *Annual Meeting of the American Educational Research Association (AERA). Washington DC, USA*. 2016.

[65] Scratch Wiki. *Blocks*. URL: https://en.scratch-wiki.info/wiki/Blocks (visited on 10/23/2019).

Appendix A

Literature search

Some of the used literature is suggested by experts in the field of (Computer Science) Education.

Literature is found via searching with the search engines Google Scholar, ERIC and Scopus. Literature is selected by judging whether the title is relevant. A title is perceived as relevant depending on the search. For instance, when looking for sources on teachers' self-efficacy a title is perceived as relevant when it mentions the educational context or professional development. When a title is perceived as relevant, the abstract is read to determine whether to proceed to the source itself. In the case of searching for literature on teachers' self-efficacy the abstract often described whether something is actually measured on teachers.

Filters are used to narrow the searches down including year of publication when searching for recent work (last five years), type of education within ERIC (Elementary education and Elementary Secondary Education) and usage of references. The last filter is used in the search for literature linking self-efficacy and adaptive expertise.

When a relevant statement is made within a paper which refers to another source, we looked at the reference of this statement. Depending on the relevance of the title the paper was looked up or not. For some papers that we find interesting as a whole, the reference list is looked through.

For some of the references used in this work, it is indicated how we found the work.

Teaching methods

Etmer and Newby [15], via a course in 'educational science'
Arends and Castle [4], recommended by expert
Hollingsworth and Ybarra [24], recommended by expert
Israel et al. [26], via ERIC by searching on 'programming' and filtering on 'Elementary Secondary Education'

Self-efficacy and adaptive expertise

Kirschner et al. [34], recommended by expert
Klahr and Nigam [36], via Kirschner et al. [34]
Bandura [5], via Velthuis et al. [62] but we came across this source multiple times

Velthuis et al. [62], via a paper which is not included in this thesis
Tschannen-Moran and Hoy [58], via Velthuis et al. [62]
Hatano and Inagaki [20], via Carbonell et al. [11]
Carbonell et al. [11], via Scopus by searching on ‘adaptive expertise’
Carbonell et al. [12], via Scopus by searching on ‘adaptive expertise’
Walker et al. [63], via Scopus by searching on ‘adaptive expertise’ and filtering on work that refers to Bandura

Appendix B

Experiment

B.1 Method - Questionnaire

Programmeren in de klas

Beste leerkracht,

Tijdens mijn studie informatica ben ik geïnteresseerd geraakt in het onderwijs en nu geef ik af en toe programmeerlessen. Voor mijn afstuderen heb ik de kans gekregen om onderzoek te doen binnen dit onderwerp. Ik focus mij hierbij op leerkrachten in het basisonderwijs.

In mijn afstudeeronderzoek kijk ik naar hoe leerkrachten in Nederland denken over programmeren en hoe programmeren kan worden toegepast in de klas. Ik hoop hiermee bij te kunnen dragen aan het toegankelijk(er) maken van programmeren in de klas. Daarom wil ik u vragen om mijn vragenlijst in te vullen. Hierbij zijn er geen juiste of onjuiste antwoorden. Hoe meer deelnemers aan het onderzoek deelnemen hoe beter ik een beeld kan krijgen van programmeren in het basisonderwijs.

Het invullen van de vragenlijst duurt ongeveer 15 minuten. De eerste vragen zijn algemene vragen, het tweede deel gaat over programmeren in de klas. Ten slotte vraag ik u om een aantal persoonsgegevens. De ingevulde vragenlijst zal vertrouwelijk worden behandeld en de ingevulde antwoorden zullen in publicatie anoniem gebruikt worden.

Indien u vragen heeft over het onderzoek en/of de ingevulde data dan kunt u contact met mij opnemen via S.deWit@tudelft.nl.

Alvast bedankt voor het invullen!

Shirley de Wit
Masterstudent aan de TU Delft

Lesvormen

Een aantal vragen gaan over verschillende lesvormen. Hieronder staan drie lesvormen beschreven om te verduidelijken wat we onder de lesvormen verstaan.

Directe instructie. Tijdens directe instructie legt een leerkracht klassikaal de leerstof uit. Daarbij is er veel structuur en is de les opgebouwd uit een aantal fasen: terugblik, oriëntatie, instructie, begeleid inoefenen, controle, verwerking en afronding. In deze lesvorm wordt het begrip van de leerling regelmatig gecontroleerd.

Onderzoekend leren. Bij onderzoekend leren, onderdeel van onderzoekend en ontwerpend leren, werken de leerlingen in een cyclus die bestaat uit een aantal onderdelen, namelijk introductie van het probleem, verkennen, opzetten experiment, uitvoeren experiment, concluderen, presenteren en tot slot verbreden en verdiepen. De leerkracht begeleid de leerling door de cyclus.

Ontdekkend leren. Bij ontdekkend leren wordt de natuurlijke nieuwsgierigheid van de leerlingen als uitgangspunt genomen. Er wordt thematisch gewerkt en nieuwe kennis wordt in context geplaatst. De leerling ontdekt de leerstof zelf waarbij de leerkracht een coachende rol heeft.

Bron: www.wij-leren.nl

Algemeen deel

De vragen in dit deel zijn **niet** gerelateerd aan programmeren maar gaan over het onderwijs in het algemeen.

1. **Geef bij elke lesvorm aan of u deze lesvorm kent. Geef ook aan of u de lesvorm regelmatig in de klas gebruikt. Omcirkel wat van toepassing is.**

	Ik ken deze lesvorm	Ik gebruik deze lesvorm regelmatig
Directe instructie	Ja / Nee	Ja / Nee
Onderzoekend leren	Ja / Nee	Ja / Nee
Ontdekkend leren	Ja / Nee	Ja / Nee

2. **Welke lesvorm heeft uw voorkeur?**

- ☐ Directe instructie
- ☐ Onderzoekend leren
- ☐ Ontdekkend leren
- ☐ Anders, namelijk _____

3. **Hoeveel vakkennis heeft een leerkracht nodig voor onderstaande lesvormen denkt u?**

	Leerkracht heeft veel kennis nodig						Leerkracht heeft weinig kennis nodig
	1	**2**	**3**	**4**	**5**	**6**	**7**
Directe instructie	O	O	O	O	O	O	O
Onderzoekend leren	O	O	O	O	O	O	O
Ontdekkend leren	O	O	O	O	O	O	O

Kunt u uw bovenstaande keuzes toelichten?

_____

_____

_____

4. **Hoe ervaart u het werken met computers? Let op, het gaat hier om situaties waarbij u zelf met de computer werkt zoals het opzoeken van informatie op internet, het gebruiken van een tekstverwerker en het doen van digitale administratie.**

- ☐ Ik vind het leuk om met de computer te werken
- ☐ Ik heb geen probleem met werken met computers
- ☐ Ik heb zo nu en dan problemen met computers maar ik kom er meestal wel uit
- ☐ Ik zie op tegen werken met computers
- ☐ Anders, namelijk _____

Programmeeronderwijs
De onderstaande vragen zijn **wel** gerelateerd aan programmeren.

5. **Welke woorden associeert u met 'programmeren'? Schrijf de eerste drie dingen die in u opkomen op.**

   1. _____
   2. _____
   3. _____

6. **Wat doet een programmeur volgens u? Geef maximaal drie voorbeelden.**

   _____

   _____

   _____

7. **Heeft u ervaring met programmeren?**

   ☐ Ja, namelijk in Scratch / Micro:Bit / Python / HTML / Anders, namelijk _____
   ☐ Nee

   Zo ja, hoe bent u aan de slag gegaan met programmeren? Welke bronnen (bijvoorbeeld websites) gebruikt u / heeft u gebruikt?

   _____

   _____

8. **Heeft u ervaring met zelf een programmeerles geven in de klas?**

   ☐ Ja, dit heb ik meer dan 10 keer gedaan
   ☐ Ja, dit heb ik tussen de 5 en 10 keer gedaan
   ☐ Ja, dit heb ik 3 tot 5 keer gedaan
   ☐ Ja, dit heb ik 1 of 2 keer gedaan
   ☐ Nee, maar ik heb wel een (gast)les programmeren ervaren
   ☐ Nee, nog nooit

   Zo ja, hoe heeft u deze les(sen) vorm gegeven?

   _____

   _____

   Zo ja, kunt u uw ervaring beschrijven? Wat ging goed en wat was lastiger?

   _____

   _____

   Zo nee, wat is hier de reden voor denkt u?

   _____

   _____

9. Heeft u weleens deelgenomen aan een training of online cursus welke te maken heeft met programmeren in de klas?

- ☐ Ja, namelijk _____
- ☐ Nee

Zo ja, met welke reden(en) heeft u mee gedaan aan deze training of cursus?

- ☐ Ik heb zelf interesse in het toepassen van programmeren in de klas
- ☐ Vanuit de leerlingen is er vraag naar programmeren in de klas
- ☐ Vanuit de ouders wordt er gevraagd naar programmeren in de klas
- ☐ Vanuit het schoolbestuur is er een push voor programmeren in de klas
- ☐ Anders, namelijk _____

10. Wat is er volgens u nodig om programmeren in het basisonderwijs tot een succes te brengen? Noem maximaal drie dingen op volgorde van effect, dus op 1. hetgeen wat volgens u het meest effect heeft.

1. _____
2. _____
3. _____

11. Bij een programmeerles verwacht ik aan de slag te gaan met

- ☐ Directe instructie
- ☐ Onderzoekend leren
- ☐ Ontdekkend leren
- ☐ Anders, namelijk _____

12. Geef bij de volgende stellingen aan in hoeverre u het eens of oneens bent. Mocht u nog geen programmeerles geven, vul dan in wat uw verwachtingen zijn.

	Volledig oneens	Oneens	Noch eens / Noch oneens	Eens	Helemaal eens
Als een leerling beter presteert dan anderen tijdens een programmeerles, dan kan dat komen doordat de leerkracht zich extra heeft ingespannen voor deze leerling	O	O	O	O	O
Wanneer de programmeerresultaten van leerlingen verbeteren, komt dit vaak door een verandering in de manier van lesgeven door de leerkracht	O	O	O	O	O
Als leerlingen goed presteren op het gebied van programmeren ligt dat zeer waarschijnlijk aan het aangeboden programmeerprogramma op de school	O	O	O	O	O
Onvoldoende achtergrondkennis van leerlingen op het gebied van programmeren kan overwonnen worden door programmeeronderwijs op een andere manier aan te bieden	O	O	O	O	O
Goede programmeerprestaties van leerlingen zijn over het algemeen te danken aan de kwaliteit van de leerkracht	O	O	O	O	O

	Volledig oneens	Oneens	Noch eens / Noch oneens	Eens	Helemaal eens
Wanneer een slecht presterend kind vooruitgang boekt bij programmeren, is het vaak het resultaat van extra aandacht door de leerkracht	O	O	O	O	O
Extra inspanningen voor programmeeronderwijs door de leerkracht leiden tot verbeteringen in de leerlingresultaten	O	O	O	O	O
De leerkracht is over het algemeen verantwoordelijk voor de resultaten van de kinderen voor programmeren	O	O	O	O	O
De prestaties van leerlingen voor programmeren zijn direct afhankelijk van de invulling van de programmeerlessen van hun leerkracht	O	O	O	O	O
Als ouders aangeven dat hun kind meer interesse toont voor programmeren, komt dat waarschijnlijk door de kwaliteiten van de leerkracht op dit gebied	O	O	O	O	O
Goed programmeeronderwijs heeft weinig effect op de prestaties van leerlingen met een slechte motivatie	O	O	O	O	O
Leraren met veel kennis en vaardigheden op het gebied van programeeronderwijs kunnen de meeste leerlingen helpen om programmeeronderwerpen te begrijpen	O	O	O	O	O
Programeeronderwijs geef ik net zo goed als andere vakken	O	O	O	O	O
Ik weet hoe ik leerlingen concepten uit het programeerdomein moet aanleren	O	O	O	O	O
Ik kan leerlingen zodanig begeleiden bij programmeeropdrachten, dat zij zelf antwoorden kunnen vinden op hun eigen vragen	O	O	O	O	O
Over het algemeen ben ik tevreden over de manier waarop ik programmeeronderwijs geef	O	O	O	O	O
Ik begrijp zelf de programmeerconcepten goed genoeg om de kinderen deze concepten op een effectieve manier te leren	O	O	O	O	O
Ik kan leerlingen uitleggen wat de onderliggende theorie is bij een programmeeropdracht	O	O	O	O	O
Ik ben over het algemeen in staat om programmeervragen van kinderen te beantwoorden	O	O	O	O	O
Ik heb de benodigde vakdidactische vaardigheden om les te geven in programmeren	O	O	O	O	O
Als mijn directeur of een collega bij een les aanwezig is, dan vind ik het prima als dat een programmeerles is	O	O	O	O	O
Als een leerling moeite heeft met een programmeerconcept, dan weet ik hoe ik de leerling moet helpen om het beter te begrijpen	O	O	O	O	O
Als ik programmeren geef vind ik het fijn als leerlingen vragen stellen	O	O	O	O	O
Ik weet wat ik moet doen om leerlingen voor programmeren te motiveren	O	O	O	O	O

Let op, de vragenlijst gaat door op de achterkant

Gegevens

13. Wat is uw geslacht?

- ☐ Man
- ☐ Vrouw
- ☐ Anders

14. Wat is uw leeftijd?

_____

15. Aan welke groep(en) geeft u les?

_____

16. Hoeveel jaar bent u werkzaam als leerkracht in het onderwijs?

- ☐ 0 - 2 jaar
- ☐ 2 - 5 jaar
- ☐ 5 - 10 jaar
- ☐ Meer dan 10 jaar
- ☐ Anders, namelijk _____

17. Op wat voor type school geeft u les?

- ☐ Openbare school
- ☐ School met godsdienstige of levensbeschouwelijke grondslag
- ☐ Jenaplan
- ☐ Montessori
- ☐ Daltonschool
- ☐ Vrijeschool
- ☐ Speciaal onderwijs
- ☐ Anders, namelijk _____

Tot slot

Bedankt voor het invullen van de vragenlijst!

Voor het vervolg van mijn onderzoek zoek ik nog leerkrachten. In dit onderzoek kom ik naar uw school voor een interview en geeft u zelf een aantal programmeerlessen in de klas. Het is hiervoor niet nodig om ervaring te hebben met programmeren. Het lesmateriaal voor de lessen krijgt u van mij.

Zou ik u hiervoor mogen benaderen? U zou mij er erg mee helpen.

Wanneer u benaderd wordt kunt u altijd, zonder gespecificeerde reden, deelname aan het onderzoek weigeren. De onderstaande gegevens zullen niet meegenomen worden in de data analyse en enkel gebruikt worden om u te benaderen omtrent onderzoek naar programmeren in de klas.

Naam _____

E-mail _____

School _____ in _____

B.2 Results

B.2.1 Demographics of participants

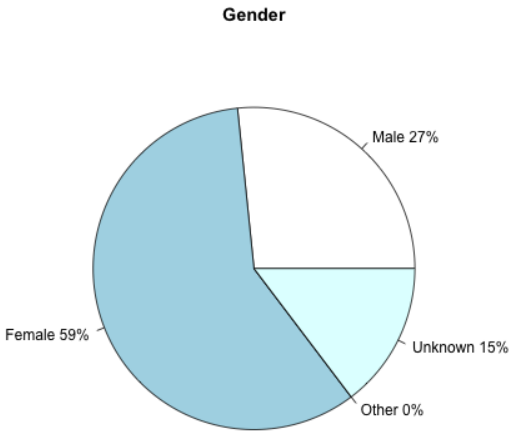


Figure B.1: Gender

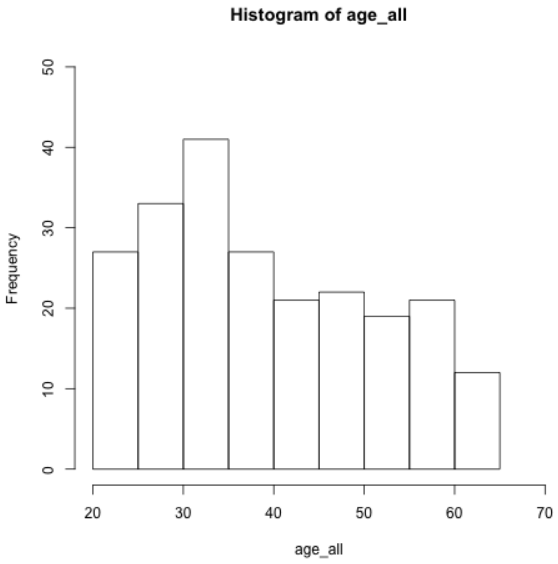


Figure B.2: Age

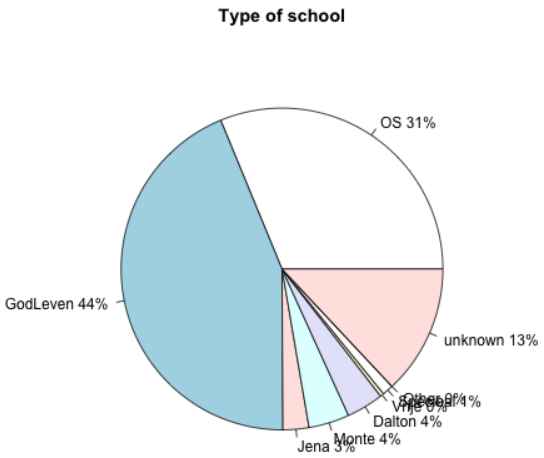


Figure B.3: Type of school

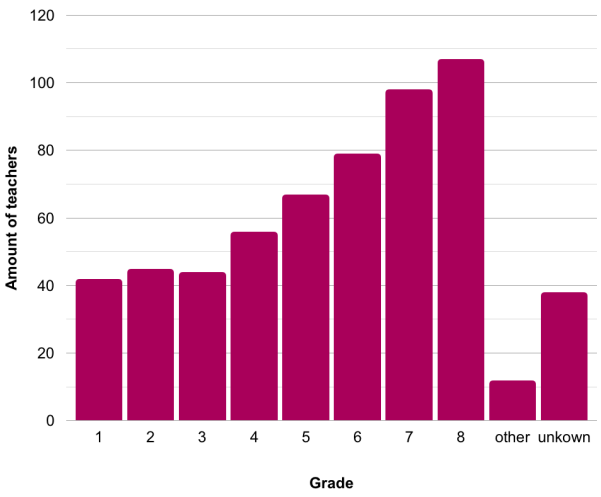


Figure B.4: Classes participants teach

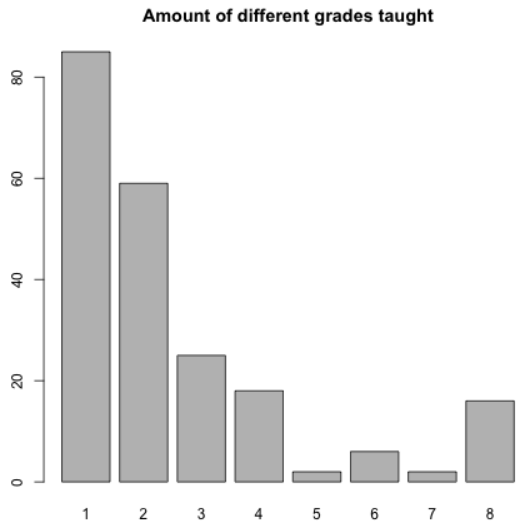


Figure B.5: Amount of grades participants teach

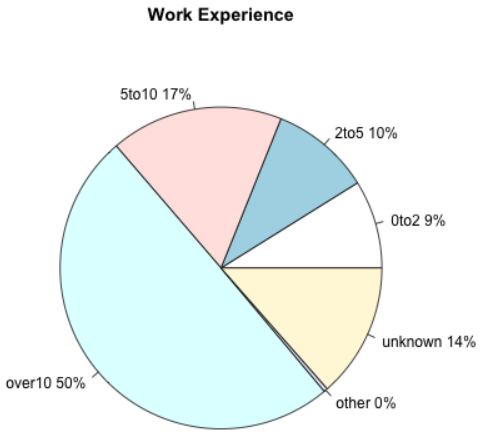


Figure B.6: Work experience

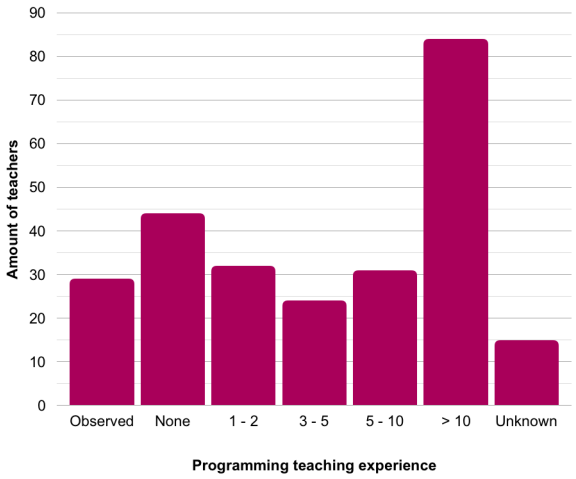


Figure B.8: Experience in teaching programming

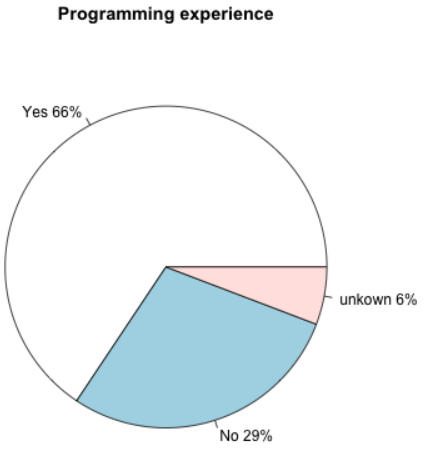


Figure B.7: Experience in programming

B.2.2 Familiarity of teaching methods

Table B.1: Frequency of response for teaching method per demographic

		Direct instruction				Discovery learning				
		Know	**Don't know**	**Use**	**Don't use**	**Know**	**Don't know**	**Use**	**Don't use**	**Total**
All		238	0	219	19	234	4	127	111	238
Gender	**Male**	65	0	61	4	63	2	30	35	65
	Female	136	0	123	13	136	0	79	57	136
	Other	0	0	0	0	0	0	0	0	0
	Unknown	37	0	35	2	35	2	18	19	37
Age	**Under 25**	14	0	14	0	14	0	8	6	14
	25-34	74	0	70	4	73	1	36	38	74
	35-44	46	0	42	4	45	1	25	21	46
	45-34	39	0	33	6	39	0	22	17	39
	>55	30	0	27	3	30	0	19	11	30
	Unknown	35	0	33	2	33	2	17	18	35
Grade	**1**	34	0	29	5	34	0	29	5	34
	2	36	0	29	7	36	0	29	7	36
	3	39	0	33	6	39	0	27	12	39
	4	51	0	45	6	51	0	28	23	51
	5	60	0	50	10	60	0	38	22	60
	6	72	0	62	10	71	1	39	33	72
	7	92	0	81	11	92	0	49	43	92
	8	99	0	88	11	98	1	56	43	99
	Other	10	0	8	2	10	0	6	4	10
	Unknown	37	0	35	2	35	2	19	18	37
Schooltype	**Openbaar**	75	0	68	7	75	0	40	35	75
	Godsdienstig	107	0	104	3	105	2	55	52	107
	Jenaplan	7	0	6	1	7	0	5	2	7
	Montessori	10	0	6	4	10	0	7	3	10
	Dalton	10	0	10	0	10	0	5	5	10
	Vrije school	1	0	1	0	1	0	1	0	1
	Special	3	0	3	0	3	0	2	1	3
	Other	0	0	0	0	0	0	0	0	0
	Unknown	34	0	32	2	32	2	16	18	34

Table B.2: Frequency of response for teaching method per demographic

		Direct instruction				Discovery learning				
		Know	**Don't know**	**Use**	**Don't use**	**Know**	**Don't know**	**Use**	**Don't use**	**Total**
All		238	0	219	19	234	4	127	111	238
Teaching experience	**0-2**	21	0	21	0	21	0	12	9	21
	2-5	24	0	22	2	23	1	13	11	24
	5-10	41	0	37	4	41	0	23	18	41
	>10	117	0	106	11	116	1	63	54	117
	Other	1	0	1	0	1	0	0	1	1
	Unknown	34	0	32	2	32	2	16	18	34
Programming Experience	**Yes (total)**	155	0	141	14	152	3	86	69	155
	Scratch	126	0	114	12	124	2	66	60	126
	Textual	52	0	44	8	51	1	31	21	52
	Hardware	74	0	65	9	73	1	43	31	74
	Other	28	0	27	1	28	0	16	12	28
	No	68	0	64	4	68	0	35	33	68
	Unknown	14	0	13	1	13	1	6	8	14
Teaching programming experience	**Yes (total)**	154	0	141	13	152	2	91	63	154
	1-2	29	0	28	1	28	1	16	13	29
	3-5	21	0	20	1	21	0	10	11	21
	5-10	28	0	28	0	27	1	11	17	28
	>10	76	0	65	11	76	0	54	22	76
	No (total)	69	0	64	5	68	1	31	38	69
	Guest	28	0	26	2	27	1	15	13	28
	Unknown	15	0	14	1	14	1	6	9	15

B.2.3 Self-efficacy

With this analysis, we can see whether different factors can be identified within a variable or component. The analysis tells us something about the uniqueness of the statements (which should be low) and the loadings onto factors (which should be high). Factor analysis also tells us whether the amount of factors chosen is sufficient, based on the variance the factors cover. When all 24 statements of the STEBI-NL are loaded onto two factors, as can be found in Figure B.2.3, we found that two factors are not sufficient. The items relating to STOE all have high uniqueness and loading is relatively low. The PSTE shows better results for both the uniqueness and loading. When only loading PSTE on a single factor, as shown in Figure B.2.3, the factor covers 0.670 of the variance. This still does not result in a single factor being sufficient. However, when adding more factors, all statements load over multiple factors.

```
Call:
factanal(x = data[, 57:80], factors = 2, rotation = "varimax")

Uniquenesses:
SelfEfficacy_A SelfEfficacy_B SelfEfficacy_C SelfEfficacy_D SelfEfficacy_E SelfEfficacy_F SelfEfficacy_G SelfEfficacy_H SelfEfficacy_I
         0.856          0.885          0.787          0.999          0.519          0.762          0.981          0.613          0.769
SelfEfficacy_J SelfEfficacy_K SelfEfficacy_L SelfEfficacy_M SelfEfficacy_N SelfEfficacy_O SelfEfficacy_P SelfEfficacy_Q SelfEfficacy_R
         0.825          0.973          0.999          0.334          0.273          0.336          0.389          0.145          0.237
SelfEfficacy_S SelfEfficacy_T SelfEfficacy_U SelfEfficacy_V SelfEfficacy_W SelfEfficacy_X
         0.146          0.365          0.414          0.190          0.598          0.486

Loadings:
               Factor1 Factor2
SelfEfficacy_A          0.379
SelfEfficacy_B          0.330
SelfEfficacy_C          0.461
SelfEfficacy_D
SelfEfficacy_E -0.133   0.681
SelfEfficacy_F -0.108   0.476
SelfEfficacy_G          0.134
SelfEfficacy_H          0.622
SelfEfficacy_I          0.478
SelfEfficacy_J          0.418
SelfEfficacy_K -0.162
SelfEfficacy_L
SelfEfficacy_M  0.816
SelfEfficacy_N  0.852
SelfEfficacy_O  0.814
SelfEfficacy_P  0.782
SelfEfficacy_Q  0.923
SelfEfficacy_R  0.871
SelfEfficacy_S  0.920
SelfEfficacy_T  0.796
SelfEfficacy_U  0.733  -0.220
SelfEfficacy_V  0.900
SelfEfficacy_W  0.631
SelfEfficacy_X  0.711

               Factor1 Factor2
SS loadings      8.075   2.044
Proportion Var   0.336   0.085
Cumulative Var   0.336   0.422

Test of the hypothesis that 2 factors are sufficient.
The chi square statistic is 486.91 on 229 degrees of freedom.
The p-value is 1.05e-20
```

Figure B.9: Factor Analysis of the adapted STEBI-NL

```
Call:
factanal(x = data[, 69:80], factors = 1, rotation = "varimax")

Uniquenesses:
SelfEfficacy_M SelfEfficacy_N SelfEfficacy_O SelfEfficacy_P SelfEfficacy_Q SelfEfficacy_R SelfEfficacy_S SelfEfficacy_T SelfEfficacy_U
         0.336          0.277          0.337          0.391          0.144          0.236          0.147          0.365          0.448
SelfEfficacy_V SelfEfficacy_W SelfEfficacy_X
         0.193          0.599          0.488

Loadings:
               Factor1
SelfEfficacy_M 0.815
SelfEfficacy_N 0.850
SelfEfficacy_O 0.814
SelfEfficacy_P 0.780
SelfEfficacy_Q 0.925
SelfEfficacy_R 0.874
SelfEfficacy_S 0.924
SelfEfficacy_T 0.797
SelfEfficacy_U 0.743
SelfEfficacy_V 0.899
SelfEfficacy_W 0.633
SelfEfficacy_X 0.715

               Factor1
SS loadings      8.038
Proportion Var   0.670

Test of the hypothesis that 1 factor is sufficient.
The chi square statistic is 221.44 on 54 degrees of freedom.
The p-value is 3.74e-22
```

Figure B.10: Factor Analysis of the adapted PSTE statements from STEBI-NL

		STOE			PSTE			
		Low	**Middle**	**High**	**Low**	**Middle**	**High**	**Total**
All		68	98	42	56	101	51	208
Gender	**Male**	12	29	21	11	30	21	62
	Female	55	68	18	44	69	28	141
	Other	0	0	0	0	0	0	0
	Unknown	1	1	3	1	2	2	5
Age	**Under 25**	4	8	2	3	9	2	14
	25-34	25	38	13	23	35	18	76
	35-44	19	18	9	9	22	15	46
	45-34	12	17	8	7	21	9	37
	>55	8	16	8	13	13	6	32
	Unknown	0	1	2	1	1	1	3
Grade	**1**	17	16	7	8	21	11	40
	2	15	20	7	10	22	10	42
	3	13	21	6	12	18	10	40
	4	17	29	5	13	26	12	51
	5	22	33	6	10	33	18	61
	6	25	36	9	13	32	25	70
	7	29	39	19	14	42	31	87
	8	33	46	18	17	49	31	97
	Other	3	7	2	3	6	3	12
	Unknown	1	2	2	2	1	2	5
Schooltype	**Openbaar**	27	36	15	22	38	18	78
	Godsdienstig	33	53	23	30	51	28	109
	Jenaplan	2	3	2	2	3	2	7
	Montessori	5	3	2	2	4	4	10
	Dalton	1	7	1	4	4	1	9
	Vrije school	0	1	0	0	1	0	1
	Speciaal	1	2	0	0	2	1	3
	Other	0	0	0	0	0	0	0
	Unknown	0	1	1	1	0	1	2

		STOE			PSTE			
		Low	**Middle**	**High**	**Low**	**Middle**	**High**	**Total**
All		68	98	42	56	101	51	208
Teaching experience	**0-2**	7	10	4	4	14	3	21
	2-5	5	12	7	7	11	6	24
	5-10	20	19	4	10	20	13	43
	>10	36	55	26	33	56	28	117
	Other	0	1	0	1	0	0	1
	Unknown	0	1	1	1	0	1	2
Programming Experience	**Yes (total)**	49	70	26	24	71	50	145
	Scratch	39	58	22	17	55	47	119
	Textual	16	25	6	6	18	23	47
	Hardware	26	35	12	7	32	34	73
	Other	12	12	5	2	15	12	29
	No	19	28	16	32	30	1	63
	Unknown	0	0	0	0	0	0	0
Teaching programming experience	**Yes (total)**	48	74	29	23	78	50	151
	1-2	12	14	4	14	16	0	30
	3-5	5	13	3	3	14	4	21
	5-10	5	14	6	2	17	6	25
	>10	26	33	16	4	31	40	75
	No (total)	21	24	13	33	23	2	58
	Guest	8	8	4	10	10	0	20
	Unknown	0	0	0	0	0	0	0

Appendix C

Questionnaire

C.1 Educational material

This appendix shows the educational material used in the experiment. The direct instruction manuals include the appendixes of the manual. The discovery learning manuals do not since they always include the same ‘Overzicht van blokken die leerlingen nodig hebben per categorie’ and ‘Stap voor stap:’ sections as the direct instruction manuals. Discovery learning manuals do not include the ’Opgave‘, ‘Opgave - extra’, and ‘Zelfstandig werken’.

Story telling - direct instruction

Verhaal vertellen - docentenhandleiding

In deze les gaan we een verhaal vertellen in Scratch. We laten hiervoor verschillende poppetjes om de beurt iets zeggen waardoor een dialoog ontstaat. Welke blokjes er tijdens de les gebruikt worden, is te vinden in de bijlage.

Voorbereiding	- De leerdoelen klaarzetten - Scratch-omgeving klaarzetten link: https://scratch.mit.edu/projects/292876407/editor/ - Zelfstandige oefeningen klaarzetten
Lesduur	60 minuten
Niveau	Groep 6, 7, 8. Geen eerdere Scratch-ervaring vereist.
Werkvorm	Klassikaal
Materiaal	Digibord en wisbordjes
Lesdoel	Bekend raken met Scratch door een verhaal te programmeren
Leerdoelen	• Aan het eind van de les zijn de leerlingen dusdanig bekend met de Scratch-omgeving dat ze weten waar ze blokjes kunnen vinden, waar ze blokjes kunnen neerzetten en hoe ze een programma kunnen uitvoeren • De leerlingen kunnen voorspellen wat een programma doet, dus in welke volgorde de poppetjes in het verhaal iets vertellen

Handig om te weten

Scratch is een gratis programmeeromgeving waarvoor je alleen een browser, zoals Chrome en Firefox, en een internet verbinding nodig hebt. Je hoeft dus niks te installeren. Scratch is gemaakt speciaal voor kinderen. Het programmeren gebeurt door verschillende blokjes in het programmeerveld te slepen en aan elkaar te klikken, net als Legoblokjes.

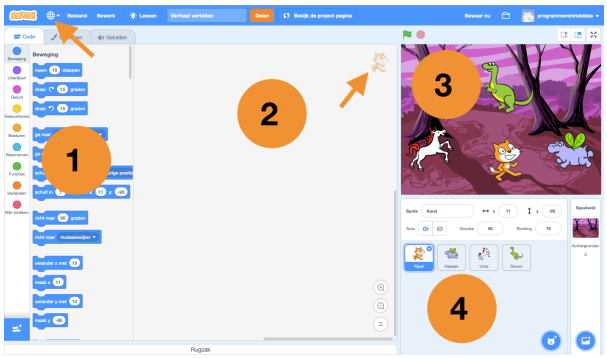
De les is gebaseerd op het materiaal van de MOOC Programmeren voor leerkrachten met Scratch. Dit is een gratis cursus die online te volgen is via https://www.edx.org/course/programmeren-voor-leerkrachten-met-scratch.

Voorbeeld van een eindprogramma: https://scratch.mit.edu/projects/291017419/editor/.
De stap-voor-stap uitwerking om deze les te maken is te vinden in de bijlage.

Introductie

Aan het begin van de les worden het lesdoel en leerdoelen gepresenteerd en doorgenomen. Hierbij wordt ook het belang van de les aangegeven, namelijk kunnen communiceren (een verhaal vertellen) via de computer. Verder helpt deze les om de volgende lessen te begrijpen. Vervolgens zal de leerkracht de Scratch-omgeving op het digibord aan de leerlingen laten zien en de belangrijkste onderdelen uitlichten.

De leerkracht opent de Scratch-omgeving (https://scratch.mit.edu/projects/292876407/editor/) en legt de volgende onderdelen uit:



1. Hier vind je alle blokken die we kunnen gebruiken om te programmeren. Met de blokken kun je de poppetjes iets laten doen. Mochten de blokken niet in het Nederlands zijn kun je bij de pijl links boven op het wereldbolletje klikken en de taal veranderen.
2. In dit veld kun je je blokjes neerzetten door ze uit de linkerkolom te slepen. Alles wat in dit grijze vlak staat vormt jouw programma. In de hoek (bij de pijl) zie je welk poppetje je op dit moment aan het programmeren bent. Er is nu nog niks geprogrammeerd, daar gaan we zometeen mee beginnen.
3. Rechtsboven zien je hoe het programma er nu uitziet. Wanneer je op de groene vlag klikt wordt je programma in dit scherm uitgevoerd. Om je programma stil te zetten kun je op de rode knop drukken.
4. Hier zien we welke poppetjes je op dit moment kunt programmeren. Deze worden in Scratch ‘sprite’ genoemd.

De leerkracht laat de leerlingen de verschillende onderdelen herhalen door de onderstaande vragen te stellen aan een willekeurige leerling.
- Wat zien we hier aan de linkerkant gebeuren?
- Wat is het grijze vlak in het midden?
- Waar wordt je programma uitgevoerd?
- Waar moet je klikken om een andere sprite te programmeren?
- Kun jij nog een keer herhalen wat we zien hier op het scherm?

De leerkracht herhaalt nu samenvattend wat er op het scherm te zien is en vertelt de leerlingen dat er klassikaal een programma gemaakt gaat worden. Verder vraagt de leerkracht de leerlingen: ‘Wat gingen we ook alweer precies doen?’.

Aan de slag

Nu de leerkracht de les heeft geïntroduceerd wordt het verhaal klassikaal gemaakt op het digibord. Hierbij is er veel interactie tussen de leerlingen en de leerkracht doordat de leerkracht kleine opdrachten tussendoor geeft en regelmatig dingen vraagt zoals;
- ‘Wat gebeurt er nu als ik op de start knop druk?’
- ‘Wat gaat er hierna gebeuren?’
- ‘Welke blokjes heb ik hiervoor nodig?’
- ‘Wat doet dit blokje?’

Daarnaast kan de leerkracht ervoor kiezen om een leerling de computer te laten besturen terwijl hij/zij zelf voor het digibord staat. Hierbij kan er regelmatig afgewisseld worden zodat meerdere leerlingen aan de beurt komen.

Verhaal verzinnen
Voordat we gaan beginnen met het programmeren gaan we eerst verzinnen welk verhaal we gaan vertellen. De personages staan al klaar op het scherm. We hebben Karel, Hassan, Uma en Devon. Laat de leerlingen allemaal een onderwerp opschrijven op hun wisbordje waarover de personages het gaan hebben en deze omhoog houden. Kies een onderwerp uit. Vraag nu aan de leerlingen hoe het gesprek begint: wie zegt er wat?

Het verhaal beginnen
We weten nu wat het onderwerp is van ons verhaal en wat er als eerst gezegd moet worden. Programmeer nu het eerste stukje van het verhaal en denk hierbij hardop voor.

Stel; het onderwerp is buitenspelen en het programma begint met Karel die zich voorstelt. Het hardop denken kan dan als volgt klinken:

“Ik wil dat Karel iets gaat zeggen. Ik kijk tussen mijn blokjes in de categorie ‘uiterlijken’ en zie daar het blokje ‘zeg hallo 2 sec’ staan. Ik klik er op om te zien wat dit blokje doet. Kijk, Karel zegt ‘hallo’ voor 2 seconden. Dat blokje kunnen we gebruiken, dus dat sleep ik naar het grijze vlak. Maar ik wil dat Karel meer zegt dan alleen ‘hallo’. Ik wil dat Karel zegt ‘Hallo, ik ben Karel’. Ik klik daarom op de tekst in het blokje en verander die naar ‘Hallo, ik ben Karel’. Als ik nu klik op het blokje, dan kijk zegt Karel inderdaad ‘Hallo, ik ben Karel’. Maar het is best onhandig om steeds op het blokje te klikken. Ik wil dat als ik op de groene vlag klik, mijn programma begint. Hiervoor heb ik nog een extra blokje nodig. Ik ga nu naar de categorie ‘Gebeurtenissen’ en kies daar voor het blokje ‘Wanneer er op de groene vlag wordt geklikt’. Ik sleep dit naar het grijze veld en let op, als ik deze dicht genoeg boven mijn andere blokje hou en daarna los laat dan zitten ze aan elkaar. Als ik nu op de groene vlag klik dan gaat Karel wel praten.”

Laat vervolgens dezelfde sprite nog iets zeggen. Vraag aan de leerlingen welke blokje er nu nodig is. Wanneer er een tweede  blokje aan het programma is toegevoegd, wordt er weer op de groene knop geklikt en merkt de leerkracht op dat de teksten nu wel snel achter elkaar verschijnen. Hiervoor kunnen we een ander blokje gebruiken, namelijk het  blokje. De leerkracht legt uit dat dit blokje het programma even op pauze kan zetten. Plak dit blokje tussen de twee  blokken die er al stonden en vraag aan de leerlingen: ‘Wat gebeurt er nu als ik op de groene vlag klik?’.

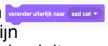
Een dialoog
Nadat Karel iets gezegd heeft willen we ook een andere sprite iets laten zeggen. Besluit met de klas welke sprite dit gaat worden en wat deze precies gaat zeggen. Vraag nu aan de klas om op hun wisbordje op te schrijven welke blokjes ze precies nodig hebben om de tweede sprite te laten praten. Laat de zojuist geprogrammeerde blokjes van Karel in beeld staan. De moeilijkheid bij deze opdracht is dat het programma moet beginnen met een  blok en dat er van sprite gewisseld moet worden.

Laat een leerling uitleggen wat hij/zij heeft bedacht en programmeer de voorgestelde blokken. Vraag voor dat er op de groene knop gedrukt wordt aan een andere leerling wat hij/zij denkt dat er gaat gebeuren. Druk vervolgens op de groene knop. Indien de timing onjuist is, laat de leerkracht de leerlingen bekijken wat er niet goed gaat en laat ze dit overleggen met hun buurman of buurvrouw. De leerkracht geeft daarna een leerling de beurt om een verbetering voor te stellen. Programmeer het voorstel en herhaal bovenstaande indien de nieuwe oplossing onjuist is.

Het verhaal uitbreiden
Het verhaal kan hierna uitgebreid worden. Hiervoor kan er afwisselend gevraagd worden aan de leerlingen om een voorstel voor een programma te schrijven op hun wisbordje of door direct te vragen welk blokje we als eerstvolgend nodig hebben. De leerkracht blijft hierbij regelmatig vragen wat de leerlingen verwachten dat er gebeurt. Evalueer gezamenlijk als er iets anders gebeurt dan verwacht werd.

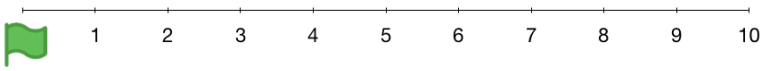
Zelfstandig oefenen
Wanneer de leerkracht het idee heeft dat de meeste leerlingen kunnen voorspellen wanneer welke sprite wat zegt kan de leerkracht stoppen met het verhaal uitbreiden en de leerlingen zelfstandig een aantal opgaven laten maken, zie bijlage. De opgave kan de leerkracht op het digibord laten

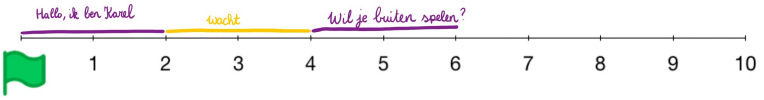
zien en de leerlingen kunnen de antwoorden op hun wisbordje opschrijven. De leerkracht zal de opgave bespreken. Wanneer de meeste leerlingen de opgave snappen kan de leerkracht door naar de volgende stap. Anders kan de leerkracht extra uitleg geven, nog een stuk voordoen in Scratch en/of extra opgave geven.

Emoties toevoegen
Deze stap is bedoeld voor wanneer de klas het  en  blokje onder de knie hebben. De sprites staan nu continu stil, wat het verhaal een beetje saai maakt. Gelukkig kunnen we het uiterlijk van de sprites aanpassen. Zorg ervoor dat er in de sprite Karel wordt geprogrammeerd. Laat dit aan de leerlingen zien door in de categorie 'Uiterlijken' het  blokje op te zoeken. Klik op dit blokje om te laten zien dat het uiterlijk verandert en selecteer verschillende uiterlijken in het dropdown menu. De leerkracht vraagt nu aan de leerlingen wanneer we Karel 'verdrietig' willen laten kijken of naar links laten draaien als hij tegen Uma praat. Voeg een  blokje toe en doe dit door hardop te denken. Besluit daarna of Karel nog ergens anders zijn uiterlijk moet veranderen. Ga daarna door de verschillende uiterlijken voor Devon heen en besluit met de leerlingen of Devon van uiterlijk moet veranderen. Doe hetzelfde voor Uma en Hassan.

Nog een aantal tips
De leerkracht zorgt ervoor dat er regelmatig op de groene vlag wordt geklikt. Elke keer als er een nieuw stukje is geprogrammeerd wordt er gecontroleerd of het verwachte resultaat ook echt bereikt is. Daarbij vraagt de leerkracht regelmatig wat leerlingen verwachten dat er gebeurt. Het is ook niet erg als een programma niet in één keer klopt omdat dit een mooie gelegenheid biedt om te evalueren wat er gebeurt en goed kritisch te zijn op het programma.

Het moeilijkste element in dit programma is de timing. Wanneer de leerlingen dit lastig vinden kan de leerkracht een tijdlijn tekenen, zoals hieronder. Vervolgens kunnen de gebeurtenissen in het programma in de tijdlijn getekend worden. Ook kan de leerkracht vragen of de leerlingen een tijdlijn willen tekenen van het programma tot dusver.





Afsluiting
In de laatste paar minuten van de les stelt de leerkracht een of meerdere van de volgende vragen:
- Wat heb je geleerd?
- Wat zou je hier nog meer over willen leren?
- Waar kun je het voor gebruiken?
- Wie zou je een compliment willen geven?
- Wat kan er beter in de volgende les?
- Wat vond je deze les juist erg goed gaan?
- Wat vind je nu nog lastig?
- Wat vond je het leukst aan de les?
- Noem drie woorden die goed bij deze les passen

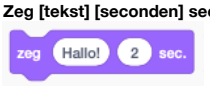
Daarnaast bespreekt de leerkracht het proces (werkhouding, werkpunten, opvallendheden) en herhaalt de leerkracht het lesdoel en de leerdoelen.

Bijlage

Overzicht van gebruikte blokken per categorie

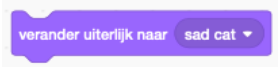
Uiterlijken

Zeg [tekst] [seconden] sec.



De sprite zegt de [tekst] voor [seconden] seconden.
Zeggen in Scratch betekent dat er een tekstballon verschijnt. Er komt dus *geen* geluid.

Verander uiterlijk naar [uiterlijk]



Het uiterlijk van de sprite verandert naar [uiterlijk].

Gebeurtenissen

Wanneer op de groene vlag wordt geklikt

Dit blokje kun je bovenaan een reeks blokjes gebruiken. Alle blokjes die aan dit blok zijn vast geplakt worden uitgevoerd wanneer er op de groene vlag wordt geklikt.

Besturen

Wacht [seconden] sec.

Het programma gaat [seconden] seconden op pauze. Het blokje onder het wacht blokje wordt dus na [seconden] seconden uitgevoerd in plaats van direct.

Opgave

Bij de opgave zie je steeds een programma met daarbij een meerkeuzevraag.

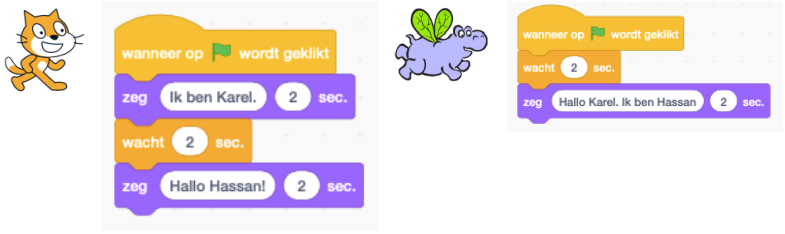
Karel **Programma van Karel**

Wat gebeurt er als er op de groene vlag wordt geklikt?
A: Karel zegt ‘Hoi!’ en gelijk daarna ‘Ik ben Karel.'
B: Karel zegt ‘Hoi!’, wacht 1 seconden en zegt daarna ‘Ik ben Karel.'
C: Karel zegt ‘Hoi!’, wacht 2 seconden en zegt daarna ‘Ik ben Karel.'
D: Niks

Karel **Programma van Karel**

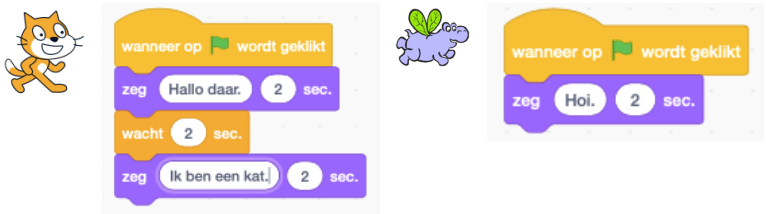
Wat gebeurt er als er op de groene vlag wordt geklikt?
A: Karel zegt ‘Het is lekker weer vandaag!’ en gelijk daarna ‘De zon schijnt!’
B: Karel zegt ‘Het is lekker weer vandaag!’, wacht 2 seconden en zegt daarna ‘De zon schijnt!’
C: Karel zegt ‘Hoi!’, wacht 2 seconden en zegt daarna ‘Ik ben Karel.'
D: Niks

Karel **Programma van Karel** **Hassan** **Programma van Hassan**



Wat gebeurt er als er op de groene vlag wordt geklikt?
A: Karel zegt 'Ik ben Karel', daarna zegt Hassan 'Hallo Karel. Ik ben Hassan'. Vervolgens zegt Karel: 'Hallo Hassan!'.
B: Karel zegt 'Ik ben Karel', wacht 2 seconden en zegt daarna 'Hallo Hassan!'. Vervolgens zegt Hassan 'Hallo Karel. Ik ben Hassan'.
C: Karel en Hassan beginnen tegelijkertijd te praten.
D: Niks

Karel **Programma van Karel** **Hassan** **Programma van Hassan**



Wat gebeurt er als er op de groene vlag wordt geklikt?
A: Karel zegt 'Hallo daar'. Daarna zegt Hassan 'Hoi.' Vervolgens zegt Karel 'Ik ben een kat.'
B: Karel zegt 'Hallo daar', wacht 2 seconden en zegt daarna 'Ik ben een kat.'. Vervolgens zegt Hassan: 'Hoi.'.
C: Karel en Hassan beginnen tegelijkertijd te praten.
D: Niks

Opgave - extra

Karel **Programma van Karel**

Wat gebeurt er als er op de groene vlag wordt geklikt?
A: Karel zegt 'Is het al tijd om te eten?' en wacht daarna 2 seconden.
B: Karel zegt 'Is het al tijd om te eten?'
C: Karel wacht 2 seconden en zegt daarna 'Is het al tijd om te eten?'
D: Niks

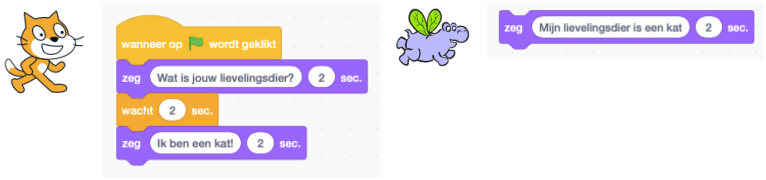
Karel **Programma van Karel**

Wat gebeurt er als er op de groene vlag wordt geklikt?
A: Karel zegt 'is gezien', wacht 2 seconden en zegt 'Wie niet weg is'.
B: Karel zegt 'Wie niet weg is', wacht 2 seconden en zegt 'is gezien'.
C: Karel zegt 'is gezien' en gelijk daarna zegt Karel 'Wie niet weg is'.
D: Niks

Karel **Programma van Karel**

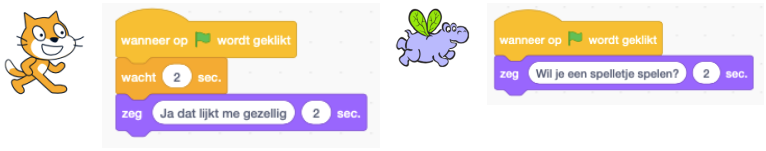
Wat gebeurt er als er op de groene vlag wordt geklikt?
A: Karel zegt 'Hassan', wacht 2 seconden en zegt daarna 'Hassan, waar ben je?'
B: Karel wacht 2 seconden, zegt 'Hassan', wacht 2 seconden en zegt daarna 'Hassan, waar ben je?'
C: Karel zegt 'Hassan' en zegt gelijk daarna 'Hassan, waar ben je?'
D: Niks

Karel **Programma van Karel** **Hassan** **Programma van Hassan**



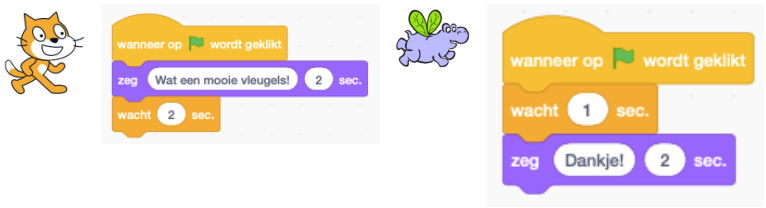
Wat gebeurt er als er op de groene vlag wordt geklikt?
A: Karel vraagt 'Wat is jouw lievelingsdier?'. Daarna zegt Hassan 'Mijn lievelingsdier is een kat'. Vervolgens zegt Karel 'Ik ben een kat!'.
B: Karel vraagt 'Wat is jouw lievelingsdier?', wacht 2 seconden en zegt daarna 'Ik ben een kat!'.
C: Karel en Hassan beginnen tegelijkertijd te praten.
D: Niks

Karel **Programma van Karel** **Hassan** **Programma van Hassan**



Wat gebeurt er als er op de groene vlag wordt geklikt?
A: Hassan zegt 'Wil je een spelletje spelen?'. Daarna zegt Karel 'Ja dat lijkt me gezellig'.
B: Karel wacht 2 seconden en zegt 'Ja dat lijkt me gezellig'. Daarna zegt Hassan 'Wil je een spelletje spelen?'
C: Karel en Hassan beginnen tegelijkertijd te praten.
D: Niks

Karel **Programma van Karel** **Hassan** **Programma van Hassan**



Wat gebeurt er als er op de groene vlag wordt geklikt?
A: Karel zegt 'Wat een mooie vleugels!'. Daarna zegt Hassan 'Dankje!'.
B: Karel zegt 'Wat een mooie vleugels!' en *voordat* Karel is uitgepraat zegt Hassan 'Dankje!'.
C: Karel en Hassan beginnen tegelijkertijd te praten.
D: Niks

Stap voor stap: verhaal vertellen

We gaan vandaag een verhaal programmeren in Scratch. Deze les is gebaseerd op de MOOC Programmeren voor leerkrachten met Scratch, week 2. In deze handleiding staat stap voor stap uitgelegd hoe je dit programma maakt: *https://scratch.mit.edu/projects/291017419/*.
Bij elke paragraaf staat ook een link naar een video uit de MOOC waar de stappen op gebaseerd zijn. Let op, de filmpjes gebruiken een oudere versie van Scratch waardoor het er allemaal iets anders uit ziet.

We gaan beginnen

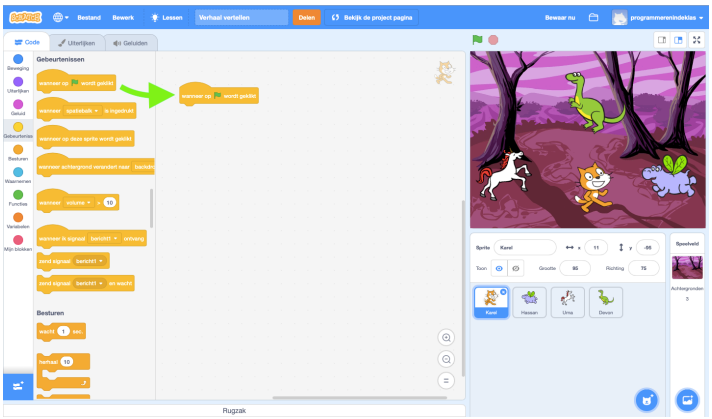
Video: https://www.youtube.com/watch?v=jUZPZoCzWCU

Ga naar https://scratch.mit.edu/projects/292876407/editor.

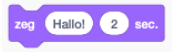
Vaak starten we het programma door op de groene vlag te klikken. Wanneer je nu op de groene vlag klikt gebeurt er nog helemaal niks.

Klik in de linkerkolom op de categorie ‘Gebeurtenissen’. Sleep vervolgens het 

blokje naar het middenveld.

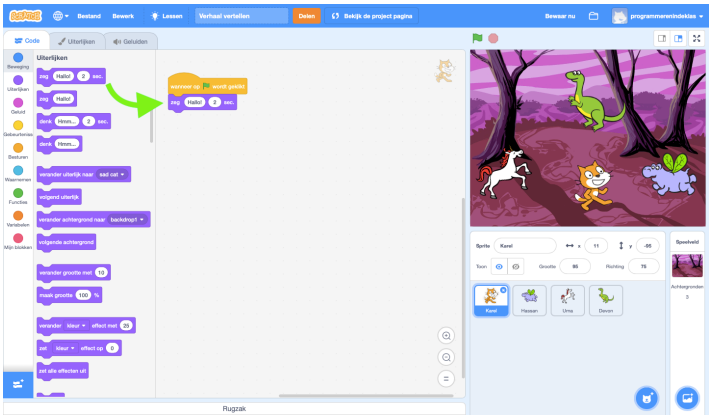


Wanneer er op de groene vlag geklikt wordt, willen we dat Karel de kat iets gaat zeggen. Ga naar de categorie ‘Uiterlijken’.

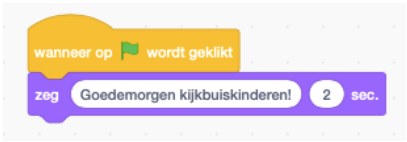
Sleep het 

blokje naar het grijze veld en plak deze onder het 

blok.



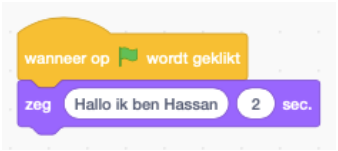
Klik nu maar eens op de groene vlag. Nu komt er een spreekballonnetje bij de kat. De kat zegt ‘Hallo’, maar dat is een beetje saai. Daarom maken we er iets leukers van, bijvoorbeeld ‘Goedemorgen kijkbuiskinderen’. Het programma ziet er nu dus zo uit:



Timing

Video: https://www.youtube.com/watch?v=Ue2VNrKrbpU

Nu gaan we een ander poppetje iets laten zeggen. Een poppetje in Scratch noemen we een sprite. Klik maar eens op Hassan het nijlpaard (rechts onderin). Het grijze vlak van Hassan is nog leeg. Laten we Hassan ook iets laten zeggen, bijvoorbeeld ‘Hallo ik ben Hassan’. We doen dat met de volgende blokjes:



Wat gebeurt er nu als je op de groene vlag klikt?

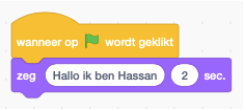
Dan gaan Karel en Hassan tegelijkertijd praten! Dat is natuurlijk niet zo netjes, om daar elkaar te praten.

We gaan dit oplossen met een 

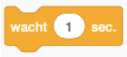
blokje. Je kunt dit blokje vinden in de categorie ‘Besturen’.

Je kunt dit blokje tussen de twee blokken zetten die we al hadden. Wanneer je een blok over blokjes heen sleept die er al staan, maken ze ruimte.

Probeer maar uit en sleep het 

blok tussen 

in.

Klik nu weer op de groene vlag en kijk of het programma nu al goed werkt of dat we misschien nog iets moeten aanpassen in het 

blok.

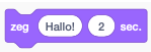
Karel de kat praat voor 2 seconden dus we moeten Hassan eerst 2 seconden laten wachten voordat Hassan kan beginnen met praten.

Antwoorden

Video: https://www.youtube.com/watch?v=P-Kfo7npkTE

We willen graag dat verschillende sprites (poppetjes) op elkaar gaan reageren zodat het verhaal wat leuker gaat worden. We gaan Karel de Kat laten reageren op Hassan de Nijlpaard. Klik hiervoor op Karel (rechts onderin) zodat we Karel verder kunnen programmeren.

Plak een 

en 

blok onder de blokjes die we al hadden staan.

Zorg ervoor dat Karel lang genoeg wacht om Hassan eerst uit te laten praten. Laat Karel bijvoorbeeld vragen: ‘Wil je met mij spelen?’. Controleer of het programma doet wat je wilt door op de groene vlag te klikken.

Laat vervolgens Hassan reageren op Karel, bijvoorbeeld door te zeggen: ‘Sorry vriend, ik ga liever programmeren’. Probeer dit zelf te programmeren voordat je verder gaat naar de volgend bladzijde.

Het programma ziet er nu zo uit:

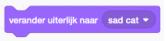
Karel

Hassan

Uiterlijken

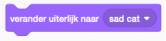
Video: https://www.youtube.com/watch?v=oNkl3-F8AcY

Karel vindt het jammer dat Hassan liever gaat programmeren en wordt daar een beetje sip van. In Scratch kunnen we de sprites van uiterlijk laten veranderen om het verhaal wat levendiger te maken.

Ga naar de categorie 'Uiterlijken' en slepen het 

naar het grijze vlak. Wanneer je op 'verdrietig' klikt verschijnen er allerlei verschillende uiterlijken die bij Karel de kat horen. We kunnen nu Karel verdrietig laten kijken na dat Hassan antwoord heeft gegeven.

Karel wacht dus eerst op het antwoord van Hassan, wat we programmeren met een 

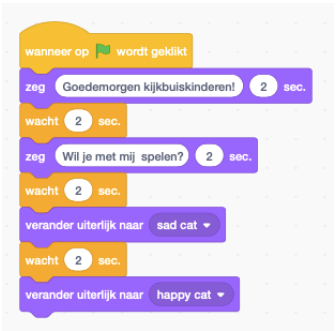
blok. Daarna plakken we het 

aan de andere blokken vast.

Druk op de groene vlag om te zien wat er precies gebeurt nu.

Karel gaat niet bij de pakken neer zetten en kijkt na 2 seconden weer vrolijk. Probeer dat zelf te programmeren.

Het programma van Karel ziet er nu dus zo uit:



Story telling - discovery learning

Verhaal vertellen - docentenhandleiding

In deze les gaan we een verhaal vertellen in Scratch. We laten hiervoor verschillende poppetjes om de beurt iets zeggen waardoor een dialoog ontstaat.

Voorbereiding	Scratch-omgeving klaarzetten https://scratch.mit.edu/projects/291017419/editor/ en https://scratch.mit.edu/projects/292876407/editor/
Lesduur	60 minuten
Niveau	Groep 6, 7, 8. Geen eerdere Scratch-ervaring vereist.
Werkvorm	Introductie klassikaal daarna gaan de leerlingen in tweetallen zelfstandig aan de slag
Materiaal	Digibord en laptops
Lesdoel	Bekend raken met Scratch door een verhaal te programmeren
Leerdoelen	• Aan het eind van de les zijn de leerlingen dusdanig bekend met de Scratch omgeving dat ze weten waar ze blokjes kunnen vinden, waar ze blokjes kunnen neerzetten en hoe ze een programma kunnen uitvoeren • De leerlingen kunnen voorspellen wat het programma doet. In dit geval kunnen ze dus vertellen wanneer welk poppetjes iets vertelt in het verhaal

Handig om te weten

Scratch is een gratis programmeeromgeving waarvoor je alleen een browser, zoals Chrome en Firefox, en een internet verbinding nodig hebt. Je hoeft dus niks te installeren. Scratch is gemaakt speciaal voor kinderen. Het programmeren gebeurt door verschillende blokjes in het programmeerveld te slepen en aan elkaar te klikken, net als Legoblokjes.

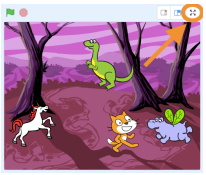
De les is gebaseerd op het materiaal van de MOOC Programmeren voor leerkrachten met Scratch. Dit is een gratis cursus die online te volgen is via https://www.edx.org/course/programmeren-voor-leerkrachten-met-scratch.

Elke les is er een demonstratie beschikbaar. In de bijlage is de stap-voor-stap uitwerking te vinden van hoe dit programma gemaakt is. Tijdens de les mogen de leerlingen natuurlijk hun eigen verhaal verzinnen.

Zet voor het begin van de les alvast de laptops klaar. Je kunt bijvoorbeeld twee leerlingen vijf minuten voor de les vragen de laptops vast neer en aan te zetten. Zo kunnen de leerlingen sneller van start.

Introductie

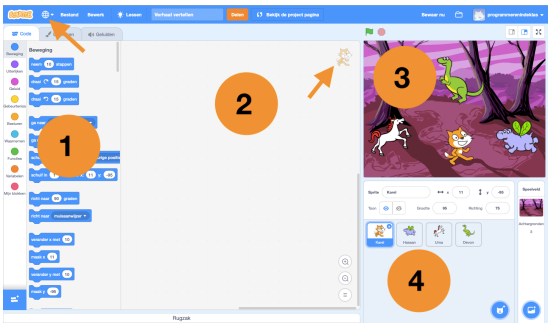
Aan het begin van de les wordt er een opwarm oefening gedaan waarna het lesdoel wordt geïntroduceerd. Vervolgens zal de leerkracht de Scratch omgeving op het digibord aan de leerlingen laten zien en de belangrijkste onderdelen uitlichten. Ook zal de leerkracht een voorbeeld programma laten zien.



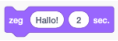
De leerkracht opent de link https://scratch.mit.edu/projects/291017419/ en vergroot het programma door op de vergrootknop te klikken (zoals hieronder bij de oranje pijl te zien is). De leerkracht vertelt dat we vier verschillende personages zien die een avontuur gaan beleven. De leerkracht stelt vervolgens vragen aan de leerlingen zoals:
- Waarom zijn deze personages hier?
- Wat gaan ze doen?
- Wat zullen ze zeggen?

Vervolgens introduceert de leerkracht het lesdoel en demonstreert het programma door op de groene vlag te klikken.

Hierna gaat de leerkracht uitleg geven over Scratch. Hiervoor opent de leerkracht de link https://scratch.mit.edu/projects/292876407/editor/ en legt de volgende onderdelen uit:



1. Hier vind je alle blokken die we kunnen gebruiken om te programmeren. Met de blokken kun je de poppetjes iets laten doen. Mochten de blokken niet in het Nederlands zijn kun je bij de pijl links boven op het wereldbolletje klikken en de taal veranderen.
2. In dit veld kun je je blokjes neerzetten door ze uit de linkerkolom te slepen. Alles wat in dit grijze vlak staat vormt jouw programma. In de hoek (bij de pijl) zie je welk poppetje je op dit moment aan het programmeren bent. Er is nu nog niks geprogrammeerd, daar gaan we zometeen mee beginnen.
3. Rechtsboven zien je hoe het programma er nu uitziet. Wanneer je op de groene vlag klikt wordt je programma in dit scherm uitgevoerd. Om je programma stil te zetten kun je op de rode knop drukken.
4. Hier zien we welke poppetjes je op dit moment kunt programmeren. Deze worden in Scratch ‘sprite’ genoemd.

Je kunt een blokje altijd uitproberen door een keer te klikken op een blok. De leerkracht sleept het 

blok naar het grijze vlak en klikt er een keer op om te laten zien wat er gebeurt.

Aan de slag

De leerlingen gaan na de introductie zelfstandig in tweetallen aan de slag. De rol van de leerkracht is het coachen van de leerlingen. De leerlingen gaan dus zelf ontdekken wat ze kunnen doen in Scratch om zo hun verhaal te vormen.

De leerkracht zorgt ervoor dat de begin link https://scratch.mit.edu/projects/292876407/editor/ zichtbaar is op het digibord.

Verder kan de leerkracht kan als tip mee geven dat de leerlingen kunnen beginnen met het uitproberen van de blokken in de categorie ‘Uiterlijken’. Dit zijn de paarse blokken.

Wanneer een leerling vast zit kan de leerkracht de volgende vragen stellen:
- Wat wil je dat je programma doet?
- Welke blokken heb je hiervoor nodig?
- Kun je me uitleggen wat de blokken in je programmeer veld nu doen?
- Wat is er anders in de uitvoer dan je verwacht had?

De leerkracht zegt dus **niet** voor wat de oplossing kan zijn.

Wanneer een leerling niet weet wat hij/zij moet programmeren kan de leerkracht vragen stellen zoals:
- Waar speelt het verhaal zich af?
- Wat is het onderwerp?
- Wat gaan de sprites doen?
- Wat gebeurt er als eerst?
- Wie zegt er wat?

Een valkuil tijdens de les kan zijn dat een leerling te veel ter gelijker tijd wil programmeren. De leerkracht kan de leerlingen helpen met het opdelen in kleinere stukken. Bijvoorbeeld door te vragen 'Welke stap moet er als eerste gebeuren? Wat heb je hiervoor nodig?'.

De leerkracht probeert de leerlingen te stimuleren om regelmatig op de groene vlag te klikken. Zodat de leerlingen elke keer als ze een nieuw stuk hebben geprogrammeerd controleren of het verwachte resultaat ook echt bereikt is. Het is niet erg als een programma niet in één keer klopt omdat dit een mooie gelegenheid biedt om te evalueren wat er gebeurt en goed kritisch te zijn op het programma.

Een veelvoorkomende beginnersfout bij het programmeren in Scratch is dat leerlingen vergeten hun blokken te koppelen aan een zogenoemde 'gebeurtenis'. In dit programma wordt het verhaal gestart door op de groene vlag te klikken en zal er dus een 

blok bovenaan het programma geplakt moeten worden.

Het kan gebeuren dat een tweetal leerlingen niet goed samenwerkt doordat één leerling dominant is en degene is die de laptop continu bestuurt. De leerkracht kan in dit geval besluiten een timer te zetten en de leerlingen verplichten om om de 5 minuten te wisselen van bestuurder.

Afsluiting

Tijdens de afsluiting mogen leerlingen bij elkaars projecten kijken. De leerlingen stellen daarbij vragen aan elkaar. De leerkracht kan één of meerdere van de volgende vragen mee geven:
- Wat vind je het leukst aan het programma?
- Wat was het lastigst?
- Als je meer tijd zou hebben, wat zou je dan toevoegen of veranderen?

Funny stories - direct instruction

Gek verhaal - docentenhandleiding

In deze les gaan we een programma maken dat zelf een gek verhaal maakt. Hiervoor maken we lijstjes met dieren, dingen om te doen en plekken.

Voorbereiding	- De leerdoelen klaarzetten - Scratch-omgeving klaarzetten op het digibord https://scratch.mit.edu/projects/292970518/editor - Scratch-omgeving voor de leerlingen https://scratch.mit.edu/projects/292980678/editor - Zelfstandige oefeningen klaarzetten / uitprinten
Lesduur	60 minuten
Niveau	Groep 6, 7, 8. De leerlingen hebben de Scratch-omgeving al een keer gezien
Werkvorm	Introductie klassikaal en daarna werken in twee tallen
Materiaal	Digibord, wisbordjes en laptops
Lesdoel	Een gek verhaal maken met onder ander lijstjes van dieren en activiteiten
Leerdoelen	• Leerlingen kunnen voorbeelden geven van wat lijsten zijn • Leerlingen kunnen een lijst maken, items toevoegen en de lijst leeg maken • Leerlingen kunnen uitleggen wat er gebeurt als je een willekeurig item uit de lijst kiest

Handig om te weten

De les is gebaseerd op het materiaal van de MOOC Programmeren voor leerkrachten met Scratch. Dit is een gratis cursus die online te volgen is via https://www.edx.org/course/programmeren-voor-leerkrachten-met-scratch.

Voorbeeld van een eindprogramma: https://scratch.mit.edu/projects/292939083/editor/.
De stap-voor-stap uitwerking om deze les te maken is te vinden in de bijlage.

Zet voor het begin van de les alvast de laptops klaar. Je kunt bijvoorbeeld twee leerlingen vijf minuten voor de les vragen de laptops vast neer en aan te zetten. Zo kunnen de leerlingen sneller van start.

Klassikaal

Aan het begin van de les vraagt de leerkracht aan de leerlingen wat ze vorige programmeerles hebben gedaan. Vervolgens worden het lesdoel en de leerdoelen gepresenteerd en doorgenomen.

Hierna stelt de leerkracht de volgende vraag: ‘Wat voor soort lijstjes ken jij?’ Dit mogen dus lijstjes uit de ‘echte’ wereld zijn. Laat de leerlingen de lijstjes opschrijven op hun wisbordjes. Als de leerlingen het lastig vinden om dingen te verzinnen geeft de leerkracht een voorbeeld: een klassenlijst. De leerkracht vraagt de leerlingen om de bordjes om hoog te houden en bespreekt welke lijstjes ze hebben opgeschreven en concludeert dat we lijstjes veel gebruiken. Daarnaast beschrijft de leerkracht een lijst als een opsomming van dingen en benadrukt vervolgens dat lijstjes ook goed van pas kunnen komen bij het programmeren.

De leerkracht opent de link https://scratch.mit.edu/projects/292970518/editor en vraagt de leerlingen welke onderdelen we op het scherm zien (blokken, programmeervlak, programma uitvoer en sprites).

Nu de leerkracht de les heeft geïntroduceerd wordt het begin van het programma klassikaal gemaakt op het digibord. De leerkracht gaat het eerste stuk van het programma voordoen en doet

dit door hardop te denken. De leerkracht maakt een nieuwe lijst aan (dieren) en laat zien hoe het eerste dier aan de lijst wordt toegevoegd. De leerkracht vraagt de leerlingen ook allemaal één dier op hun wisbordje te schrijven. Vervolgens voegt de leerkracht een aantal door de leerlingen gekozen dieren toe en controleert tussendoor het begrip door vragen te stellen zoals:

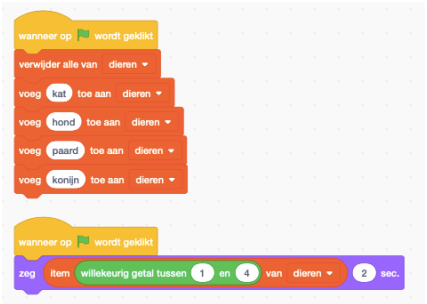
- Wat gebeurt er nu als ik op de groene vlag druk?
- Welke blokje heb ik nodig om een nieuw dier aan de lijst toe te voegen?
- Wat doet dit blokje?
- Wat zit er nu in de lijst?

Vervolgens klikt de leerkracht een aantal keer op de groene vlag en vraagt de leerlingen of hun iets opvalt aan de lijst. De leerkracht kan hier hinten naar de lengte van de lijst. De lijst wordt namelijk steeds langer omdat deze nooit leeg wordt gemaakt. Nu voegt de leerkracht het  blokje toe door hard op te denken. Voordat hij/zij op de groene vlag klikt vraagt hij/zij de leerlingen wat er nu in de lijst zit. Hierbij kan gebruik worden gemaakt van de wisbordjes.

Nu is het tijd om iets met de lijst te gaan doen. De leerkracht voegt een nieuw  blokje toe en denkt hardop dat het tijd wordt om Dani iets te laten zeggen. De leerkracht vraagt de leerlingen welk blokje we hiervoor kunnen gebruiken. Vervolgens gebruikt de leerkracht het  blokje om Dani een item uit het lijstje te laten zeggen. De leerkracht veranderd het itemnummer en vraagt aan een willekeurige leerling wat ze nu verwacht dat Dani gaat zeggen.

De leerkracht benoemd dat het nog niet echt een spannend verhaal is en brengt het  blokje in het programmeerveld. De leerkracht legt uit wat willekeurig is. Na dit blokje op de juiste plek te hebben gezet, vraagt de leerkracht of de leerlingen op willen schrijven op hun wisbordje welk dier Dani gaat noemen. De leerkracht demonstreert wat er gebeurt wanneer hij/zij op de groene vlag klikt.

Het programma ziet er ongeveer zo uit:



Dit is het programma waar de leerlingen tijdens het zelfstandig werken mee starten.

Het laatste onderdeel voordat de leerlingen zelfstandig aan het werk gaan is het beantwoorden van een aantal opgave, zie bijlage. Deze kunnen op het digibord worden weergeven en de leerlingen kunnen de antwoorden op hun wisbordje schrijven. De leerkracht controleert vervolgens de antwoorden. Wanneer de meeste leerlingen het juiste antwoord hebben gegeven geeft de leerkracht geen uitleg meer. Wanneer een groter deel van de klas de vraag niet juist heeft begrepen geeft de leerkracht extra uitleg en geeft hij/zij eventueel nog een extra opgave.

Zelfstandig werken

De leerlingen werken in tweetallen aan hun programma. Ze kunnen dit programma openen via de link https://scratch.mit.edu/projects/292980678/editor. Vervolgens kunnen ze aan de slag met de zelfstandig werk opgave die op het digibord weer worden gegeven of worden uitgedeeld. Tijdens

het zelfstandig werken loopt de leerkracht langs de verschillende tweetallen en vraagt de leerlingen om hun programma en de gebruikte blokjes uit te leggen. Wanneer iets onduidelijk is kan de leerkracht extra uitleg geven aan de individuele leerling of klassikaal.

Bij het zelfstandig werken is het risico dat leerlingen afdwalen en bijvoorbeeld de sprite van uiterlijk laten veranderen. Vraag de kinderen terug aan de slag te gaan met de opgave.
Bij het maken van lijsten kan het zijn dat leerlingen lang bezig zijn met heel veel verschillende dingen aan de lijsten toevoegen. Wijs de leerlingen dan op de opgave waarin staat dat ze maximum 5 items aan een lijst mogen toevoegen.

Wanneer leerlingen vragen hebben is het goed om eerst te controleren of ze de opgave gelezen en begrepen hebben. Wanneer leerlingen vragen hebben over elementen van Scratch waar de les niet over gaat kan de leerkracht besluiten de vraag niet te beantwoorden maar de leerlingen terug te sturen naar de opgave.

Een valkuil bij het gebruik van het 

blokje, is dat leerlingen het bereik niet aanpassen. Leerlingen laten dan het ‘tussen de 1 en 10’ laten staan terwijl ze maar 5 dingen in de lijst hebben staan. Wanneer het willekeurig getal 6 of hoger is, is er geen item en voert Scratch het blokje niet uit. Dani zegt dan dus niks.

Het kan gebeuren dat een tweetal leerlingen niet goed samenwerkt doordat één leerling dominant is en degene is die de laptop continu bestuurt. De leerkracht kan in dit geval besluiten een timer te zetten en de leerlingen verplichten om om de 5 minuten te wisselen van bestuurder.

Afsluiting

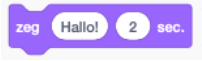
In de laatste paar minuten van de les, vraagt de leerkracht aan de leerlingen om de les samen te vatten. De leerkracht vraagt hierbij ook wat de leerlingen vandaag geleerd hebben. Daarnaast herhaalt de leerkracht het lesdoel en de leerdoelen.

Bijlage

Overzicht van blokken per categorie

Uiterlijken

Zeg [tekst] [seconden] sec.



De sprite zegt de [tekst] voor [seconden] seconden.
Zeggen in Scratch betekent dat er een tekstballon verschijnt. Er komt dus *geen* geluid.

Gebeurtenissen

Wanneer op de groene vlag wordt geklikt

Dit blokje kun je bovenaan een reeks blokjes gebruiken. Alle blokjes die aan dit blok zijn vast geplakt worden uitgevoerd wanneer er op de groene vlag wordt geklikt.

Besturen

Herhaal [getal]

Alle blokjes binnen het herhaal blok worden [getal] keer uitgevoerd.

Variabelen

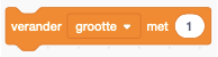
[variabele naam]

Dit blokje is een variabele waar een getal of stukje tekst in kan zijn opgeslagen.

Maak [variabele] [waarde]

Wijs [waarde] toe aan [variabele].
Met dit blokje kun je de variabele ook hernoemen en verwijderen. Klik hiervoor op [variabele].

Verander [variabele] met [getal]



Verander [variabele] met [getal]. [getal] kan zowel een positief als een negatief getal zijn. Wanneer [variabele] een stukje tekst is wordt [variabele] overschreven met waarde 1.

Opgave

Geef drie voorbeelden van een lijst.

Wanneer je op de groene vlag klikt, wat zit er dan in de lijst boodschappen?
A: noedels, aardappel, broccoli, koekjes
B: koekjes, broccoli, aardappel, noedels
C: noedels
D: Niks

Wanneer je op de groene vlag klikt, wat zit er dan in de lijst boodschappen?
A: banaan
B: mandarijn, thee, banaan
C: banaan, thee, mandarijn
D: Niks

Wanneer je <u>twee keer</u> de groene vlag klikt, wat zit er dan in de lijst boodschappen?
A: peer, tomaat, komkommer
B: peer, peer, tomaat, tomaat, komkommer, komkommer
C: peer, tomaat, komkommer, peer, tomaat, komkommer
D: Niks

Wat zegt Dani wanneer je op de groene vlag klikt?
A: hond
B: kat
C: paard
D: dat kun je niet weten

Wat zegt Dani wanneer je op de groene vlag klikt?
A: hond
B: kat
C: paard
D: dat kun je niet weten

Opgave - Extra

Maak een lijst met verschillende soorten fruit.

Verzin een lijst een schrijf op wat er in die lijst kan zitten.

Veranderd de lijst die je verzonnen hebt? Waarom?
Voorbeeld voor de leerkracht: Klassenlijst, veranderd elk jaar. Puntenlijst van een wedstrijd, veranderd tijdens de wedstrijd maar na de wedstrijd niet meer.

Wanneer je op de groene vlag klikt, wat zit er dan in de lijst boodschappen?
A: koffie, appel, vis
B: koffie, appel
C: vis
D: Niks

Wanneer je op de groene vlag klikt, wat zit er dan in de lijst dieren?
A: hond, kat, paard
B: hond
C: kat, paard
D: Niks

Wanneer je <u>twee keer</u> de groene vlag klikt, wat zit er dan in de lijst boodschappen?
A: hond, paard, kat
B: hond, paard, kat, hond, paard, kat
C: hond, hond, paard, paard, kat, kat
D: Niks

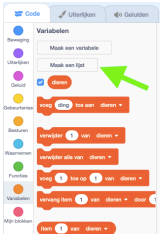
Wat zegt Dani wanneer je op de groene vlag klikt?
A: tomaat
B: sla
C: ei
D: dat kun je niet weten

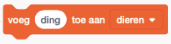
Wat zegt Dani wanneer je op de groene vlag klikt?
A: tomaat
B: sla
C: ei
D: dat kun je niet weten

Zelfstandig werken

1. Welke dingen om te doen kun jij verzinnen? Denk bijvoorbeeld aan praten en dansen. Verzin er maximaal vijf.

2. Maak een nieuwe lijst in Scratch en noem deze ‘dingen om te doen’. Een nieuwe lijst kun je maken met de knop ‘Maak een lijst’, zoals je ook hier bij de groene pijl ziet staan.

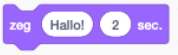


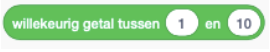
3. Voeg de dingen om te doen die je verzonnen hebt toe aan je lijst ‘dingen om te doen’. Je kunt hiervoor het  blokje gebruiken.

4. Bespreek samen wat je denkt dat er nu gebeurt als je op de groene vlag klikt. En wat gebeurt er als je twee keer op de groene vlag klikt? Doet het programma iets anders dan je wilt? Pas het programma dan aan.

5. We gaan Dani nu niet alleen een willekeurig dier maar ook een willekeurig ding om te doen laten zeggen. Weet jij hoe dit moet?

Tip: je kunt deze blokken hiervoor gebruiken:





6. Voer het programma een aantal keer uit om te testen of alles goed geprogrammeerd is.

Dani zegt nu losse woordjes, dat is nog niet zo spannend. We gaan er nu zinnen van maken.

7. Laat Dani beginnen met: ‘De hoofdpersoon van het verhaal is een …’. Hierna zegt ze het willekeurige dier en plaats. Je programma ziet er dan ongeveer zo uit:

8. Je kunt het  blokje gebruiken om tekst samen te voegen. Verander het laatste zeg-blokje zodanig dat Dani de zin begint met: ‘De hoofdpersoon gaat ‘.

9. Voeg een nieuwe lijst toe aan je programma, namelijk ‘plekken’. Voeg maximaal 5 plekken toe aan deze nieuwe lijst en laat je hoofdpersoon een willekeurig ding doen op een willekeurige plek.

10. Je mag je verhaal nu zelf uitbreiden met nieuwe lijsten en zinnen. Succes!

Voorbeeld een gek verhaal

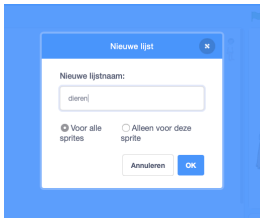
We gaan vandaag een verhaal programmeren in Scratch. Deze les is gebaseerd op de MOOC Programmeren voor leerkrachten met Scratch, week 4. In deze handleiding staat stap voor stap uitgelegd hoe je dit programma maakt: https://scratch.mit.edu/projects/292939083/.
Bij elke paragraaf staat ook een link naar een video uit de MOOC waar de stappen op gebaseerd zijn. Let op, de filmpjes gebruiken een oudere versie van Scratch waardoor het er allemaal iets anders uit ziet.

De eerste lijst

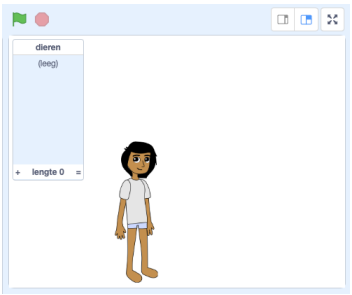
Video: https://www.youtube.com/watch?v=4NEBLDNxh1U

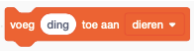
Ga naar https://scratch.mit.edu/projects/292970518/editor.

We gaan beginnen met de eerste lijst maken. Daarvoor gaan we naar de categorie 'Variabelen'. Er staat hier een knop 'Maak een lijst'. Wanneer je hier op klikt krijg je een popup zoals hieronder te zien is. Hier kun je de lijst een naam geven. De eerste lijst die we maken, noemen we 'dieren'.



Klik op 'ok'. Er gebeuren nu twee dingen. Je ziet dat er in de categorie 'Variabelen' allemaal nieuwe blokjes zijn toegevoegd. Verder zie je in je programma (rechtsboven) een lijst verschijnen genaamd 'dieren'. Deze lijst is nu nog helemaal leeg en heeft lengte 0.



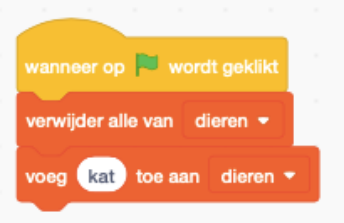
Sleep het blokje 

in het grijze vlak en klik een keer op dit blokje. Je ziet nu dat aan de lijst 'dieren' het woord 'ding' is toegevoegd.

Maak de lijst weer leeg door het 

blokje in het grijze vlak te slepen en een keer op dit blokje te klikken.

Programmeer nu het volgende: Wanneer je op de groene vlag klikt dan wordt de lijst 'dieren' helemaal leeg gemaakt en vervolgens wordt er het dier 'kat' aan de lijst 'dieren' toegevoegd. Probeer dit eerst zelf te programmeren en ga daarna door naar de volgende bladzijde. Vergeet niet op de groene vlag te klikken om het programma te testen.



Het programma ziet er nu zo uit. Voeg nog een aantal dieren toe aan de lijst, bijvoorbeeld ‘hond’, ‘paard’ en ‘konijn’.

Tip: Wanneer je ctrl ingedrukt houdt wanneer je op een blokje klikt komt er de optie om een kopie te maken van je blokje. Dan hoef je niet steeds nieuwe blokjes uit de linkerkolom te slepen.

Items

Video: https://www.youtube.com/watch?time_continue=1&v=PNy7ovdSvXY

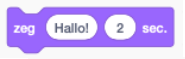
Behalve dingen in een lijst stoppen, kunnen we nog niet zo veel met dit programma. Daarom gaan we nu kijken naar hoe we items uit een lijst kunnen halen.

Hiervoor gebruiken we een speciaal blokje, namelijk 

. Je ziet dat dit blokje nergens aan vast kan worden geplakt omdat dit blokje alleen maar ronde randen heeft.

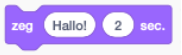
Sleep het 

blokje naar het grijze vlak en klik erop. Het blokje laat nu weten welk item er op plek 1 in de lijst ‘dieren’ zit.

Ga naar de categorie ‘Uiterlijken’ en sleep een 

blok onder je reeks met blokjes zodat deze aan elkaar vast plakken.

Sleep nu het 

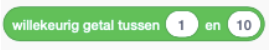
blok *in* het 

blok op de plek van de tekst.

Je programma ziet er nu dus zo uit:

Wat gebeurt er nu als je op de groene vlag klikt? Probeer maar uit!

Je kunt Dani dus een woord uit de lijst ‘dieren’ laten zeggen. Het getal 1 kun je ook aanpassen zodat Dani een ander item zegt. Nu gaan we Dani elke keer een willekeurig item uit de lijst laten zeggen.

Ga hiervoor naar ‘Functies’ en kies het blokje 

. Dit blokje kiest elke keer een ander willekeurig getal tussen de 1 en 10. Maar we hebben geen 10 dieren in het lijstje staan maar 4. Verander de 10 dus in een 4.

Vervolgens kunnen we dit blokje op de plek van het item nummer in 

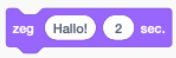
stoppen.

Het programma ziet er nu zo uit:

Klik een aantal keer op de groene vlag om te zien of Dani nu elke keer een ander willekeurig dier uit het lijstje kiest.

Van één los woorden naar een zin

We kunnen ons programma nu een willekeurig dier laten zeggen. De volgende stap is om het willekeurige dier in een verhaal te zetten.

Voeg een 

blok toe boven het zeg blok wat er al staat. Veranderd de tekst naar ‘De hoofdpersoon is een …’. Druk op de groene vlag. Dat klinkt al meer als een verhaal.

Maak nu een nieuwe lijst ‘dingen om te doen’ en stop daar verschillende werkwoorden in, bijvoorbeeld ‘lopen’, ‘dansen’, ‘zwemmen’ en ‘denken’. Je kunt hiervoor een nieuwe blokkenreeks starten. Het vlak waarin je programmeert ziet er nu ongeveer zo uit:

Laat Dani nu ook iets zeggen over wat de hoofdpersoon gaat doen. De onderstaande blokken die we al eerder hebben gezien kunnen we hiervoor gebruiken.

Een ander blok wat we hier goed voor kunnen gebruiken is het 

blok. Dit blok plakt twee stukken tekst aan elkaar. Op deze manier kunnen we de volgende zin maken:
'De hoofdpersoon gaat [willekeurig ding doen]'.

Uiteindelijk ziet het voorbeeld programma er zo uit

Funny stories - discovery learning

Gek verhaal - docentenhandleiding

In deze les gaan we een programma maken dat zelf een gek verhaal maakt. Hiervoor maken we lijstjes met dieren, dingen om te doen en plekken.

Voorbereiding	Scratch-omgeving klaarzetten (digibord) https://scratch.mit.edu/projects/292939083/editor Scratch-omgeving klaarzetten (leerlingen) https://scratch.mit.edu/projects/292970518/editor
Lesduur	60 minuten
Niveau	Groep 6, 7, 8. De leerlingen hebben de Scratch-omgeving al een keer gezien
Werkvorm	Introductie klassikaal en daarna werken in twee tallen
Materiaal	Digibord, pen en papier of wisbordje en laptops
Lesdoel	Een gek verhaal maken met onder ander lijstjes van dieren en activiteiten
Leerdoelen	• Leerlingen kunnen voorbeelden geven van wat lijsten zijn • Leerlingen kunnen een lijst maken, items toevoegen en de lijst leeg maken • Leerlingen kunnen uitleggen wat er gebeurt als je een willekeurig item uit de lijst kiest

Handig om te weten

De les is gebaseerd op het materiaal van de MOOC Programmeren voor leerkrachten met Scratch. Dit is een gratis cursus die online te volgen is via https://www.edx.org/course/programmeren-voor-leerkrachten-met-scratch.

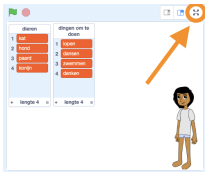
Elke les is er een demonstratie beschikbaar. In de bijlage is de stap-voor-stap uitwerking te vinden van hoe dit programma gemaakt is. Tijdens de les mogen de leerlingen natuurlijk hun eigen verhaal verzinnen.

Zet voor het begin van de les alvast de laptops klaar. Je kunt bijvoorbeeld twee leerlingen vijf minuten voor de les vragen de laptops vast neer en aan te zetten. Zo kunnen de leerlingen sneller van start.

Introductie

Aan het begin van de les wordt er een opwarm oefening gedaan. De leerlingen werken hierbij in tweetallen. Een van de twee leerlingen schrijft een superheld en een kleur op. De andere leerling schrijft een werkwoord en een voorwerp op. Ze mogen niet overleggen. Nadat alle leerlingen iets op hebben geschreven maken ze de volgende zin met de vier dingen de ze hebben opgeschreven:
[Superheld] gaat [werkwoord] met een [kleur] [voorwerp].
De leerlingen maken zo een gekke zin, bijvoorbeeld ‘Superman gaat dansen met een groene stoel’. De leerkracht koppelt het maken van een gekke zin aan het lesdoel: het maken van een gek verhaal.

Vervolgens zal de leerkracht een voorbeeldprogramma laten zien https://scratch.mit.edu/projects/292939083/editor. De leerkracht kan het spel eerst vergroten door op de knop met de pijltjes naar buiten te klikken, zie onderstaande afbeelding, en vervolgens op de groene knop om het verhaal te demonstreren.



Aan de slag

De leerlingen gaan na de introductie zelfstandig in tweetallen aan de slag. De rol van de leerkracht is het coachen van de leerlingen. De leerlingen gaan dus zelf ontdekken wat ze kunnen doen in Scratch om zo hun verhaal te vormen.

De leerkracht zorgt ervoor dat de begin link https://scratch.mit.edu/projects/292970518/editor zichtbaar is op het digibord. In dit project is de eerste lijst genaamd 'dieren' alvast aangemaakt voor de leerlingen. De leerkracht kan de leerlingen als tip meegeven om te beginnen met het uitproberen van de blokken in de categorie 'Variabelen'. Dit zijn de donker oranje blokken. Het uitproberen kan bijvoorbeeld door op een blok te klikken en te ontdekken wat er dan gebeurt.

Wanneer een leerling vast zit kan de leerkracht de volgende vragen stellen:
- Wat wil je dat je programma doet?
- Welke blokken heb je hiervoor nodig?
- Kun je me uitleggen wat de blokken in je programmeerveld nu doen?
- Wat is er anders in de uitvoer dan je verwacht had?

De leerkracht zegt dus **niet** voor wat de oplossing kan zijn.

Het werken met lijsten is het lastigst in deze les. De leerkracht kan vragen stellen over lijsten aan de leerlingen zoals:
- Zie je iets in de categorie 'variabelen' staan waar je meerdere dingen in kunt opslaan?
- Hoe ziet een lijst er in de 'echte' wereld uit?
- Wat zit er op dit moment in je lijst?
- Wat gebeurt er als je dit blokje toevoegt aan je programma?

Afsluiting

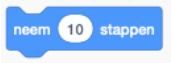
Tijdens de afsluiting mogen leerlingen bij elkaars projecten kijken. De leerlingen stellen daarbij vragen aan elkaar. De leerkracht kan één of meerdere van de volgende vragen mee geven:
- Wat vind je het leukst aan het programma?
- Wat was het lastigst?
- Als je meer tijd zou hebben, wat zou je dan toevoegen of veranderen?

Bijlage

Overzicht van blokken per categorie

Beweging

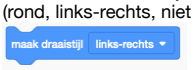
Neem [getal] stappen

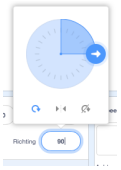


De sprite beweegt een stukje (naar de rechterkant van het scherm). Hoe groter [getal] hoe groter het stuk is dat de sprite verplaatst. [getal] kan ook een negatief getal zijn, dan beweegt de sprite de andere kant op (dus richting de linker kant van het scherm).

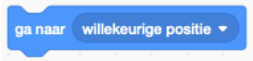
Draai [getal] graden

De sprite draait [getal] graden. Let op, de sprite kan op verschillende manieren draaien. Je kunt de draaistijl aanpassen door op de 'Richting' van een sprite te klikken en de draaistijl te selecteren (rond, links-rechts, niet draaien) of met het blokje





Ga naar [lokatie]



Verplaats de sprite naar:
- willekeurige positie
- muisaanwijzer
- andere sprite in het programma

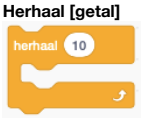
Gebeurtenissen

Wanneer op de groene vlag wordt geklikt

Dit blokje kun je bovenaan een reeks blokjes gebruiken. Alle blokjes die aan dit blok zijn vastgeplakt worden uitgevoerd wanneer er op de groene vlag wordt geklikt.

Besturen

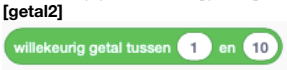
Herhaal [getal]



Alle blokjes binnen het herhaal-blok worden [getal] keer uitgevoerd.

Functies

Willekeurig getal tussen [getal1] en [getal2]



Dit blokje kiest een willekeurig getal tussen [getal1] en [getal2] waarbij [getal1] en [getal2] ook gekozen kunnen worden.

Variabelen

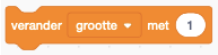
[variabele naam]

Dit blokje is een variabele waar een getal of stukje tekst in kan zijn opgeslagen.

Maak [variabele] [waarde]

Wijs [waarde] toe aan [variabele].
Met dit blokje kun je de variabele ook hernoemen en verwijderen. Klik hiervoor op [variabele].

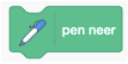
Verander [variabele] met [getal]



Verander [variabele] met [getal]. [getal] kan zowel een positief als een negatief getal zijn. Wanneer [variabele] een stukje tekst is wordt [variabele] overschreven met waarde 1.

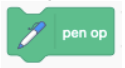
Pen

Pen neer



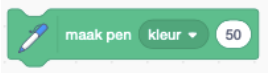
Zet de pen neer op het digitale papier. Wanneer de sprite beweegt zal deze een streep achterlaten.

Pen op



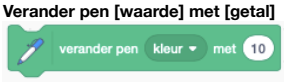
Haalt de pen van het digitale papier af. De sprite zal geen streep meer achterlaten.

Maak pen [waarde] met [getal]



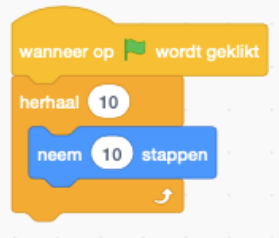
Wijs een [getal] toe aan de onderstaande [waarde] zodat de lijn er anders uitziet;
- Kleur
- Verzadiging
- Helderheid
- Doorzichtigheid

Verander pen [waarde] met [getal]



Verander de lijn van de pen door een van de volgende [waarde] aan te passen naar [getal]:
- Kleur
- Verzadiging
- Helderheid
- Doorzichtigheid

Opgave - Herhalen



Hoeveel stappen worden er genomen als er op de groene vlag wordt geklikt?
A: 0
B: 10
C: 100
D: Geen van bovenstaande

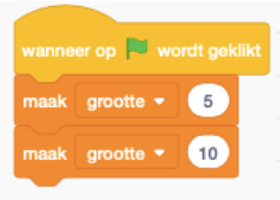
Hoeveel stappen worden er genomen als er op de groene vlag wordt geklikt?
A: 2
B: 10
C: 30
D: Geen van bovenstaande

Teken het figuur dat verschijnt nadat er op de groene vlag is geklikt.

Opgave - Variabelen

Welk getal zit er in 'grootte' nadat er op de groene vlag is geklikt?
A: 0
B: 10
C: 20
D: Geen van bovenstaande

Welk getal zit er in 'grootte' nadat er op de groene vlag is geklikt?
A: 0
B: 1
C: 5
D: Geen van bovenstaande



Welk getal zit er in 'grootte' nadat er op de groene vlag is geklikt?
A: 0
B: 10
C: 15
D: Geen van bovenstaande

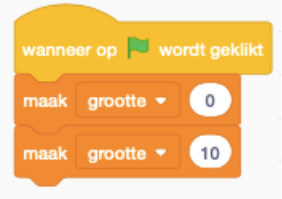
Welk getal zit er in 'grootte' nadat er op de groene vlag is geklikt?
A: 5
B: 10
C: 15
D: Geen van bovenstaande

Hoeveel stappen worden er genomen als er op de groene vlag wordt geklikt?
A: 5
B: 10
C: 60
D: Geen van bovenstaande

Opgave - Extra

Welk getal zit er in 'grootte' nadat er op de groene vlag is geklikt?
A: 0
B: 10
C: 11
D: Geen van bovenstaande

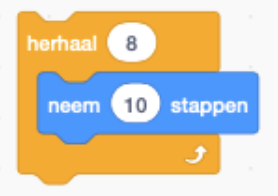
Welk getal zit er in 'grootte' nadat er op de groene vlag is geklikt?
A: 1
B: 5
C: 6
D: Geen van bovenstaande



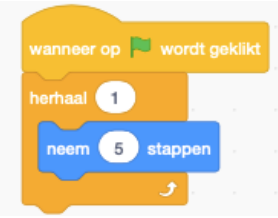
Welk getal zit er in 'grootte' nadat er op de groene vlag is geklikt?
A: 0
B: 10
C: 20
D: Geen van bovenstaande

Welk getal zit er in 'grootte' nadat er op de groene vlag is geklikt?
A: 5
B: 10
C: 15
D: Geen van bovenstaande

Welk getal zit er in 'grootte' nadat er op de groene vlag is geklikt?
A: 0
B: 1
C: 10
D: Geen van bovenstaande



Hoeveel stappen worden er genomen als er op de groene vlag wordt geklikt?
A: 8
B: 10
C: 80
D: Geen van bovenstaande



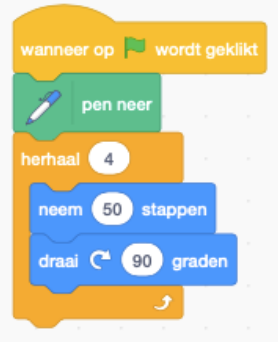
Hoeveel stappen worden er genomen als er op de groene vlag wordt geklikt?
A: 0
B: 1
C: 5
D: Geen van bovenstaande

Hoeveel stappen worden er genomen als er op de groene vlag wordt geklikt?
A: 5
B: 10
C: 60
D: Geen van bovenstaande

Hoeveel stappen worden er genomen als er op de groene vlag wordt geklikt?
A: 0
B: 10
C: 20
D: Geen van bovenstaande

Hoeveel stappen worden er genomen als er op de groene vlag wordt geklikt?
A: 0
B: 10
C: 40
D: Geen van bovenstaande

Teken het figuur dat verschijnt nadat er op de groene vlag is geklikt.



Teken het figuur dat verschijnt nadat er op de groene vlag is geklikt.

Teken het figuur dat verschijnt nadat er op de groene vlag is geklikt.

Zelfstandig werken

Open https://scratch.mit.edu/projects/293100444/editor/

In de klas hebben we het aantal stappen dat het potlood maakt in de variabele ‘grootte’ opgeslagen. We gaan nu ook een variabele maken voor het aantal graden dat het potlood draait.

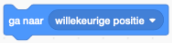
1. Maak een variabele ‘graden’
2. Maak ‘graden’ gelijk aan ‘15’
3. Gebruik de variabele ‘graden’ in het blokje 

Jullie programma ziet er nu zo uit:

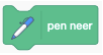
We tekenen nu elke keer precies hetzelfde figuurtje. We gaan daar verandering in brengen.

4. Verander de variabele ‘grootte’. Maak er eens 5 of 25 van. Wat verwachten jullie dat er nu gebeurt? Druk daarna op de groene vlag.
5. Verander de variabele ‘graden’. Maak er eens 120 of 150 van. Wat verwachten jullie dat er nu gebeurt? Druk daarna op de groene vlag.
6. Verander de ‘grootte’ en ‘graden’ zo dat jullie een figuur krijgen dat jullie mooi vinden.

We tekenen nu steeds één figuur en gaan daar nu een tweede figuur bij tekenen.

7. Het tekenen van een figuur deden we door het nemen van stappen en draaien te herhalen. Kunnen jullie het tekenen van een nieuw figuur onderaan het huidige programma toevoegen?
8. Zien jullie nog steeds één figuur in plaats van twee? Dat komt waarschijnlijk doordat het potlood de twee figuren over elkaar heen tekent. Zet een 

blokje tussen de twee herhaal-blokken.
9. Tussen de twee figuren wordt er een streep getekend. Kunnen jullie ervoor zorgen dat deze streep niet meer getekend wordt?
Tip: gebruik de blokken 

en 

Je programma ziet er waarschijnlijk nu ongeveer zo uit:

10. Leg aan elkaar uit wat er nu gebeurt als er op de groene vlag wordt geklikt.

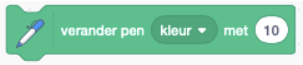
11. Verander in het **eerste** figuur de variabele 'grootte' elke keer nadat het potlood gedraaid heeft. Verander 'grootte' eens met 1, 2 en 5. Voorspel tussendoor wat je verwacht dat er gaat gebeuren en test het programma uit door op de groene vlag te klikken.

12. Verander in het **tweede** figuur de variabele 'graden' elke keer nadat het potlood gedraaid heeft. Verander 'graden' eens met 0.1, 1 en 5. Voorspel tussendoor wat je verwacht dat er gaat gebeuren en test het programma uit door op de groene vlag te klikken.

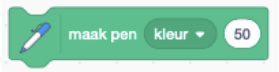
Je kunt nog van alles aan je programma aanpassen. Kies samen één of meerdere aanpassingen die jullie nog willen maken:

- Voeg een extra figuur toe
- Pas het aantal herhalingen aan
- Laat het eerste figuur op een willekeurige positie beginnen
- Pas 'grootte' van één of meerdere figuren aan
- Pas 'graden' van één of meerdere figuren aan
- Gebruik het blokje 

voor het aantal herhalingen, grootte en/of graden
- De kleur van de pen is ook een variabele. Scratch heeft daar een speciale maak- en verander-blokje voor, namelijk



en



Kun jij de kleur van de pen laten veranderen?

Voorbeeld Spirograaf

We gaan vandaag een tekening maken in Scratch. Deze les is gebaseerd op de MOOC Programmeren voor leerkrachten met Scratch, week 5. In deze handleiding staat stap voor stap uitgelegd hoe je dit programma maakt: https://scratch.mit.edu/projects/293022513/editor/.
Bij elke paragraaf staat ook een link naar een video uit de MOOC waar de stappen op gebaseerd zijn. Let op, de filmpjes gebruiken een oudere versie van Scratch waardoor het er allemaal iets anders uit ziet.

Tekenen!

Video: https://www.youtube.com/watch?v=fGjq_8CDZpI

Ga naar https://scratch.mit.edu/projects/293023636/editor.

Net zoals in de vorige lessen willen we het programma starten door op de groene vlag te klikken. Sleep het 

in het grijze vlak.

We gaan beginnen met het potlood laten bewegen.

Ga naar 'Beweging' en plak het 

blok aan 

. Klik een aantal keer op de groene vlag. Het potlood beweegt wel, maar tekent nog niet.

Ga naar de categorie 'Pen' en kies het blokje 

. Dit blokje zet de pen op het digitale papier neer. Sleep het blokje boven het 

blokje. Klik nu een aantal keer op de groene vlag. Jahoor het potlood tekent!

Wis de getekende lijn door op 

te klikken.

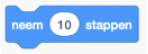
We gaan nu een rondje tekenen. Om een rondje te tekenen moeten we het potlood kunnen draaien. Ga naar 'Beweging' en plak het 

blokje onder de blokjes die je tot nu toe hebt.

Je programma ziet er nu dus zo uit: Klik een flink aantal keer achter elkaar op de groene vlag en daar verschijnt een rondje!

Herhalen

Video: https://www.youtube.com/watch?v=dxt5_A6UnPE

Zo vaak op de groene vlag klikken, dat is niet heel handig. We willen liever dat de computer dat werk voor ons doet. We willen steeds de blokjes 

en 

herhalen.

Bij de categorie 'Besturen' zijn er blokjes die precies dat voor ons doen. Sleep het blok 

om de twee blauwe blokjes heen. De code ziet er nu zo uit:

Druk maar eens op de wisknop en daarna op de groene vlag. Nu verschijnt er met één klik een stuk van de cirkel. Wanneer we niet 10 keer herhalen maar 24 keer is de cirkel zelfs met één klik op de groene vlag helemaal rond.

Variabele

Video: https://www.youtube.com/watch?v=3cFDzjBozLU
Video: https://www.youtube.com/watch?time_continue=161&v=zi_LH08n22s

We gaan nu variabelen gebruiken om de grootte van de cirkel op te slaan en makkelijk te kunnen hergebruiken.

Ga naar de categorie 'Variabelen' en klik op 'Maak een variabele'. Geef de variabele een duidelijke naam zoals 'grootte'. Klik daarna op 'ok'.

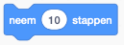
Sleep het blokje 

boven het herhaalblokje. Bedenk wat er nu gaat gebeuren en klik daarna op de groene vlag.

Er is nog niks veranderd! En dat komt doordat we de variabele 

verder nog nergens gebruiken.

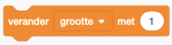
Plak de variabele 

nu in het 

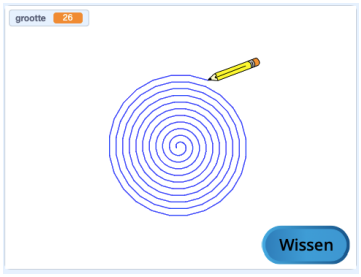
blok en druk op de groene vlag. Wat gebeurt er nu?

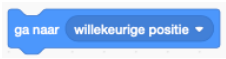
Geef ‘grootte’ nu eens andere waarden door de 0 in het 

blok aan te passen naar 5, 25 of 100.

Pak nu een 

blok, vervang 1 door 0.1 en en stop deze in het herhaal blok. Verder kun je het aantal herhalingen verhogen, bijvoorbeeld naar 250. Je programma en het resultaat zien er dan zo uit:



Nu start het nieuwe figuur elke keer op de plek waar het potlood geëindigd is of waar je het potlood hebt neergezet. We gaan dit willekeurig maken. Dit kan met een blokje in de categorie ‘Bewegen’, namelijk 

.

Zet dit blokje helemaal bovenaan de reeks blokjes, direct onder de 

. Klik vervolgens een aantal keer op de groene vlag.

We krijgen nu figuren over het hele scherm, alleen wel nog met een streep ertussen. Hiervoor moeten we de pen van het digitale papier af halen voordat we de pen gaan verplaatsen. Dit kunnen we doen met 

.

Het programma ziet er nu zo uit:

Drawing a Spirograph - discovery learning

Spirograaf- docentenhandleiding

In deze les gaan we verschillende figuren tekenen met Scratch.

Voorbereiding	Scratch-omgeving klaarzetten (digibord) https://scratch.mit.edu/projects/293022513/editor/ Scratch-omgeving klaarzetten (leerlingen) https://scratch.mit.edu/projects/293023636/editor/
Lesduur	60 minuten
Niveau	Groep 6, 7, 8. De leerlingen hebben de Scratch-omgeving al een keer gezien
Werkvorm	Introductie klassikaal en daarna werken in tweetallen
Materiaal	Digibord, pen en papier of wisbordjes en laptops
Lesdoel	Een spirograaf maken in Scratch
Leerdoelen	• Leerlingen kunnen een blokkenreeks herhalen met een herhaal-blok • Leerlingen kunnen een variabele gebruiken om een getal in op te slaan • Leerlingen kunnen de Scratch-pen gebruiken om figuurtjes te tekenen

Handig om te weten

De les is gebaseerd op het materiaal van de MOOC Programmeren voor leerkrachten met Scratch. Dit is een gratis cursus die online te volgen is via https://www.edx.org/course/programmeren-voor-leerkrachten-met-scratch.

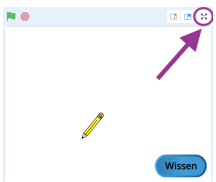
Elke les is er een demonstratie beschikbaar. In de bijlage is de stap-voor-stap uitwerking te vinden van hoe dit programma gemaakt is. Tijdens de les mogen de leerlingen natuurlijk hun eigen verhaal verzinnen.

Zet voor het begin van de les alvast de laptops klaar. Je kunt bijvoorbeeld twee leerlingen vijf minuten voor de les vragen de laptops vast neer en aan te zetten. Zo kunnen de leerlingen sneller van start.

Introductie

Aan het begin van de les wordt er een opwarmoefening gedaan. De leerkracht vraagt de leerlingen verschillende vormen te tekenen die een spirograaf kan maken. De leerkracht koppelt het maken van een spirograaf met de hand aan het lesdoel: het maken van een spirograaf in Scratch.

Vervolgens zal de leerkracht een voorbeeldprogramma laten zien https://scratch.mit.edu/projects/293022513/editor/. De leerkracht kan het spel eerst vergroten door op de knop met pijltjes naar buiten te klikken, zie onderstaande afbeelding, en vervolgens op de groene knop om het programma te demonstreren.



Aan de slag

De leerlingen werken in tweetallen aan hun programma. Ze kunnen dit programma openen via de https://scratch.mit.edu/projects/293023636/editor/. De link kan op het digibord gezet worden.

Tijdens het zelfstandig werken heeft de leerkracht een coachende rol. De leerlingen gaan dus zelf ontdekken wat ze kunnen doen in Scratch om zo hun eigen figuren te maken.

De leerkracht kan als tip meegeven dat de leerlingen kunnen beginnen met het uitproberen van de blokken in de categorie 'Pen'. Dit zijn de donkergroene blokken. Het uitproberen kan bijvoorbeeld door op een blok te klikken en te ontdekken wat er dan gebeurt.
Wanneer een leerling vast zit kan de leerkracht de volgende vragen stellen:
- Wat wil je dat je programma doet?
- Welke blokken heb je hiervoor nodig?
- Kun je me uitleggen wat de blokken in je programmeerveld nu doen?
- Wat is er anders in de uitvoer dan je verwacht had?

De leerkracht zegt dus **niet** voor wat de oplossing kan zijn.

De leerkracht stimuleert de leerlingen om een herhaal-blok en variabele te gebruiken door vragen te stellen zoals:
- Je gebruikt nu heel veel dezelfde blokjes. Is er een blokje waardoor je dit korter kan programmeren?
- Wat gebeurt er in je programma?
- Zijn er blokjes die steeds herhaald worden?

Wanneer een leerling niet weet hij/zij moet programmeren kan de leerkracht de leerling stimuleren verschillende figuren te maken en daarbij ook te kijken naar het gebruik van verschillende kleuren.

Afsluiting

Tijdens de afsluiting mogen leerlingen bij elkaars projecten kijken. De leerlingen stellen daarbij vragen aan elkaar. De leerkracht kan één of meerdere van de volgende vragen mee geven:
- Wat vind je het leukst aan het programma?
- Wat was het lastigst?
- Als je meer tijd zou hebben, wat zou je dan toevoegen of veranderen?

Mathematical tables - direct instruction

Tafels- docentenhandleiding

In deze les gaan we verschillende figuren tekenen met Scratch.

Voorbereiding	- De leerdoelen klaarzetten - Scratch-omgeving klaarzetten op het digibord https://scratch.mit.edu/projects/294321441/editor/ - Scratch-omgeving voor de leerlingen https://scratch.mit.edu/projects/294343117/editor/ - Zelfstandige oefeningen klaarzetten / uitprinten
Lesduur	60 minuten
Niveau	Groep 6, 7, 8. De leerlingen hebben de Scratch-omgeving al een keer gezien
Werkvorm	Introductie klassikaal en daarna werken in tweetallen
Materiaal	Digibord, wisbordjes en laptops
Lesdoel	Verschillende rekentafels in Scratch programmeren
Leerdoelen	• Leerlingen kunnen blokjes herhalen met een herhaal-blok • Leerlingen kunnen een variabele gebruiken om een getal in op te slaan • Leerlingen kunnen de gebruiker een vraag stellen en het gegeven antwoord verwerken in hun programma

Handig om te weten

De les is gebaseerd op het materiaal van de MOOC Programmeren voor leerkrachten met Scratch. Dit is een gratis cursus die online te volgen is via https://www.edx.org/course/programmeren-voor-leerkrachten-met-scratch.

Voorbeeld van een eindprogramma: https://scratch.mit.edu/projects/294320676/editor/.
De stap-voor-stap uitwerking om deze les te maken is te vinden in de bijlage.

Zet voor het begin van de les alvast de laptops klaar. Je kunt bijvoorbeeld twee leerlingen vijf minuten voor de les vragen de laptops vast neer en aan te zetten. Zo kunnen de leerlingen sneller van start.

Klassikaal

Aan het begin van de les vraagt de leerkracht aan de leerlingen wat ze vorige programmeerles hebben gedaan. De leerkracht vraagt hierbij naar de concepten herhalen en variabele. Vervolgens bespreekt de leerkracht het lesdoel en de leerdoelen.

Nu de leerkracht de les heeft geïntroduceerd wordt het begin van het programma klassikaal gemaakt op het Digibord. De leerkracht opent de link https://scratch.mit.edu/projects/294321441/editor/.

De leerkracht vraagt aan de klas hoe we Karel 1 seconde 5 kunnen laten zeggen zodra er op de groene vlag is geklikt. De leerlingen mogen dit in tweetallen op hun wisbordje opschrijven.

De leerkracht programmeert bovenstaande op het digibord en denkt hardop terwijl hij/zij dit uitbreidt met meer 

blokjes zodat Karel uiteindelijk de antwoorden van de tafel van 5 opzegt. De leerkracht vraagt zich hardop af of dit nu wel zo'n handige manier is om te programmeren omdat als we een andere tafel willen hebben we best veel moeite moeten doen. De leerkracht vraagt de leerlingen op hun wisbordje op te schrijven wat voor soort blokje we kunnen gebruiken zodat we met minder moeite de tafel kunnen programmeren.

De leerkracht maakt een nieuwe variabele 'getal dat ik zeg' door hardop te denken. Hierbij benadrukt hij/zij dat het kiezen van een duidelijke naam handig is zodat we weten wat we in de variabele opslaan. De leerkracht vervangt de getallen in de zeg-blokken door de variabele 

en het blok 

te gebruiken.

Tussendoor vraagt de leerkracht aan de leerlingen wat er gebeurt als hij/zij op de groene vlag klikt. Wellicht is het concept ‘herhalen’ door de leerlingen al genoemd. Zo niet vraagt de leerkracht welk concept er nog meer gebruikt kan worden. De leerkracht gebruikt een 

en vraagt daarbij aan de leerlingen waar deze moet komen te staan en welke blokjes er precies in het herhaal-blok moeten zitten.

De leerkracht vraagt aan de leerlingen hoe hij/zij de tafel van 5 kan veranderen in de tafel van 4, past het programma aan en test het programma door op de groene vlag te klikken.

Voordat de leerlingen zelfstandig aan het werk gaan, gaan de leerlingen eerst een aantal opgaven maken die de leerkracht op het digibord kan zetten en die de leerlingen individueel hun wisbordje maken, zie bijlage. De leerkracht controleert vervolgens de antwoorden. Wanneer de meeste leerlingen het juiste antwoord hebben gegeven geeft de leerkracht geen uitleg meer. Wanneer een groter deel van de klas de vraag niet juist heeft begrepen geeft de leerkracht extra uitleg en geeft hij/zij eventueel nog een extra opgave.

Zelfstandig werken

De leerlingen werken in tweetallen aan hun programma. Ze kunnen dit programma openen via de https://scratch.mit.edu/projects/294343117/editor. Vervolgens kunnen ze aan de slag met de zelfstandigwerk-opgave die op het digibord weer wordt gegeven of wordt uitgedeeld. Tijdens het zelfstandig werken loopt de leerkracht langs de verschillende tweetallen en vraagt de leerlingen om hun programma en de gebruikte blokjes uit te leggen. Wanneer iets onduidelijk is kan de leerkracht extra uitleg geven aan de individuele leerling of klassikaal.

Wanneer leerlingen vragen hebben is het goed om eerst te controleren of ze de opgave gelezen en begrepen hebben voordat de leerkracht ze met inhoudelijke vragen helpt. Wanneer leerlingen vragen hebben over elementen van Scratch waar de les niet over gaat kan de leerkracht besluiten de vraag niet te beantwoorden maar de leerlingen terug te sturen naar de opgave.

Afsluiting

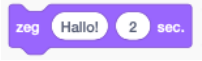
In de laatste paar minuten van de les vraagt de leerkracht aan de leerlingen om de les samen te vatten. De leerkracht vraagt hierbij ook wat de leerlingen vandaag geleerd hebben. Daarnaast herhaalt de leerkracht het lesdoel en de leerdoelen.

Bijlage

Overzicht van blokken per categorie

Uiterlijken

Zeg [tekst] [seconden] sec.



De sprite zegt de [tekst] voor [seconden] seconden.
Zeggen in Scratch betekent dat er een tekstballon verschijnt. Er komt dus *geen* geluid.

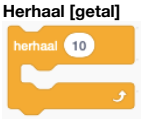
Gebeurtenissen

Wanneer op de groene vlag wordt geklikt

Dit blokje kun je bovenaan een reeks blokjes gebruiken. Alle blokjes die aan dit blok zijn vastgeplakt worden uitgevoerd wanneer er op de groene vlag wordt geklikt.

Besturen

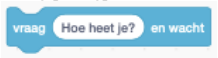
Herhaal [getal]



Alle blokjes binnen het herhaal-blok worden [getal] keer uitgevoerd.

Waarnemen

Vraag [vraag] en wacht



Met dit blokje kun je een vraag stellen aan de gebruiker van het programma. De sprite stelt de vraag en een invoerbalkje verschijnt in beeld. Het programma gaat pas verder wanneer de gebruiker op enter heeft gedrukt.

Antwoord

Antwoord is een speciale variabele. Hetgeen wat de gebruiker heeft geantwoord op de vraag wordt in deze variabele opgeslagen.

Variabelen

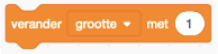
[variabele naam]

Dit blokje is een variabele waar een getal of stukje tekst in kan zijn opgeslagen.

Maak [variabele] [waarde]

Wijs [waarde] toe aan [variabele].
Met dit blokje kun je de variabele ook hernoemen en verwijderen. Klik hiervoor op [variabele].

Verander [variabele] met [getal]



Verander [variabele] met [getal]. [getal] kan zowel een positief als een negatief getal zijn. Wanneer [variabele] een stukje tekst is wordt [variabele] overschreven met waarde 1.

Opgave

Wanneer er op de groene vlag wordt geklikt zegt Karel
A. 0
B. 2
C. 5
D. Niks

Wanneer er op de groene vlag wordt geklikt zegt Karel
A. 0 2 4 2 4 0 2
B. 0 4 0
C. 0
D. Niks

Wanneer er op de groene vlag wordt geklikt zegt Karel
A. 0 3 6 9 12
B. 3 6 9 12 15
C. 0 3 6 9 12 15
D. Niks

Wanneer er op de groene vlag wordt geklikt zegt Karel
A. 0 5 10 15 20
B. 0 2 4 6 8
C. 0 0 0 0 0
D. Niks

Wanneer er op de groene vlag wordt geklikt zegt Karel
A. 3 3 3 3
B. 3 6 9 12
C. 3 9
D. Niks

Opgave - Extra

Wanneer er op de groene vlag wordt geklikt zegt Karel
A. 0 4 8
B. 0 4 4
C. 0 4
D. Niks

Wanneer er op de groene vlag wordt geklikt zegt Karel
A. 3 3 3 3 3
B. 3 6 9 12 15
C. 0 3 6 9 12
D. Niks

Wanneer er op de groene vlag wordt geklikt zegt Karel
A. 0 3 6 9 12
B. 3 6 9 12 15
C. 0 3 6 9 12 15
D. Niks

Wanneer er op de groene vlag wordt geklikt zegt Karel
A. 10
B. 12
C. 2 4 6 8 10
D. Niks

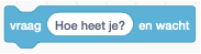
Wanneer er op de groene vlag wordt geklikt zegt Karel
A. 0 3 6 9
B. 3 6 9 12
C. 6 9 12 15
D. Niks

Wanneer er op de groene vlag wordt geklikt zegt Karel
A. 10 20 30 40 50 100
B. 10 20 30 40 50
C. 10 20 30 40 50 60
D. Niks

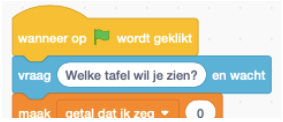
Zelfstandig werken

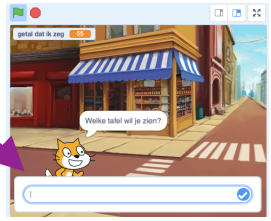
Open https://scratch.mit.edu/projects/294343117/editor.

1. Verander de tafel van 5 in de tafel van 6.
2. Het programma gaat nu van 1x tot 10x. Pas het programma zo aan dat de tafel loopt van 0x tot 10x.
3. Laat Karel nadat hij de tafel heeft opgenoemd zeggen: 'En nu gaan we achteruit terug!'.
4. Programmeer het programma zodat Karel de tafel achteruit zegt nadat hij zegt 'En nu gaan we achteruit terug!'. Vergeet niet om het programma te testen.

   Je moet nu steeds je programma aanpassen als je een andere tafel wilt. Laten we dit eens veranderen.
5. Ga naar de categorie 'Waarnemen' en sleep het blokje 

bovenaan je programma.
6. Verander de vraag in: Welke tafel wil je zien? De bovenkant van je programma ziet er dus zo uit:





   Druk op de groene vlag. Er verschijnt nu een tekstbalk in beeld. Hier kun je een getal invullen. Zodra je op enter drukt slaat het programma jouw antwoord op in de variabele Deze variabele vind je in de categorie 'Waarnemen'
7. Verander de variabele 

met 

   De bovenkant van je programma ziet er dus zo uit:

8. Bespreek nu samen wat er precies gebeurt als je op de groene vlag klikt. Probeer je programma een aantal keer uit.
9. Het terug tellen werkt niet helemaal goed meer. Kun jij ervoor zorgen dat je programma weer kloppend is?
   Tip: Gebruik hiervoor het blokje  in de categorie 'Functies'.

Voorbeeld Tafels

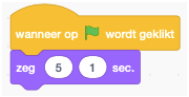
We gaan vandaag de tafels programmeren in Scratch. Deze les is gebaseerd op de MOOC Programmeren voor leerkrachten met Scratch, week 6. In deze handleiding staat stap voor stap uitgelegd hoe je dit programma maakt: https://scratch.mit.edu/projects/294320676/editor/.

Tafel van 5

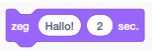
Video: https://www.youtube.com/watch?v=vBRwYvqYCe0

Ga naar https://scratch.mit.edu/projects/294321441/.

We gaan Karel de tafel van 5 laten zeggen. We beginnen door Karel 1 seconde 5 te laten zeggen zodra er op de groene vlag is geklikt. Dat ziet er zo uit:



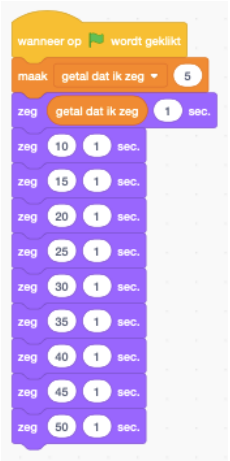
Voeg nu blokjes toe zodat Karel de tafel opnoemt van 5 tot en met 50. Als je er nu de tafel van 7 van wilt maken, kost dat best veel moeite. Daarom gaan we een variabele aanmaken. Die noemen we 'getal dat ik ga zeggen'.

Maak de variabele 'getal dat ik ga zeggen' gelijk aan 5 en vervang de '5' in het 

blokje door de variabele.

Het programma ziet er nu dus zo uit:

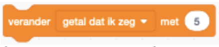
Druk op de groene vlag om te controleren of Karel nog steeds de tafel van 5 zegt.



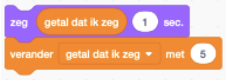
Maar nu gebruiken we de variabele alleen in het eerste zeg-blokje.

Vervang alle getallen met de variabele 

en druk op de groene vlag. Karel zegt nu steeds 5. De variabele moet dus nog veranderen.

Gebruik het 

blokje in je programma zodat Karel weer de tafel van 5 opnoemt. Je programma ziet er dan zo uit zoals je hier rechts kunt zien.

Misschien was je het al opgevallen, maar we doen steeds hetzelfde. De blokjes 

worden steeds herhaald.

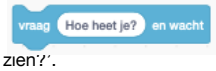
Ga naar 'Besturen' en sleep een 

blok naar je programma.

Zet het herhaal-blok om het eerste zeg- en verander-blok en verwijder de andere blokjes. Je programma ziet er nu dus zo uit:

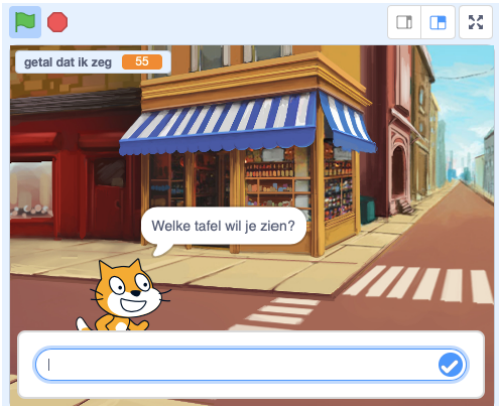
Om de tafel aan te passen naar bijvoorbeeld de tafel van 7 hoeven we nu dus maar twee getallen aan te passen. Maar is het niet leuk als de gebruiker van het programma zelf een tafel kan kiezen?

Welke tafel wil je zien?

Ga naar ‘Waarnemen’ en sleep het blokje 

bovenaan je programma.
Verander de vraag naar ‘Welke tafel wil je zien?’.

Druk op de groene vlag en je ziet dat er een balkje verschijnt waarin je iets kan typen. Vul in ‘7’.
Wat verwacht je dat er gebeurt?



Druk op enter. Karel noemt nog steeds de tafel van 5. Dat komt omdat we nog niks doen met wat er ingevuld wordt. Het getal dat we invullen wordt opgeslagen in een variabele, namelijk in 

Je vind deze in de categorie ‘Waarnemen’.

Vervang de 5’en in je programma met 

en klik op de groene vlag.

Je programma ziet er nu zo uit:

Mathematical tables - discover learning

Tafels- docentenhandleiding

In deze les gaan we verschillende figuren tekenen met Scratch.

Voorbereiding	Scratch-omgeving klaarzetten op het digibord https://scratch.mit.edu/projects/294320676/editor/ Scratch-omgeving voor de leerlingen https://scratch.mit.edu/projects/294321441/editor/
Lesduur	60 minuten
Niveau	Groep 6, 7, 8. De leerlingen hebben de Scratch-omgeving al een keer gezien
Werkvorm	Introductie klassikaal en daarna werken in tweetallen
Materiaal	Digibord en laptops
Lesdoel	Verschillende rekentafels in Scratch programmeren
Leerdoelen	• Leerlingen kunnen blokjes herhalen met een herhaal-blok • Leerlingen kunnen een variabele gebruiken om een getal in op te slaan • Leerlingen kunnen de gebruiker een vraag stellen en het gegeven antwoord verwerken in hun programma

Handig om te weten

De les is gebaseerd op het materiaal van de MOOC Programmeren voor leerkrachten met Scratch. Dit is een gratis cursus die online te volgen is via https://www.edx.org/course/programmeren-voor-leerkrachten-met-scratch.

Elke les is er een demonstratie beschikbaar. In de bijlage is de stap-voor-stap uitwerking te vinden van hoe dit programma gemaakt is.

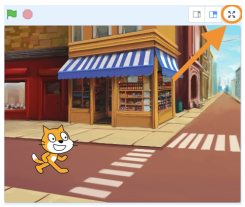
Zet voor het begin van de les alvast de laptops klaar. Je kunt bijvoorbeeld twee leerlingen vijf minuten voor de les vragen de laptops vast neer en aan te zetten. Zo kunnen de leerlingen sneller van start.

Introductie

Aan het begin van de les warmt de leerkracht de leerlingen op door een aantal vragen te stellen over tafels, zoals:
- Heb je een favoriete tafel?
- Welke tafel vind je moeilijk?
- Hoe zou je Scratch kunnen gebruiken om de tafels te oefenen?

De leerkracht koppelt het de opwarm oefening aan het lesdoel: rekentafels in Scratch programmeren.

Vervolgens zal de leerkracht een voorbeeldprogramma laten zien https://scratch.mit.edu/projects/294320676/editor/. De leerkracht kan het spel eerst vergroten door op de knop met de pijltjes naar buiten te klikken, zie onderstaande afbeelding, en vervolgens op de groene knop om het verhaal te demonstreren.



Aan de slag

De leerlingen gaan na de introductie zelfstandig in tweetallen aan de slag. De rol van de leerkracht is het coachen van de leerlingen. De leerlingen gaan dus zelf ontdekken wat ze kunnen doen in Scratch om zo hun verhaal te vormen.

De leerkracht zorgt ervoor dat de beginlink met het startprogramma https://scratch.mit.edu/projects/294321441/editor/ zichtbaar is op het digibord. De leerkracht kan de leerlingen als tip meegeven om Karel eerst één tafel te zeggen.

Wanneer een leerling vastzit kan de leerkracht de volgende vragen stellen:
- Wat wil je dat je programma doet?
- Welke blokken heb je hiervoor nodig?
- Kun je me uitleggen wat de blokken in je programmeerveld nu doen?
- Wat is er anders in de uitvoer dan je verwacht had?

De leerkracht zegt dus **niet** voor wat de oplossing kan zijn.

De leerkracht stimuleert de leerlingen om een herhaal-blok en variabele te gebruiken door vragen te stellen zoals:
- Je gebruikt nu heel veel dezelfde blokjes. Is er een blokje waardoor je dit korter kan programmeren?
- Wat gebeurt er in je programma?
- Zijn er blokjes die steeds herhaald worden?

Afsluiting

Tijdens de afsluiting mogen leerlingen bij elkaars projecten kijken. De leerlingen stellen daarbij vragen aan elkaar. De leerkracht kan één of meerdere van de volgende vragen mee geven:
- Wat vind je het leukst aan het programma?
- Wat was het lastigst?
- Als je meer tijd zou hebben, wat zou je dan toevoegen of veranderen?

C.2 Method

C.2.1 Code Tree

id	Vraag	Categorie	Labels	Keywords / uitleg
1	Hoe legt de leerkracht nieuwe stof uit?	uitleg methode	student centraal	Ervaren, zelf opzoeken, zelf uitzoeken, gesprek, begeleiden, uit leerlingen halen
1			mix	Zie student en leerkracht centraal
1			leerkracht centraal	Lesdoel, voorkennis ophalen, voordoen, controle begrip, uitleg geven, verlengde instructie
2	Wat is de programmeer ervaring van de leerkracht?	Ervaring	geen	
2			buiten onderwijzende rol	vroeger op middelbare school, opleiding informatica
2			op school, buiten klas	workshop, training, uitproberen met oog op gebruik in de klas
2			op school, geobserveerd	geen leidende rol, maar wel leerlingen gezien die aan de slag gingen met het materiaal
2			op school, geven bestaand materiaal	BomberBot
2			op school, eigen materiaal	zelf ontwikkelen
3	Welk gevoel heeft de leerkracht bij programmeren in het onderwijs?	Gevoel	Leuk	interessant, benieuwd
3			Spanend	
3			Onzeker	ingewikkeld (kan deze hierbij?)
3			Frustrerend	
3			Teleurstellend	
4	Hoe zou de leerkracht een Scratch les geven	Scratch methode	student centraal	Ervaren, zelf opzoeken, zelf uitzoeken, gesprek, begeleiden, uit leerlingen halen
4			mix	Zie student en leerkracht centraal
4			leerkracht centraal	Lesdoel, voorkennis ophalen, voordoen, controle begrip, uitleg geven, verlengde instructie
5	Verwacht de leerkracht genoeg kennis/vaardigheden te hebben om de programmeerlessen te geven?	Instructional strategies	Ja	Zeker
5			Ja maar	Oppervlakkig / basis wel maar verdieping of specifieke situaties niet
5			Weet het niet zeker / geen idee	
5			Nee	
6	Verwacht de leerkracht verschillen in het klassenmanagement?	Klassenmanagement	Geen verschil	
6			één aspect anders	
6			twee of meer aspecten anders	
7	Heeft de leerkracht een idee hoe hij/zij de leerlingen kan betrekken	Student engagement	Ja	
7			Soms	
7			Nee	inclusief twijfel / uitdaging
8	Overig	Overig	Online opzoeken	
8			Quotes	
8			Stapjes - programmeren	
8			Stapjes - directe instructie	
8			Uitdaging	

id	Vraag	Categorie	Labels	Keywords / uitleg
1	Hoe heeft de leerkracht het geven de programmeerlessen ervaren?	Ervaren	Moeilijk	Lastig, viel tegen
1			Makkelijk	simpel, eenvoudig, viel mee
2	Welk gevoel heeft de leerkracht bij het geven van programmeerlessen?	Gevoel	Leuk	interessant
2			Spanend	
2			Onzeker	klinkt stom
2			Frustrerend	
2			Teleurstellend	
3	Hoe zou de leerkracht nieuwe Scratch lesen geven?	Nieuwe lessen	student centraal	Ervaren, zelf opzoeken, zelf uitzoeken, gesprek, begeleiden, uit leerlingen halen
3			mix	Zie student en leerkracht centraal
3			leerkracht centraal	Lesdoel, voorkennis ophalen, voordoen, controle begrip, uitleg geven, verlengde instructie
3			Weet het niet zeker / geen idee	
4	Ervaart de leerkracht dat hij/zij genoeg kennis/vaardigheden heeft om de programmeerlessen te geven?	Instructional strategies	Ja	Zeker
4			Ja maar	Oppervlakkig / basis wel maar verdieping of specifieke situaties niet
4			Weet het niet zeker / geen idee	
4			Nee	
4			Niet van toepassing	Want het is niet nodig
5	Heeft de leerkracht verschillen in het klassenmanagement ervaren?	Klassenmanagement	Geen verschil	
5			één aspect anders	
5			twee of meer aspecten anders	
6	Heeft de leerkracht een idee hoe hij/zij de leerlingen kan betrekken	Student engagement	Ja (neutraal)	Geen expliciete toekenning waar dit aan ligt
6			Ja (zelf)	Expliciete toekenning aan zichzelf als leerkracht
6			Ja (extern)	Expliciete toekenning aan externe factoren zoals handleiding, de vrijheid, het werkblad, de lesen
6			Soms	
6			Nee	inclusief twijfel / uitdaging
7	Welke voordeel ziet de leerkracht voor directe instructie?	Voordelen di	Structuur en strategie	Instructie, hoe omgaan met niveauverschillen
7			Resultaten	Leerlingen vinden het lastig, succeservaringen
7			Benodigde bekwaamheid leerkracht	Basiskennis
8	Welke voordeel ziet de leerkracht voor ontdekkend leren?	Voordelen odl	Structuur en strategie	Instructie, hoe omgaan met niveauverschillen
8			Resultaten	Leerlingen vinden het lastig, succeservaringen
8			Benodigde bekwaamheid leerkracht	Basiskennis
9	Welke nadelen ziet de leerkracht voor directe instructie?	Nadelen di	Structuur en strategie	Instructie, hoe omgaan met niveauverschillen
9			Resultaten	Leerlingen vinden het lastig, succeservaringen
9			Benodigde bekwaamheid leerkracht	Basiskennis
10	Welke nadelen ziet de leerkracht voor ontdekkend leren?	Nadelen odl	Structuur en strategie	Instructie, hoe omgaan met niveauverschillen
10			Resultaten	Leerlingen vinden het lastig, succeservaringen
10			Benodigde bekwaamheid leerkracht	Basiskennis
11	Voor welke methode heeft de leerkracht een voorkeur bij programmeerlessen?	Voorkeur	Directe instructie	
11			Ontdekkend leren	
11			Neutraal	Ook leuk, geen voorkeur, weet het niet

12	Verwacht de leerkracht vaker programmeerles te gaan geven?	Vaker programmeerles	Ja	
12			Nee	
12			Weet het niet zeker / geen idee	
13	Hoe vind de leerkracht zijn/haar bekwaamheid?	Bekwaamheid	Hoog	
13			Midden	Basis okay, verdieping niet
13			Laag	
13			Niet van toepassing	Want het is niet nodig
14	Is de bekwaamheid van de leerkracht veranderd gedurende het onderzoek?	Verandering bekwaamheid	Ja gegroeid	
14			Ja gedaald	
14			Nee	
15	Heeft de leerkracht programmeren als buiten de eigen comfort zone ervaren?	Comfort zone	Ja	
15			Nee	
16	Is de leerkracht inovatief vaardig?	Innovatie skill	Ja	Flexibility, innovate, learning, challenges, creativity
16			Soms	
16			Nee	
17	Ziet de leerkracht kennis als iets statisch of dynamisch?	Kennis	Statisch	
17			Dynamisch	
18	Is de leerkracht bezig met reflecteren, evalueren en verbeteren van de eigen lessen?	Reflectie	Ja	
18			Soms	
18			Nee	
19	Overweegt de leerkracht om zelf materiaal te maken?	Nieuw materiaal maken	Ja	
19			Misschien	
19			Nee	
19			Weet het niet zeker / geen idee	
20	Heeft de leerkracht kennis/vaardigheden uit andere vakken kunnen toepassen tijdens de programmeerles?	Kennis/vaardigheden ander vak	Ja	
20			Nee	
21	Heeft de leerkracht gefocust op reproduceren van de lessen of op zelf ontdekken bij de voorbereiding in Scratch?	Voorbereiding focus	Reproduceren	
21			Mix	
21			Zelf ontdekken	
22	Overig	Overig	Online opzoeken	
22			Quotes	
22			Stapjes - programmeren	
22			Stapjes - directe instructie	
22			Uitdaging	
22			Adaptief figuur	Commentaar bij intekenen in het innovatie en efficientie diagram
22			Feedback via kinderen	
22			Scratch specifiek	
22			Ontdekken als dekmantal	Sommige leerkrachten gebruiken ontdekkend leren om zich te verschuilen van het niet weten

C.2.2 Interview protocol

Pre

Interview

Vooraf

- STEBI-NL Programmeren in Scratch in laten vullen
- Controleer of apparatuur werkt
- Bedanken
- Onderzoek: programmeren in PO met focus op leerkracht
  Benadruk: anoniem verwerkt, focus op praktijk, geen oordeel goed/fout
- Interview: lesvoorkeuren, ervaringen met programmeeronderwijs, lesmethodes en verwachtingen programmeeronderwijs
- Na het interview lessen bespreken

Vragen

Opwarm *2 minuten*
- Kunt u kort vertellen wat u het leukst vind aan lesgeven?

Voorkeur reguliere lessen *5 minuten*
- Wanneer u iets nieuws gaat uitleggen, hoe pakt u dit aan?
- Hoe test u het begrip van leerlingen in een reguliere les?
- Gebeurt het wel eens in een (reguliere) les dat u een vraag van een leerling niet kan beantwoorden? Zo ja wat doet u dan?

Ervaring *5 minuten*
- Hoe zou u programmeren beschrijven? / Wanneer is iets programmeren?
- Wat zijn uw huidige ervaringen met programmeeronderwijs?
  - Talen / hardware
  - Zelf geven van lessen (vorm en hoevaak)
  - Training / cursus
- Als u zelf een programmeerles zou moeten geven in Scratch, hoe zou u dit dan aanpakken?

Lesmethode *5 minuten*
- Hoe zou u directe instructie beschrijven? Gebruikt u dit in de klas? Zo ja, hoe?
- Hoe zou u ontdekkend leren beschrijven? Gebruikt u dit in de klas? Zo ja, hoe?

Verwachting *15 minuten*
- Wat verwacht u van de Scratch lessen?
  - Wat voor soort onderwerpen verwacht u?
  - Wat worden voor u de uitdagingen?
- Hoe zou u de kennis / het begrip van leerlingen kunnen testen in een programmeerles
- Verwacht u dat u alternatieve voorbeelden kunt geven wanneer een leerling een vraag niet snapt?
- Denkt u dat u met behulp van vragen stellen de leerling verder kan helpen?
- Verwacht u dat u alle vragen van de leerlingen kunt beantwoorden? Wat doet u als u de vraag van de leerling niet kunt beantwoorden?
- Verwacht u dat het klassenmanagement anders gaat zijn in een programmeerles dan in een reguliere les?
- Het kan gebeuren dat een leerling gefrustreerd raakt. Is er iets wat u kunt doen om leerlingen zelfvertrouwen te geven tijdens de lessen?
- Verwacht u dat u leerlingen die de programmeerlessen minder leuk vinden toch kunt motiveren?

Afsluiting *3 minuten*
- Hoe vind u het om de programmeerlessen te gaan geven?
  - Nerveus
  - Zekerheid
  - Leuk

Post

Interview

Vooraf

- STEBI-NL Programmeren in Scratch in laten vullen
- Controleer of apparatuur werkt
- Bedanken
- Interview: bespreken lessen en daarbij inzoomen op zelfbekwaamheid en adaptieve expertise

Vragen

Maar eerst *2 minuten* *(0 / 2)*
- Hoe zou u programmeren beschrijven?

Evaluatie lessen *5 minuten* *(2 / 7)*
- Hoe vond u het om zelf Scratch lessen te geven?
- Bent u tevreden met hoe de lessen zijn gegaan?
- Wat vond u goed gaan?
- Wat vond u minder goed gaan?
  - Waren er dingen in de lessen die u lastig vond als leerkracht?
- Zaten er in het geven van de lessen verschillen in vergelijking met andere (reguliere) lessen?
- Zaten er elementen in de lessen die u verwachte of een verassing waren?
- Wat vond u van de onderwerpen?
- Wat vond u van de opbouw van de lessen?
- Was er genoeg tijd?

Instructional strategies *8 minuten* *(7 / 15)*
- Hoe was het om de programmeerconcepten uit te leggen?
- Gebeurde het dat een leerling een concept niet in één keer snapte?
  - In hoeverre lukte het dan om de een alternatieve uitleg en/of voorbeeld te geven?
- In hoeverre heeft u de kennis / het begrip van de leerling kunnen toetsen?
  - Vielen er nog dingen op bij het toetsen van het begrip?
  - Zijn er nog andere manier waarop u het begrip in kaart zou kunnen brengen tijdens een programmeerles?
  - Heeft u een idee welke leerlingen de stof goed begrepen en welke minder?
  - In hoeverre heeft u het idee dat u heeft kunnen bijdragen aan het verbeteren van het begrip (van de leerlingen)?
- Lukte het om regelmatig *verschillende* vragen te stellen aan de leerlingen?
  - Kunt u vertellen wat voor type vragen u gesteld heeft?
  - Kunt u een voorbeeld geven van wat voor vragen u heeft gesteld?
  - In hoeverre heeft u met het zelf stellen van vragen leerlingen verder kunnen helpen als ze vast zaten?
- Stelde de leerlingen veel vragen? Lukte het om deze vragen te beantwoorden?
- Was er veel niveauverschil tussen de leerlingen?
  - Zo ja, hoe ging u hier mee om?
  - In hoeverre lukte het om de lessen aan te passen naar het niveau van de leerlingen?
  - In hoeverre kon u de leerlingen die het goed snapte uitdagen?

Classroom management *2 minuten* *(15 / 17)*
- Was het klassenmanagement vergelijkbaar met een reguliere les?
  - Waarom? / Hoe komt dat?

Student engagement *5 minuten* *(17 / 22)*
- Hoe vonden de leerlingen de lessen?
- Hoe pakte de leerlingen de lesstof op?
  - Waar denkt u dat dit aan lag?
  - In hoeverre heeft u hier als leerkracht invloed op?
- Waren er leerlingen die de les minder leuk vonden?
  - Zo ja, lukte het u om deze leerlingen toch te motiveren?
- Zijn er leerlingen gefrustreerd geraakt? Hoe ging u hiermee om?
- In hoeverre heeft u de leerlingen kunnen stimuleren om kritisch te nadenken?
  - Waarom? Hoe?
- In hoeverre heeft u de creativiteit van uw leerlingen kunnen aanmoedigen?

Reflectie *10 minuten* *(22 / 32)*
- Als u *dezelfde* lessen aan een nieuwe klas zou geven, wat zou u dan hetzelfde doen en wat anders?
- Als u zelf *nieuwe* programmeerles zou moeten geven (in Scratch), hoe zou u dit dan aanpakken?
  - Waar zou u het materiaal vandaan halen?
  - Zou u overwegen zelf materiaal te maken?
- Had u Scratch lessen gaan geven als u niet mee had gedaan aan dit onderzoek?
  - Waarom?
- Verwacht u vaker lessen te gaan geven in Scratch?
  - Waarom?
  - Wat zou u helpen om meer lessen te geven? / Wat heeft u hier voor nodig?
- U heeft de lessen gegeven volgens directe instructie / ontdekkend leren gebruikt. Welke voor en nadelen zaten volgens u aan de methode tijdens de programmeerles?
- Hoe zou u het vinden als u de lessen moest geven met ontdekkend leren / directe instructie? *(Voorbeeld andere les opbouw mee nemen)*

Met de verschillende vragen die ik heb gesteld probeer ik te pijlen hoe bekwaam u zichzelf voelt in het geven van programmeer onderwijs.
geloof in uw eigen capaciteiten om programmeeronderwijs te implementeren in de klas
- Zou u kunnen zeggen hoe bekwaam u zich voelt om programmeerles te geven?
- In hoeverre is uw zelfbekwaamheid veranderd gedurende het onderzoek?
- Waar komt dat door?
- Zou uw zelfbekwaamheid (nog) groter kunnen worden? Wat is hier voor nodig?

Adaptieve expertise 8 *minuten* *(32 / 40)*
- In hoeverre bent u tijdens de programmeerlessen buiten uw comfort zone gestapt?
  - Stapt u vaker buiten uw comfort zone? Op welke manier?
- Heeft u zelf dingen uitgeprobeerd in Scratch?
  - Lag de focus dan vooral op het reproduceren van mijn lessen of ging u zelf dingen uitproberen?
- In hoeverre kon u kennis en vaardigheden uit andere vakken toepassen in de programmeerlessen?
- Probeert u vaak nieuwe dingen uit in de klas? Wanneer?
  - Hoe vind je het als iets niet gaat zoals gepland?
  - Hoe staat u tegenover het maken van fouten als leerkracht?
  - Als iets even niet lukt, bijvoorbeeld bij programmeren in Scratch, wat doet u dan?
- Komt u vaak nieuwe situaties tegen tijdens het lesgeven?
- Vind u het leuk om nieuwe dingen te leren, ook als ze lastig zijn/lijken?
- In hoeverre vind u het belangrijk om als leerkracht bij te blijven leren?
- Ziet u zichzelf als innovatief als het om onderwijs gaat?
  - Waarom?
- Als u naar dit figuur kijkt
  - Waar zou u uzelf op dit moment plaatsen als het om programmeeronderwijs gaat?
  - Waar stond u voor de lessen reeks?
  - Hoe is de route van punt 1 naar punt 2 gelopen?
  - Waar staat u als leerkracht in het algemeen?
  - Is dit nog veranderd door de programmeerlessen?
- Wat verstaat u onder het begrip kennis?
  - Statisch of dynamisch?
- Krijgt u feedback op uw lessen van andere?
  - Wat doet u met feedback?
- Reflecteert u zelf op uw eigen lessen?
  - Doet u dit alleen of met collega's?
- Hoeveel ruimte krijgt u op school om uzelf verder te ontwikkelen?
  - Op welk vlak?

Toekomst *5 minuten* *(40 / 45)*
- Wat is er volgens u nodig om programmeren in het basisonderwijs een plek te geven?
  - Welke rol speelt het schoolbestuur hierbij?
  - Welke rol speelt scholing van de leerkracht hierbij?
  - Wat kunnen valkuilen zijn voor de leerkrachten?

C.2.3 Questionnaire

Naam _____

Leeftijd _____

Geslacht man / vrouw / anders

De onderstaande stellingen gaan over programmeren in Scratch. Geef aan in hoeverre u het eens of oneens bent met deze stellingen door één rondje per stelling aan te kruisen. Mocht u nog geen programmeerles geven in Scratch, vul dan in wat uw verwachtingen zijn.

	Volledig oneens	Oneens	Noch eens / Noch oneens	Eens	Helemaal eens
Als een leerling beter presteert dan anderen tijdens een programmeerles in Scratch, dan kan dat komen doordat de leerkracht zich extra heeft ingespannen voor deze leerling	O	O	O	O	O
Wanneer de programmeerresultaten in Scratch van leerlingen verbeteren, komt dit vaak door een verandering in de manier van lesgeven door de leerkracht	O	O	O	O	O
Als leerlingen goed presteren op het gebied van programmeren in Scratch ligt dat zeer waarschijnlijk aan het aangeboden programmeerprogramma op de school	O	O	O	O	O
Onvoldoende achtergrondkennis van leerlingen op het gebied van programmeren in Scratch kan overwonnen worden door programmeeronderwijs op een andere manier aan te bieden	O	O	O	O	O
Goede programmeerprestaties in Scratch van leerlingen zijn over het algemeen te danken aan de kwaliteit van de leerkracht	O	O	O	O	O
Wanneer een slecht presterend kind vooruitgang boekt bij programmeren in Scratch, is het vaak het resultaat van extra aandacht door de leerkracht	O	O	O	O	O
Extra inspanningen voor programmeeronderwijs in Scratch door de leerkracht leiden tot verbeteringen in de leerlingresultaten	O	O	O	O	O
De leerkracht is over het algemeen verantwoordelijk voor de resultaten van de kinderen voor programmeren in Scratch	O	O	O	O	O
De prestaties van leerlingen voor programmeren in Scratch zijn direct afhankelijk van de invulling van de programmeerlessen van hun leerkracht	O	O	O	O	O
Als ouders aangeven dat hun kind meer interesse toont voor programmeren in Scratch, komt dat waarschijnlijk door de kwaliteiten van de leerkracht op dit gebied	O	O	O	O	O

Let op, de vraag gaat door op de achterkant

	Volledig oneens	Oneens	Noch eens / Noch oneens	Eens	Helemaal eens
Goed programmeeronderwijs in Scratch heeft weinig effect op de prestaties van leerlingen met een slechte motivatie	O	O	O	O	O
Leraren met veel kennis en vaardigheden op het gebied van programeeronderwijs in Scratch kunnen de meeste leerlingen helpen om programmeeronderwerpen te begrijpen	O	O	O	O	O
Programeeronderwijs in Scratch geef ik net zo goed als andere vakken	O	O	O	O	O
Ik weet hoe ik leerlingen concepten uit het programeerdomein in Scratch moet aanleren	O	O	O	O	O
Ik kan leerlingen zodanig begeleiden bij programmeeropdrachten in Scratch, dat zij zelf antwoorden kunnen vinden op hun eigen vragen	O	O	O	O	O
Over het algemeen ben ik tevreden over de manier waarop ik programmeeronderwijs in Scratch geef	O	O	O	O	O
Ik begrijp zelf de programmeerconcepten in Scratch goed genoeg om de kinderen deze concepten op een effectieve manier te leren	O	O	O	O	O
Ik kan leerlingen uitleggen wat de onderliggende theorie is bij een programmeeropdracht in Scratch	O	O	O	O	O
Ik ben over het algemeen in staat om vragen over programmeren in Scratch van kinderen te beantwoorden	O	O	O	O	O
Ik heb de benodigde vakdidactische vaardigheden om les te geven in programmeren in Scratch	O	O	O	O	O
Als mijn directeur of een collega bij een les aanwezig is, dan vind ik het prima als dat een programmeerles in Scratch is	O	O	O	O	O
Als een leerling moeite heeft met een programmeerconcept in Scratch, dan weet ik hoe ik de leerling moet helpen om het beter te begrijpen	O	O	O	O	O
Als ik programmeren in Scratch geef vind ik het fijn als leerlingen vragen stellen	O	O	O	O	O
Ik weet wat ik moet doen om leerlingen voor programmeren in Scratch te motiveren	O	O	O	O	O

Geef aan in hoeverre u het eens of oneens bent met deze stellingen door één rondje per stelling aan te kruisen. Let op, deze stellingen gaan niet specifiek over programmeeronderwijs.

	Volledig oneens	Oneens	Noch eens / Noch oneens	Eens	Helemaal eens
Tijdens mijn loopbaan als leerkracht, …					
ben ik in staat geweest om, vanuit wat ik in het verleden geleerd heb, nieuwe kennis te ontwikkelen en te integreren	O	O	O	O	O
heb ik mij bezig gehouden met de nieuwste ontwikkelingen in het onderwijsveld	O	O	O	O	O
heb ik een beter begrip gekregen van concepten in mijn vakgebied	O	O	O	O	O
heb ik mij gerealiseerd dat kennis in mijn vakgebied zich blijft ontwikkelen	O	O	O	O	O
heb ik mij gerealiseerd dat ik voortdurend moet leren om een expert in mijn vakgebied te worden en te blijven	O	O	O	O	O
heb ik laten zien dat ik bereid ben om nieuwe aspecten, gerelateerd aan het onderwijs, te leren	O	O	O	O	O
heb ik mijn kennis toegepast in nieuwe en onbekende onderwijssituaties met een zekere mate van succes	O	O	O	O	O
heb ik mij gericht op nieuwe uitdagingen	O	O	O	O	O
heb ik nieuwe vakken/onderwerpen hetzelfde aangepakt als reeds bekende vakken/onderwerpen	O	O	O	O	O
ben ik in staat geweest om op hoog niveau te blijven presteren wanneer ik geconfronteerd werd met onbekende situaties of taken	O	O	O	O	O
ben ik in staat geweest om mijn kennis flexibel toe te passen binnen verschillende vakken/onderwerpen	O	O	O	O	O