To Scratch or not to Scratch?

A controlled experiment comparing plugged first and unplugged first programming lessons

Felienne Hermans, Efthimia Aivaloglou
Delft University of Technology
f.f.j.hermans@tudelft.nl,E.Aivaloglou@tudelft.nl

ABSTRACT

Programming education is in fashion: there are many methods, tools, books and apps to teach children programming. This gives rise to the question of how to teach programming. Do we teach the concepts with or without the use of a computer, also called *plugged* and *unplugged* respectively? This paper aims to measure what method is more effective to start with: plugged or unplugged first. Specifically, we are interested in examining which method is better in terms of (1) facilitating understanding of programming concepts, (2) motivating and supporting the students' sense of self-efficacy in programming tasks, and (3) motivating the students to explore and use programming constructs in their assignments. To this end we conduct a controlled study with 35 elementary school children, in which half of the children receive four plugged lessons and the other half receives four unplugged lessons. After this, both groups receive four weeks of Scratch lessons. The results show that after eight weeks there was no difference between the two groups in their mastering of programming concepts. However, the group that started with unplugged lessons was more confident of their ability to understand the concepts, i.e. demonstrated better self-efficacy beliefs. Furthermore, the children in the unplugged first group used a wider selection of Scratch blocks.

CCS CONCEPTS

•Social and professional topics → Computing education; Computational thinking; K-12 education;

KEYWORDS

programming education, Scratch, unplugged

ACM Reference format:
Felienne Hermans, Efthimia Aivaloglou. 2017. To Scratch or not to Scratch?. In *Proceedings of WiPSCE '17, Nijmegen, Netherlands, November 8–10, 2017,* 8 pages.
DOI: 10.1145/3137065.3137072

WiPSCE '17, Nijmegen, Netherlands

978-1-4503-5428-8/17/11…$15.00
DOI: 10.1145/3137065.3137072

1 INTRODUCTION

Over the last decade interest in programming education has been growing. An increasing number of countries is including programming and computational thinking (CT) in the curricula of elementary schools. In the UK, children as young as 5 are already introduced to programming concepts [3].

The introduction of programming education naturally gives rise to the question how to best teach programming and CT to children. One of the topics of these discussions is the role of the computer: should we teach with or without the computer, or *plugged* versus *unplugged*.

We do believe that ultimately children need to be able to apply programming concepts using a computer, so we are not interested in comparing unplugged entirely to plugged entirely. Hence, in this paper we focus on the question whether it is better to *start* with plugged lessons immediately, or is it better to first use unplugged materials.

Specifically, we are interested in examining which method is better in terms of (1) facilitating understanding of programming concepts, (2) motivating and supporting the students' sense of self-efficacy in programming tasks, and (3) motivating the students to explore and use programming constructs in their assignments.

Our motivation for investigating the effects the teaching methods to the student's self-efficacy beliefs is that those have been found to affect certain career entry behaviors, such as college major choices and academic performance [12, 19]. Self-efficacy was originally presented by A. Bandura as the belief that one can successfully execute behaviors required to produce a desired outcome [4]. In education research, self-efficacy has become one of the most important motivational variables that helps explain the relationship between past performance and future results [13]. This relationship had been found to be strong also in middle-school students [5] and for various subject areas, including mathematics [16] and programming [13, 18, 23].

To address our three research questions, we run a two phase experiment that compares starting with unplugged lessons to starting on the computer. We teach 35 elementary school children aged 8 to 12, separated in two random groups, for eight weeks. In the first phase, consisting of four weeks, we teach half of the children (17) with Scratch, while the other half (18) used unplugged materials only. Both the plugged and the unplugged lessons covered these concepts: loops, conditionals, procedures, broadcasts, parallelization and variables. After these four weeks, both groups receive two weeks of Scratch lessons, to practice Scratch programming in more depth. In these lessons, we repeat the concepts taught in phase one. For the unplugged group, we design one special lesson that connects the concepts as they used them unplugged to concepts in Scratch. After these two weeks, two more weeks follow

in which children create their own games in Scratch. We close the experiment with an endterm test in which we assess children's understanding and correct use of programming concepts in Scratch.

The results show that after eight weeks (1) there was no difference between the two groups in their mastering of programming concepts, (2) the unplugged first group demonstrated more self-efficacy, and (3) that they use a wider vocabulary of Scratch blocks, including more blocks that were not explained in the course materials.

2 EXPERIMENTAL SETUP

2.1 Research Questions

The goal of this study is to understand whether there is a difference between starting with Scratch immediately (Plugged first) or practicing programming concepts without a computer before (Unplugged first). We therefore answer the following three research questions:

RQ1 Do the children that start with unplugged materials understand programming concepts better?

RQ2 Do the children that start with unplugged materials have more self-efficacy about programming?

RQ3 Do the children that start with unplugged materials use a smaller vocabulary of Scratch blocks?

Associated with these research questions are three null hypotheses, which we formulate as follows:

$H1_0$ Learning about programming concepts unplugged first does not impact children's understanding of programming concepts.

$H2_0$ Learning about programming concepts unplugged first does not impact children's self-efficacy about programming.

$H3_0$ Learning about programming concepts unplugged first does not impact children's use of different Scratch blocks.

The alternative hypotheses that we use in the experiment are the following:

$H1_1$ Learning about programming concepts unplugged first increases children's understanding of programming concepts.

$H2_1$ Learning about programming concepts unplugged first increases children's self-efficacy about programming.

$H3_1$ Learning about programming concepts unplugged first impacts children's use of different Scratch blocks.

To test the null hypotheses, we perform a controlled experiment with 35 elementary school children, randomly assigned to one of two groups: starting with Scratch programming immediately or starting with four weeks of unplugged material.

2.2 Subjects

The subjects in our experiment are children from three grades of one school, varying in age between 8 and 12, with an average of 10.0.

Figure 1 shows an overview of the ages of the 35 children in the experiment. We randomly divided the 35 children into two groups, however we balanced the genders and the grades as much as possible given the constraints we were given. [1] Figure 2 shows the division of the children and their grades over the two groups, and Figure 3 shows their gender.

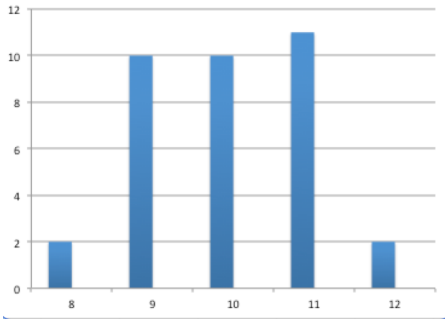

Figure 1: Ages of the children

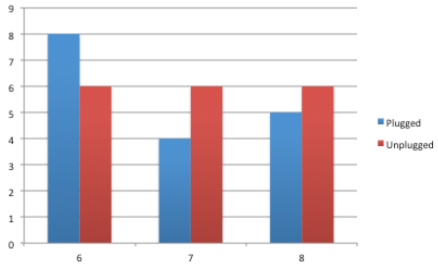

Figure 2: Grades of the children in the two groups (8 being the final grade of elementary school)

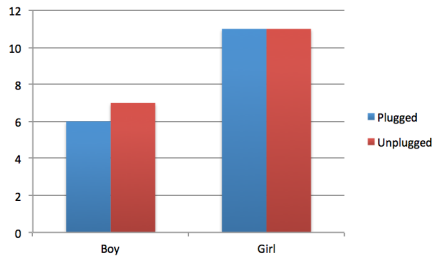

Figure 3: Genders of the children in the two groups

2.3 Programming Concepts

In the materials for both the plugged first and the unplugged first group, we teach six basic programming concepts.

- Loops
- Conditionals
- Procedures

[1] among the children were two pairs of siblings who could not be in the same group since they would work on the same laptop

Table 1: Overview of the materials of the eight week programming course

Week	Unplugged group	Plugged group	Concepts
0	Getting to know Scratch, create user accounts	Getting to know Scratch, create user accounts	—
1	CS Unplugged Marching Orders	Scratch MOOC lesson 1	Loops and Conditionals
2	Self-Efficacy Questionnaire	Self-Efficacy Questionnaire	—
2	CS Unplugged Network Protocols	Scratch MOOC lesson 2	Broadcasts and Procedures
3	Computing Without Computers Box Variables	Scratch MOOC lesson 3	Variables and Parallelization
4	Finish materials	Finish materials	All
5	Intro Scratch	More Scratch practice	Loops, Conditions and Procedures
6	Scratch Practice	Scratch Practice	Broadcasts, Variables and Parallelization
7	Final Project	Final Project	—
8	Final Project	Final Project	—
8	End Term Scratch Test	End Term Scratch Test	—
8	Self-Efficacy Questionnaire	Self-Efficacy Questionnaire	—

- Broadcasts
- Parallelization
- Variables

These concepts are divided over three chapters that teach two concepts each, both for the plugged first and for the unplugged first group.

2.4 Lesson plan

Table 1 shows an overview of our lesson plan. We start with one introductory computer lesson for all children, in which we set up wifi on their computers and create a Scratch account for all children. In addition to that, we show them a demo of how Scratch works, ensuring that also the children that will do the unplugged lessons first have context of what programming is. After that initial lesson, we divide the children into two random groups which receive different treatment. Half of the children use Scratch to learn about the six programming concepts, while the other group learns programming 'unplugged'. The following describes the eight weeks of lessons in more detail.

2.4.1 Week 0. In week 0, children set up a Scratch account, and we also show them the basic idea of Scratch: that you can control a sprite on the screen with blocks.

2.4.2 Weeks 1 to 4. After week 1, the groups are split and work on either plugged or unplugged materials. Initially we envisioned that the children would finish the material we provided in three weeks, but we gave them an additional week to finish all lessons, and also to provide an opportunity to children that missed a lesson due to sickness to catch up.

Unplugged first For the four unplugged lessons, we use material based on CS Unplugged.[2] CS Unplugged is *"a collection of free learning activities that teach Computer Science through engaging games and puzzles that use cards, string, crayons and lots of running around"*. Previous research has shown that CS Unplugged is an effective way of teaching programming concepts [21]. All our adapted materials are available online.[3]

Plugged first For the four plugged lessons, we use materials developed for the first author's online Scratch course[4], which has been used previously by over 6000 children to date. These materials too are available.[5]

2.4.3 Weeks 5 and 6. From week 5 onwards, children in both groups use Scratch to create programs and to practice the learned concepts. The plugged first group receives materials in which the concepts that were previously taught are repeated, and practiced more.

For the unplugged first group, we design a special introductory lesson, in which we explain how the programming concepts they practiced with in the unplugged lessons manifest in Scratch. An example of the connection of unplugged to Scratch concepts is shown in Figure 4.

2.4.4 Week 7 and 8. In weeks 7 and 8, we give children the opportunity to create their own programs, in order to measure children's Scratch vocabulary, and see what blocks and concepts they use when creating programs.

2.4.5 Week 8. In week 8, we close the course with an endterm test, in which we measure understanding of programming concepts, in Scratch, but performed on paper. The endterm consisted of multiple choice questions, some testing one and some testing multiple concepts. The endterm too is available.[6] Figure 5 shows questions testing the concepts operators and variables.

2.5 Self-efficacy assessment

To measure the students' self-efficacy beliefs we used the self-efficacy subscale of the Motivated Strategies for Learning Questionnaire (MSLQ) [17]. MSLQ consists of fifteen subscales designed from classic social-cognitive learning theories and is widely used as a self-report instrument for measuring student motivation and learning strategies and for subsequently predicting academic performance [6]. The self-efficacy scale comprises of eight statements which assess both expectancy for success in the course and self-efficacy. They include judgments about the student's abilities to

[2] http://csunplugged.org/
[3] link removed for double blind submission
[4] link removed for double blind submission
[5] link removed for double blind submission
[6] Link removed for double blind

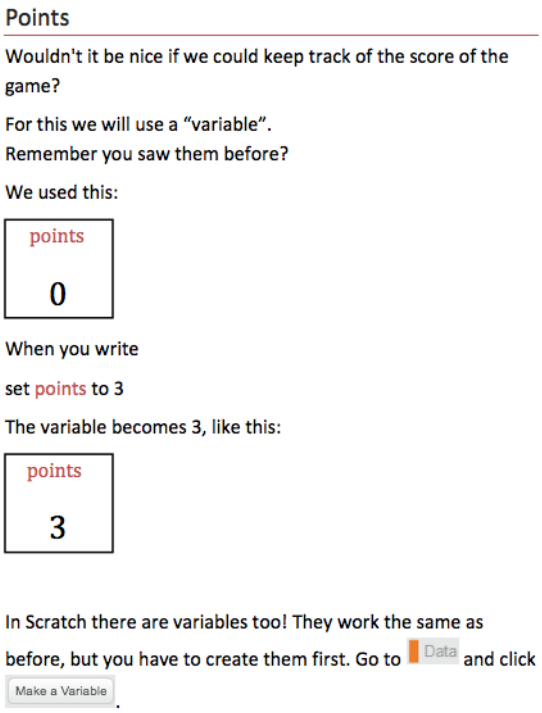

Figure 4: Explanation of unplugged concepts into Scratch

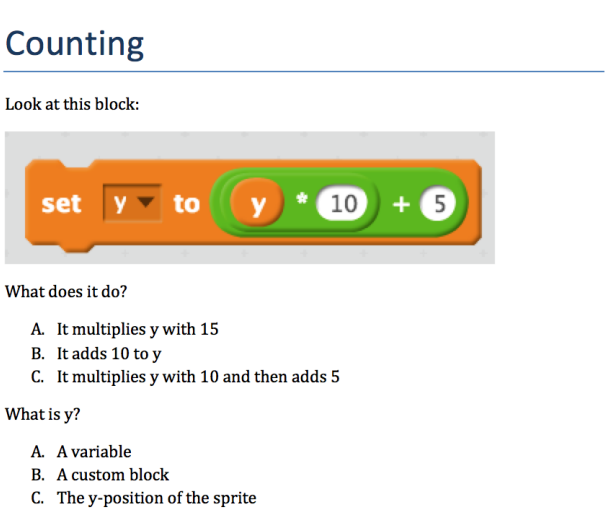

Figure 5: Questions in the endterm testing the concepts operators and variables

accomplish tasks, as well as student's confidence in their skills to perform those tasks.

Using questionnaires at the beginning (Week 2) and the end (Week 8) of the course, the students rated themselves on a seven point likert-scale from 'not at all true of me' to 'very true of me'. As specified in the MSLQ, students' self-efficacy scores were computed

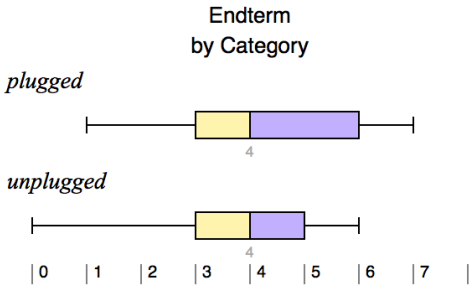

Figure 6: Scores per group on the endterm exercises, out of 8 possible points. Center lines indicate the medians; boxes indicate the 25th and 75th percentiles; whiskers extend 1.5 times the interquartile range from the 25th and 75th percentiles. n = 17, 18 sample points.
T-test shows that groups do not differ significantly.

by taking the average of the points given to the eight statements of each questionnaire.

3 RESULTS

3.1 Mastering of Programming Concepts

Figure 6 shows the results of both groups on the endterm, after eight weeks of lessons.

A Shapiro-Wilk test showed that both the plugged first and the unplugged first sample follow the normal distribution (p=0.125) with means of 4.29 and 3.73 respectively, and standard deviations of 0.87 and 0.80. Furthermore the two groups have equal variance as demonstrated by Levene's test. We therefore use a t-test to determine whether there is a difference between the samples. The test resulted in a p-value of 0.311, meaning we cannot reject $H1_0$. In other words children in both groups perform similar, and there is no effect measured of the plugged first versus unplugged first treatment on their understanding of programming concepts.

In addition to the total score on all questions, we also separated the results for the six programming concepts. On the individual concepts too we measured no significant differences.

> After eight weeks, we measure no difference in understanding of programming concepts between the plugged first and unplugged first group.

3.2 Self-Efficacy

Figure 7 shows the aggregated self-efficacy scores of the children after eight weeks, calculated as the averages of the seven-point likert-scale replies given to the 8 self-efficacy MSLQ scale statements as described in Section 2.5.

A Shapiro-Wilk test (p=0.327) showed that both the plugged and the unplugged first sample follow the normal distribution with means of respectively 5.05 and 5.75 and standard deviations of 0.54 and 0.32. Furthermore the two groups have equal variance as

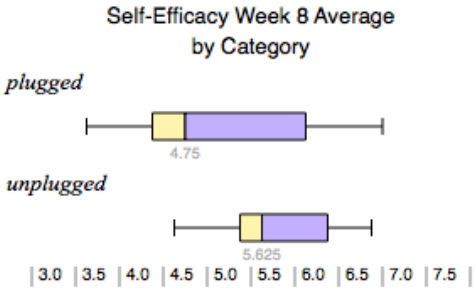

Figure 7: Self-efficacy scores at the end of the course per group. Scores are calculated as the averages of the seven-point likert-scale replies given to the 8 self-efficacy MSLQ scale statements. Center lines indicate the medians; boxes indicate the 25th and 75th percentiles; whiskers extend 1.5 times the interquartile range from the 25th and 75th percentiles. n = 18, 17 sample points. T-test shows that groups differ significantly.

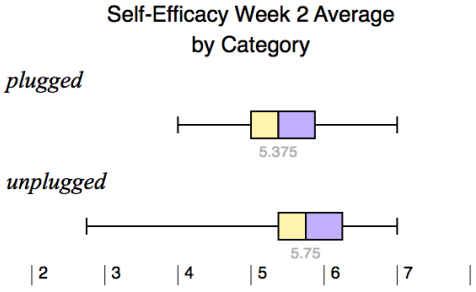

Figure 8: Self-efficacy scores at the beginning of the course per group. Scores are calculated as the averages of the seven-point likert-scale replies given to the 8 self-efficacy MSLQ scale statements. Center lines indicate the medians; boxes indicate the 25th and 75th percentiles; whiskers extend 1.5 times the interquartile range from the 25th and 75th percentiles. n = 18, 17 sample points. T-test shows that groups do not differ significantly.

demonstrated by Levene's test. We therefore use a t-test to determine whether there is a difference between the samples. The test resulted in a p-value of 0.023, meaning we must reject $H1_0$. In other words children in the unplugged first group have a significantly better sense of self-efficacy in programming.

At the beginning of the course we also measured the self-efficacy, and then we measured no difference, as shown in Figure 8.

After 8 weeks, children in the unplugged first group measure significantly better in self-efficacy beliefs.

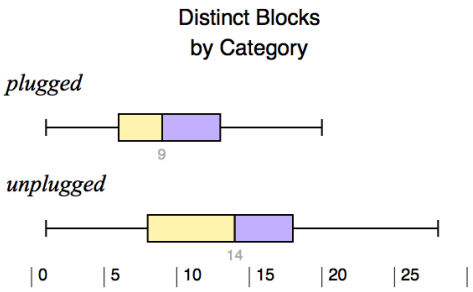

Figure 9: Distinct number of Scratch blocks per group in the end projects children built in weeks 7 and 8. Center lines indicate the medians; boxes indicate the 25th and 75th percentiles; whiskers extend 1.5 times the interquartile range from the 25th and 75th percentiles. n = 17, 18 sample points. T-test shows that groups differ significantly.

3.3 Block Vocabulary

Figure 9 shows the number of distinct Scratch blocks appearing in the end projects that children worked on in weeks 7 and 8.

A Shapiro-Wilk test showed that both the plugged and the unplugged sample follow the normal distribution (p=0.087) with means of 9.47 and 14.22 respectively, and standard deviations of 2.55 and 4.08. Furthermore the two groups have equal variance as demonstrated by Levene's test. We therefore use a t-test to determine whether there is a difference between the samples. The test resulted in a p-value of 0.047, meaning we must reject $H3_0$. In other words children in the unplugged first group use different Scratch blocks. Specifically, they use more different blocks than the Scratch first group.

Figure 10 shows the division over the categories of blocks that Scratch defines. Children in the unplugged first group use more different Motion blocks, which move the sprites over the 2d plane. They also use more Control blocks which include loops and conditionals, and more Looks blocks.

In the projects created in weeks 7 and 8, children in the unplugged first group use more different Scratch blocks than children in the plugged first group.

4 DISCUSSION

In the above, we have described an experiment in which we compare programming competency, self efficacy and vocabulary of children that started with four weeks on Scratch lessons, versus four weeks of unplugged lessons. In this section, we discuss the results.

4.1 Classroom atmosphere

A difference between the two classrooms that goes beyond measuring performance and efficacy is the 'vibe' of the classrooms, which we attribute to the presence of computers, since all other factors where equal. The groups had the same teachers and used the same physical classroom. We observed a huge difference between the plugged first group where computers where present. In this

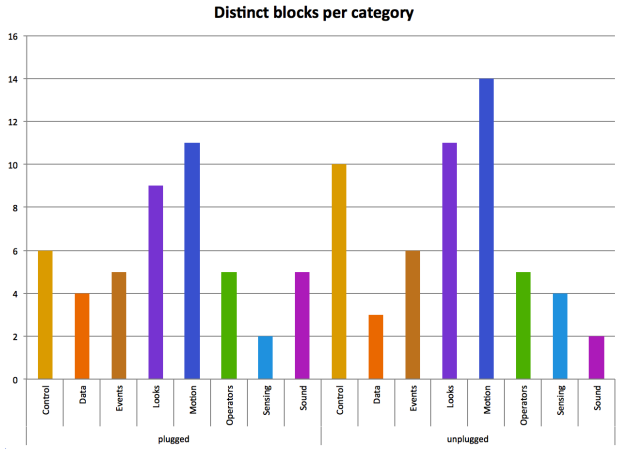

Figure 10: Distinct number of Scratch blocks per group, in 8 of the 10 different categories of blocks that Scratch defines.

group, the children were more excited by the presence of computers, which are not commonly used in class. This lead to increased excitement, but also meant that children were often distracted by the the games they created, playing them rather than programming. In the unplugged first group the vibe was more calm and classroom like, which could explain why children got more confidence in their mastering of concepts. This might have resulted in the fact that children in the unplugged first group could easier reach a state of flow, where a person is performing a task that is not too easy or too hard, and is completely engaged in the task [7]. Kinnunen and Simon [10] describe a similar phenomenon. They explored how students experience programming assignments in CS1 courses and described the true novice experience of encountering a difficulty as similar to being hit by lightning: an unexpected, shocking experience which leaves one dazed and confused, largely affecting the student's self-efficacy beliefs. Similar observations on the reciprocal relationship between performance and self-efficacy beliefs were made in [13]. Maybe the unplugged first group had a more "natural", "paved way" to learning programing, avoiding feeling confused, encountering obstacles or dealing with the increased cognitive load that the Scratch web interface imposes in comparison to unplugged learning.

4.2 Programming and Gender

Regarding the effect of gender on programming performance and orientation, a number of studies have found it to be significant. For example, strong social support and high self-efficacy have been found to be associated with strong orientation toward CS careers [19], while Lishinski et al. found that female students adjust their self-efficacy beliefs earlier in courses, which suggests that responses to early failures could be causing them to disengage from CS [13]. In week 2 of our study, we measured no difference in self-efficacy between boys and girls. In week 8, girls have lost more self-efficacy than boys, however this difference is not significant. We furthermore find no significant differences in the performance between students of different genders.

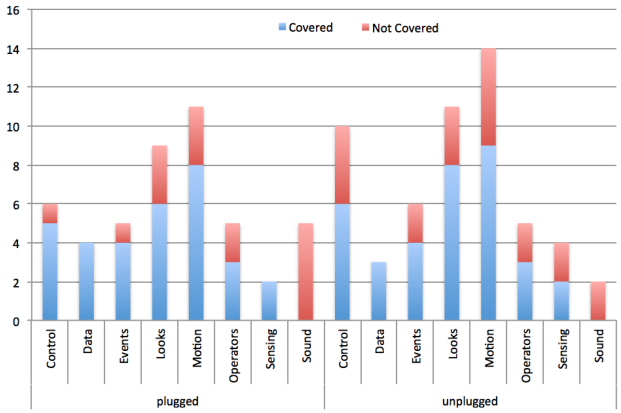

Figure 11: Distinct number of Scratch blocks per group covered and not covered by the coursework.

4.3 Self-efficacy and Vocabulary

It is interesting that we measure an increase both in the self-efficacy of children in the unplugged first group and in the diversity of the blocks they use. We hypothesize that these two observations might be related. We think that because children have increased confidence, they feel more confident exploring different Scratch blocks. This is illustrated by Figure 11, which shows the use of blocks by children in the two groups, divided into the blocks that we did and did not cover in the coursework, again divided over the categories of Scratch blocks. As you can see, the unplugged group uses more blocks that are not covered in the lessons, potentially indicating that they feel comfortable exploring those.

4.4 Threats to Validity

A threat to the external validity of our evaluation concerns the representativeness of children in the experiment. To mitigate this effect we have used three different classes and divided them in two random groups, however all children are pupils of one school, which means they might not be representative of all children or even all children in this country. We plan to repeat this study with a larger group of children later this year.

With respect to internal validity, one of the threats is the quality of the materials. It could be that the unplugged lessons are easier than the plugged materials, resulting in higher self-efficacy. However, since the children in the unplugged first group demonstrate the same understanding of concepts, this seems not to be the case.

5 RELATED WORK

There is a limited number of studies that have tried to evaluate teaching programming concepts unplugged.

Thies and Vahrenhold [21] researched the CS unplugged materials specifically as a method of teaching. They taught a group of 25 children aged 11 and 12 in a classroom setting, and used the CS Unplugged activities to teach half the students, and used alternative methods for the other half of the students, including traditional computer science textbooks, among which Algorithms Unplugged [22]. They find that CS unplugged is as effective as the

alternative methods, since there was no significant difference in achievement between the group who learned with CS Unplugged activities and the group who learned with alternative materials. However, they do not measure subsequent behavior in a plugged programming environment like we do.

A number of studies have been carried out on teaching programming concepts to novice programmers with block-based languages in general, and Scratch in particular. Scratch was taught in middle school classes containing a total of 46 students in the study presented in [15]. Evaluating the internalization of programming concepts, it was found that students had problems with concepts related to initialization, variables and concurrency. Maloney et al. [14] taught Scratch as an extracurricular activity, in an afterschool clubhouse. By analyzing the 536 students' projects for blocks that relate to programming concepts, they found that within the least used ones are boolean operators and variables. Using the performance results from an online introductory Scratch programming course that was run recently, Hermans and Aivaloglou found that students over 12 years of age perform significantly better in questions related to operators and procedures [9].

Apart from projects created during courses, other works analyze the public repository of Scratch programs for indications of learning of programming concepts. Yang et al. examined the learning patterns of programmers in terms of block use over their first 50 projects [24]. In [8], the use of programming concepts was examined in relation to the level of participation, the gender, and the account age of 5 thousand Scratch programmers. [1] analyzed 250 thousand Scratch projects in terms of complexity, programming concepts and *code smells*, bad programming practices. Seiter and Foreman [20] proposed a model for assessing computational thinking in primary school students and applied it on 150 Scratch projects, finding that design patterns requiring understanding of parallelization, conditionals and, especially, variables were under-represented until a certain age.

The relationship between performance and self-efficacy in the computing education domain is widely studied. Lishinski et al. recently examined the interaction of self-efficacy, intrinsic and extrinsic goal orientation, and metacognitive strategies and their impact on students performance in a CS1 course, and found that females' self-efficacy had a different connection to programming performance than that of their male peers [13]. In [18] it is found that, in the context of a CS1 course, self-efficacy is influenced by previous programming experience and increases as a student progresses through an introductory programming course, while self-efficacy affects course performance. The MSLQ was used in a study with 39 university students in their introductory programming course, where it was found that, of all motivational and learning strategies scales on the MSLQ, self-efficacy had the strongest correlation with the students' course performance [23]. Examining the relationships between MSLQ scores and academic performance, [6] found that the self-efficacy scores had within the highest observed validities for grades in individual classes, along with the scores for effort regulation and time and study environment.

The effects of using Scratch in the self-efficacy beliefs of students is also studied by Armoni et al., who found that learning programming through Scratch in middle school greatly facilitated learning professional textual programming languages (C# or Java) in secondary school, while students with Scratch experience were observed to display higher levels of motivation and self-efficacy [2]. On university-level students, however, [11] examined the effect of introducing Scratch-based game activities at the introductory programming course and found that they contributed significantly to their C++ programming performance, but had no effect on their sense of self-efficacy towards the C++ programming language.

6 CONCLUDING REMARKS

The goal of this paper is to explore the impact of starting with unplugged lessons before programming in Scratch or another programming system on the computer. To that end we have designed and executed a controlled experiment in which we randomly divided 35 children aged 8 to 12 into two groups. One group was taught with Scratch for the first four weeks (plugged first) while the other did exercises on paper for four weeks (unplugged first). After these lessons, both groups used Scratch for four weeks, two weeks of repeated concepts lessons and two weeks of free program creation.

We find that after these eight weeks, there is no difference in performance on the understanding of programming concepts. However, the unplugged first group shows more self-efficacy and uses a wider vocabulary of Scratch blocks, including more blocks not covered by the lessons.

The contributions of this paper are as follows:

- Lessons teaching loops, conditionals, procedures, broadcasts, parallelization and variables in a plugged and unplugged fashion. (Section 2.4)
- A controlled experiment comparing the use of these plugged and unplugged lessons (Section 2, Section 3)

Our current research gives rise to several directions for future work. Firstly, we want to replicate these findings on a larger scale. Furthermore, we are interested in understanding more deeply why children in the unplugged first group show improved self-efficacy beliefs. An interesting way of studying this would be to use Scratch printouts and lessons in the unplugged first group. That way we could study if it is indeed the atmosphere in the classroom without computers that influences children's self-efficacy or whether it is due to the unplugged teaching methods.

REFERENCES

[1] Efthimia Aivaloglou and Felienne Hermans. 2016. 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 (ICER '16)*. ACM, 53–61. DOI: https://doi.org/10.1145/2960310.2960325

[2] Michal Armoni, Orni Meerbaum-Salant, and Mordechai Ben-Ari. 2015. From Scratch to Real Programming. *Trans. Comput. Educ.* 14, 4, Article 25 (Feb. 2015), 15 pages. DOI: https://doi.org/10.1145/2677087

[3] Computing at School Working Group. 2012. *Computer Science: A Curriculum for Schools.*

[4] A. Bandura. 1977. Self-efficacy: toward a unifying theory of behavioral change. *Psychological review* 2, 84 (1977), 191–215.

[5] Shari L. Britner and Frank Pajares. 2006. Sources of science self-efficacy beliefs of middle school students. *Journal of Research in Science Teaching* 43, 5 (2006), 485–499. DOI: https://doi.org/10.1002/tea.20131

[6] Marcus Cred and L. Alison Phillips. 2011. A meta-analytic review of the Motivated Strategies for Learning Questionnaire. *Learning and Individual Differences* 21, 4 (2011), 337 – 346. DOI: https://doi.org/10.1016/j.lindif.2011.03.002

[7] Mihaly Csikszentmihalyi. 2000. *Beyond boredom and anxiety.* Jossey-Bass.

[8] Deborah A. Fields, Michael Giang, and Yasmin Kafai. 2014. Programming in the Wild: Trends in Youth Computational Participation in the Online Scratch Community. In *Proceedings of the 9th Workshop in Primary and Secondary Computing Education (WiPSCE '14)*. ACM, 2–11. DOI: https://doi.org/10.1145/2670757.2670768

[9] Felienne Hermans and Efthimia Aivaloglou. Teaching Software Engineering Principles to K-12 Students: A MOOC on Scratch. In *Proceedings of the 39th International Conference on Software Engineering Companion (ICSE '17)*.

[10] Paivi Kinnunen and Beth Simon. 2010. Experiencing Programming Assignments in CS1: The Emotional Toll. In *Proceedings of the Sixth International Workshop on Computing Education Research (ICER '10)*. ACM, New York, NY, USA, 77–86. DOI: https://doi.org/10.1145/1839594.1839609

[11] O. Korkmaz. 2016. The Effects of Scratch-Based Game Activities on Students' Attitudes, Self-Efficacy and Academic Achievement. *International Journal of Modern Education and Computer Science* 8, 1 (2016).

[12] Robert W Lent and Gail Hackett. 1987. Career self-efficacy: Empirical status and future directions. *Journal of Vocational Behavior* 30, 3 (1987), 347 – 382. DOI: https://doi.org/10.1016/0001-8791(87)90010-8

[13] Alex Lishinski, Aman Yadav, Jon Good, and Richard Enbody. 2016. Learning to Program: Gender Differences and Interactive Effects of Students' Motivation, Goals, and Self-Efficacy on Performance. In *Proceedings of the 2016 ACM Conference on International Computing Education Research (ICER '16)*. ACM, New York, NY, USA, 211–220. DOI: https://doi.org/10.1145/2960310.2960329

[14] John H. Maloney, Kylie Peppler, Yasmin Kafai, Mitchel Resnick, and Natalie Rusk. 2008. Programming by Choice: Urban Youth Learning Programming with Scratch. In *Proceedings of the 39th SIGCSE Technical Symposium on Computer Science Education (SIGCSE '08)*. ACM, 367–371. DOI: https://doi.org/10.1145/1352135.1352260

[15] Orni Meerbaum-Salant, Michal Armoni, and Mordechai (Moti) Ben-Ari. 2010. Learning Computer Science Concepts with Scratch. In *Proceedings of the Sixth International Workshop on Computing Education Research (ICER '10)*. ACM, New York, NY, USA, 69–76. DOI: https://doi.org/10.1145/1839594.1839607

[16] F. Pajares and M. D. Miller. 1994. Role of self-efficacy and self-concept beliefs in mathematical problem solving: A path analysis. *Journal of educational psychology* 2, 86 (1994), 193.

[17] R. Paul, S. Smith, M. L. Genthon, G. G. Martens, C. L. Hauen, G. G. Martens, M. Genthon, and P. Wren. 1991. *Technical Report No. 91-B-004: A Manual for the Use of the Motivated Strategies for Learning Questionnaire (MSLQ).* Technical Report. The Regents of The University of Michigan.

[18] Vennila Ramalingam, Deborah LaBelle, and Susan Wiedenbeck. 2004. Self-efficacy and Mental Models in Learning to Program. In *Proceedings of the 9th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education (ITiCSE '04)*. ACM, New York, NY, USA, 171–175. DOI: https://doi.org/10.1145/1007996.1008042

[19] Mary Beth Rosson, John M. Carroll, and Hansa Sinha. 2011. Orientation of Undergraduates Toward Careers in the Computer and Information Sciences: Gender, Self-Efficacy and Social Support. *Trans. Comput. Educ.* 11, 3, Article 14 (Oct. 2011), 23 pages. DOI: https://doi.org/10.1145/2037276.2037278

[20] Linda Seiter and Brendan Foreman. 2013. Modeling the Learning Progressions of Computational Thinking of Primary Grade Students. In *Proceedings of the Ninth Annual International ACM Conference on International Computing Education Research*. ACM, 59–66. DOI: https://doi.org/10.1145/2493394.2493403

[21] Renate Thies and Jan Vahrenhold. 2013. On Plugging "Unplugged" into CS Classes. In *Proceeding of the 44th ACM Technical Symposium on Computer Science Education (SIGCSE '13)*. ACM, New York, NY, USA, 365–370. DOI: https://doi.org/10.1145/2445196.2445303

[22] Berthold Vcking, Helmut Alt, Martin Dietzfelbinger, Rdiger Reischuk, Christian Scheideler, Heribert Vollmer, and Dorothea Wagner. 2011. *Algorithms Unplugged* (1st ed.). Springer Publishing Company, Incorporated.

[23] Christopher Watson, Frederick W.B. Li, and Jamie L. Godwin. 2014. No Tests Required: Comparing Traditional and Dynamic Predictors of Programming Success. In *Proceedings of the 45th ACM Technical Symposium on Computer Science Education (SIGCSE '14)*. ACM, New York, NY, USA, 469–474. DOI: https://doi.org/10.1145/2538862.2538930

[24] Seungwon Yang, Carlotta Domeniconi, Matt Revelle, Mack Sweeney, Ben U. Gelman, Chris Beckley, and Aditya Johri. 2015. Uncovering Trajectories of Informal Learning in Large Online Communities of Creators. In *Proceedings of the Second ACM Conference on Learning @ Scale*. ACM, 131–140. DOI: https://doi.org/10.1145/2724660.2724674