Reduce the gender gap in computer science education using creative programming

LOUISE HELLBERG
Reduce the gender gap in computer science education using creative programming

LOUISE HELLBERG
Reduce the gender gap in computer science education using creative programming

LOUISE HELLBERG

Women are lacking in Science, Engineering, Technology, and Mathematics (STEM). One of the proven solutions to reduce this gap is to offer extra support to the students needing it. Further, creativity has been shown to give positive effects on females' participation in STEM. In this thesis, a creative tool for programming education in junior high schools in Sweden was developed to give alternatives to how programming is taught and further evaluate the possible contributions it can give. The tool was tested among 14 students participating in a face-to-face workshop. To further evaluate the tool use, the study participants took part in interviews and a survey. The results show that further iterations of the tool would have been needed since the tested tool was not perceived as easy to use. Regarding perceived usefulness, the tool is on the right path since the results indicate it was fun to use but again, improvements need to be done. The contributions made are therefore an exploration of the creative alternative to programming learning in the Swedish school. Furthermore, to find out what improvements needed to be done to offer an, by users, accepted tool. Offering creative programming tools will hopefully attract a more diverse group of people to programming and decrease the gender gap within computer science.

1 INTRODUCTION

A part of the digital transformation is UNESCO’s work to get more girls to become leaders in STEM [53], and therefore, the issue of gender disparities within STEM has called for special investigation. UNESCO has highlighted some

© 2022 Copyright held by the author.
strategies to solve the issue, for example, that women’s participation and continuation in STEM needs to be improved. Other methods are to ensure that countries can deliver and pursue gender-responsive STEM education and that STEM education for women is considered necessary in all countries.

In Sweden, only 11% of those who graduated in computer science in 2018/2019 were women [6]. Of all female students enrolled in higher education, 30% choose a major within STEM and only 3% choose information and communication technologies (ICT) as their major. Overall, the mentioned statistics show that new ways to encourage people of different backgrounds and genders to choose an education in STEM are needed, especially in computer science.

Another reason for the importance of finding working methods to teach programming is that technology is a big part of humanity’s everyday life. For people to keep up in society schools need to adjust their education within technology to fit all students [38, 54]. Actions are needed on several fronts to get a more diverse tech industry and a more technically educated population. Therefore, the current study will mainly focus on one front, schools, and how schools are working with programming in Sweden to fulfill the future’s needs. The study will also focus on the environmental factors indirectly connected to programming, such as stereotypes and support from teachers since it affects how the students are performing in programming. Support from teachers means getting help from teachers to learn programming and having someone push in the right direction. The topic is chosen due to the gender gap within tech and the fact that schools have the opportunity to reach everyone and educate them with the tools needed.

There have been four essential factors for getting positive effects on teaching and learning when using digital tools. The factors are the teacher’s digital competence, ability to lead school work, integration of digital tools and resources in teaching, and giving students clear but achievable challenges. These factors strengthen the need for competent teachers to choose the best tool and use digital tools in school [19]. Even more research is needed as there is little research on how digital tools can contribute to increasing students’ achievement of learning goals. The current knowledge is based on smaller studies or experiences gained in practice.

Research has shown that it is during middle school math and science-related disciplines, like computer science, begin to be deselected by girls [28]. Stereotypes of math and science being masculine or having masculine traits have been proven to be one of the explanations for the deselection. For example, females who did see a potential of choosing a STEM-related major at the university did not have as strong masculine images of math, physics, and chemistry as the ones aiming for a non-STEM-related major [33]. Among males, this only applied to mathematics, not science. It has also been shown that current stereotypes within computer science, together with a lower sense of belonging to those stereotypes are a reason for girls to not enroll in computer science classes in high school [34]. However, earlier research proved that more girls were interested in computer science classes when the classroom did not project stereotypes of the field, and the boys contained the same interest as in a stereotypical classroom [34]. Other reasons that have strong indications of why girls deselect science-related disciplines have been observed in classrooms in Sweden. There is strong evidence that the deselection are due to the adaptation made to friends’ preferences [40]. At the same time, girls lean on to retain their interest in STEM if other girls share their interest. Due to the amplification of pre-existing interests, it becomes more difficult to break the pattern of a gender gap within STEM [40]. To address this gender gap, several digital technology-supported solutions have recently been offered, such as DojaCode [2] and Imagi Labs [4].

DojaCode was released in 2021 and is a music video focusing on encouraging young women to try out programming by having them change details in the video by using code. DojaCode has been developed in a collaboration between the artist Doja Cat and the organization ‘Girls Who Code’, which is working to fill the gender gap in technology. The main purpose of their work is to expose women to the programming profession and show them the creative side of it [1].

Another initiative has been using creative methods to see if it benefits women’s interest in STEM. It was proved that design thinking was great to understand what a STEM profession could imply, increasing interest and making
young women more confident in the subject, among other things [30]. When it comes to digital programming solutions, Scratch, Alice, and Code.org have been used with positive results to women’s interest in technology [8, 29].

Inspired by the popular launch of DojaCode, this study aims to understand how students experience the use of popular culture such as videos with programming exercise adaptations in a creative programming learning tool. In other words, what is the perceived usability of the tool? Also, is the design of the creative teaching tool easy enough to be used by high school students? Being creative in this study means that the users did storytelling and created videos by using code.

First, the background sections uncover what strategies for addressing the needed digitalization in Sweden are done today as well as what influences children to later choose, or not choose, a career path within STEM. Furthermore, the background presents how programming has been taught to beginners, and how teachers in Swedish schools teach programming, including a presentation of current tools that are being used.

Second, the methods for data collection and analysis are presented together with the design process for the integrated coding exercises. After that, the results from the interviews and questionnaire are uncovered. Then the tool and its design are discussed taking into account the research results on programming learning.

2 BACKGROUND
This section presents an overview of digitalization in Swedish schools as well as what is important when teaching programming to beginners. Further, the gender perspective and examples of methods and initiatives to increase women in technology are given. Lastly, the technology acceptance model (TAM) is introduced.

2.1 Digitalization goals for schools in Sweden
In 2017, the Swedish government presented a list of overall goals for the digitalization of the school system in Sweden in their report ‘National Digitization Strategy’ [54]. The goals were set to make the Swedish school system use the opportunities that come with digitalization in the best possible way until the year 2022. The report covers three focus areas. The first one is digital competence for everyone in the school system. The meaning of digital competence is defined by the Digitalization Commission [54] as the “extent to which an individual is familiar with digital tools and services and has the ability to keep up with digital development and its impact on one’s life”. Because it is expected that digital tools will be even more integrated into our daily lives in the future than today, critical thinking will be even more important as we are going toward a more digitized world. If people do not have the digital competence needed, there will be people struggling with everyday life. Here, the school has a significant role. For this to be possible, pupils and teachers should have satisfying and equal access to digital tools, which is the second focus area. Further, the Digitalization Commission has pointed out that increased elements of IT in school promote girls’ interest in IT and, in the long run, contribute to reducing the skewed recruitment to higher education with an IT focus [16]. That means that the inclusion of a gender equality perspective in work with digitalization is vital for achieving the gender equality policy goal of gender equality education [51]. The third digitalization area is to research the impact of digitalization on teaching and learning, which will support the development of activities and initiatives [54]. Not least in programming since digital tools are fundamental when learning it.

Based on these goals, The Swedish Association of Local Authorities and Regions (SALAR) created a national action plan with 18 different suggested initiatives to implement the digitization strategy in practice. It was handed over to the Swedish government in 2019, and two years later a follow-up was made. The follow-up showed that ten initiatives were a work in progress whereas three of them were decided and one partly decided. Three initiatives were still under investigation and five had not been started yet. The status of the initiatives shows that there is still more work to do with the digitalization of schools, which is necessary for a better education in programming.
2.2 Programming in the curriculum

Since 2017, the Swedish curriculum states that compulsory school students should learn about programming in technology and mathematics [44, 56]. In grade one to three of primary school, one of the goals in technology is to be able to control objects with programming. The pupils should also be able to understand what a computer is used for and know about some of its fundamental parts. In algebra, they should grasp how unambiguous step-by-step instructions can be constructed, described, and followed as a basis for programming. In grade four to six in primary school the goals are a bit more advanced. For example, one is about being able to control their constructions with programming, and they should also understand how computers are controlled by programs and how they can be connected to networks. Furthermore, they should understand how algorithms can be created and used when programming. When becoming a high school student in grade seven to nine, the level of knowledge is slightly more complex. They should be able to understand the bigger picture, meaning how technical solutions are using technology and how they can be programmed. Programming-wise, they should be able to construct their constructions where control and regulation are applied, among other things, with the help of programming. In mathematics, they should be able to know how algorithms can be created and used when programming. In high school, a new perspective is added because they should also be able to reflect on how cultural stereotypes about technology affect people’s career choices. This is relevant since there is a gender gap in the tech industry.

2.3 How should programming be taught to beginners?

When programming, a good developer has a toolbox of common problems and their solutions close at hand [9]. It is then these solutions that are used when breaking down a program and choosing which algorithms to use and create [46]. A beginner does not have the toolbox, and therefore one step on the way is to create it.

There have been several studies on teaching programming for beginners and what elements to introduce first [7]. When trying to explain why so many students failed their beginner course in programming, it seemed to be due to the subject’s complexity, and that one needed to master several new elements and how they are connected. This made the students give up if the teacher taught too fast, if the support was too little, or sometimes if relevant examples were missing [42, 43]. Therefore, elements such as loops, variables, repetition statements, and conditional statements that have been proven difficult to understand should be introduced later. The easier elements such as easy sequences, output, and control are more suited to be introduced first, as well as a few new concepts and structures at a time that the student needs to master before continuing to new learnings [7].

There are other factors to think of when teaching programming, and one of them is computer anxiety. Computer anxiety, a phenomenon that is described as a feeling of stress and anxiety when working with a computer, is a phenomenon that affects women the most and is still a phenomenon that has not been fully explained [39]. What can be concluded though is that it is very important to proceed slowly at the beginning of programming education, meaning that it should not get too difficult too fast, to avoid computer anxiety. Also, students in need of help should get strong individual support since facing too many new, nonunderstandable terms and concepts may reduce their self-esteem [7, 39, 52].

Another critical part of teaching and learning programming is to let the student succeed and feel satisfied with their progress to avoid anxiety feelings [17]. It was investigated how the upcoming feelings when using Scratch combined with Arduino affected twelve-year-old girls’ attitudes toward participating in similar activities again [23]. In the study, it was confirmed that feeling happiness during the activity gave a positive effect on wanting to participate again in a similar activity. The results were the opposite when the girls felt anxiety but enjoyment did not affect their intention to participate again.
Another way of working that has been shown to often benefit women is pair programming. Especially when working with another woman, as they feel supported by their partner when trying out uncertain solutions and find it easier to ask their teacher for help [24, 35].

Classes separated by gender have been shown to have similar effects as pair programming. The distribution provides more space for girls to experiment and ask questions during laboratory sessions than when laborating with boys, where they often take a passive spectator seat instead [27]. It is also a fact that classes separated by gender impact the students’ interests. Girls in same-sex schools had a much bigger interest in physics, and boys were more interested in biology than students in a school with both genders [48]. In schools with both genders, there was a distinct difference in the students’ subject interest in natural science and technology between boys and girls. Girls cared more about subjects such as the body, health, the environment, and sustainable development, while boys cared more about technology [26].

Further, students’ attitudes toward technology and their motivation to learn technology have been studied [8]. The result of that study showed that women seek meaning and relevance in their learning and see the computer as a tool to achieve something else. Whereas the boys find motivation in understanding how the computer works and how to program.

When asking university students how they wanted to be examined on their programming skills out of three ways, the answers differed depending on gender [57]. Women preferred to be examined by solving a reality-related problem. Among men, a more popular alternative was to create a game. The last option, a mathematical problem, was the least popular one but slightly more popular among women with 14 % preferring that examination alternative compared to 9 % among men.

The support of teachers and parents is important for students’ enrollment in STEM. Both personal, behavioral, and environmental factors affect women in such a way that results in lower participation in STEM education [50]. In the same study, teachers and parents have been seen to have an important role in young people’s future choices. The study found that 47 % of the female students and 39 % of the male students claimed that with the right support, a STEM degree could have been possible. The study results support that girls and boys think differently about their competence. Young women tend to think that they are not competent enough to choose a major within STEM [50], which goes hand in hand with the current gender stereotypes.

2.4 Gender stereotypes

Before the decrease of women in computer science, back in the 1950s, gender stereotypes worked in women’s favor. At this time, employers wanted candidates that were good at math, had logical thinking and were meticulous. They wanted a woman since these qualities were needed when knitting, for example, which women were good at. Nowadays the stereotypes work in the opposite direction in such a way that science is associated with masculine traits in the western world [11, 37]. Several studies support that these stereotypes about math and science make young women avoid starting a career in STEM [31, 41]. In a study in computer science, when the perception of the discipline was changed it resulted in an increased interest in studying computer science among girls [10]. But it is not only traditional gender norms that affect children’s choices, their friends also have an important role.

2.5 Friends and peers

Friends in school can have a large impact on children’s future. It has been seen in classrooms in Sweden that the proportion of girls preferring STEM subjects decreased by four percent within a year (from 19 to 15 percent) whereas, among the boys, it only decreased by one percent (from 21 to 20 percent) [40]. By studying the children’s friendships, there was strong evidence that those changes in preferences can be explained by their friends’ preferences since children tend to adjust their interests to those. Moreover, girls’ interest in STEM subjects tended to be kept when other girls
in the classroom shared the same interest. Another research reported that friends’ gender norms also have an impact on education choices after secondary education [55]. Traditional gender norms in a group of friends affected girls immensely to not choose STEM. Boys were less affected by these norms and in a group with only boys, the likelihood of them choosing STEM increased.

2.6 How Swedish schools are working with programming

In Sweden, it is mandatory to integrate programming into education. To simplify for the teachers, the National Agency for Education is taking some actions. For example, they provide programming videos with good programming exercises for teachers to draw inspiration from for their lectures. Teachers are also able to educate themselves via web-based courses about programming [45].

2.7 An example of change in a technical education that increased girls’ motivation

Carnegie Mellon is a school that wanted to attract more women to computer science in the US. After evaluating a study where women were interviewed, the school decided to take action based on the results [21]. The main change made was to separate the classes by their pre-knowledge in coding since many women did not have any experience with computers. They also offered extra tutoring and modified the courses so that they had a connection to and practical use in the real world. Some changes were also made in the admission process, no longer favoring experience in coding. These changes were a huge success. The percentage of women entering computer science increased from 7 percent to 42 percent and other schools followed the same track with the same result [21].

![Figure 1: Teaching model of programming ability development. G stands for the education level, A represents the four important programming abilities, M stands for a teaching method, T stands for the programming tool, and E stands for the evaluation method. The arrows indicate what combinations that are suitable for a certain education level and ability [49].](image-url)
2.8 Teaching model for developing programming abilities

A teaching model (see Fig 1) was developed to show how to cultivate four important abilities when programming: Cognition (A1), metacognition (A2), operation (A3), and communication (A4), based on the student’s education level [49]. The model makes it easy to see when to use what teaching method and what kind of tool that can be appropriate for that method. For students in grade six to eight (G3), the model says that to grow the cognition ability, problem-based learning (M2) or scaffolded (M6) are the best teaching methods. The block-based tool (T2) is a great choice for the problem-based method together with observation and interview (E5) as the evaluation method. Whereas when using the scaffolded method, Logo (T3) should be used as a tool and either observation and interview or programming ability test (E1) as evaluation of the students’ abilities. To make secondary school students grow their metacognitive ability, project-based learning (M3) should be used together with a block-based tool and the observation and interview method for assessment. Further, the operation skill can be developed by using a problem based learning method, also in combination with a block-based tool and observation and interview. For the last ability, communication, either game based learning (M1) or problem based learning should be used according to the model. The recommended tool is, again, a block based tool with observation and interview as the evaluation method.

2.9 Initiatives to improve programming knowledge

There are many options for a person who wants to learn to program. There is no longer a need to get into school for several years, instead, there are many coding schools online, lots of videos, and organizations aiming at getting more people into coding. One of the many existing initiatives is Dojacats music video, and there is also software such as Scratch and imagiLabs.

Other organizations that work to gain children’s attention to the world of coding in Sweden are Kodcentrum and Hello World, which creates coding events for kids of primary school age. Tjejer kodar, Teknikkvinnor and Women in Tech are working towards more women in the male-dominated tech field by arranging meetups, presenting female role models, and encouraging girls to try programming.

2.10 Design as a way to inspire programming

Several factors must be considered when designing for children between the ages of 11 and 13. At that age, gross motor skills, such as movements involving big muscle groups, and fine motor skills, for example, handwriting, are well developed. Anyhow, motor coordination is only partially developed between the ages of 9 and 12, suggesting that the combination of gross and fine motor skills could be difficult for them [32]. That is, small objects to be clicked on should be avoided. Also, colors convey emotions to the user, and color choices are therefore important [5].

A design thinking intervention led to several findings on how young women’s interest in STEM had increased and suggestions on how to continue increasing the enthusiasm [30]. The findings included an increased interest and a changed perception of engineering. The perception of STEM was also changed as it was understood that education within STEM could lead to many different professions and that those professions were more fluid and did not always contain technical tasks. It was also more understood that design could be used in engineering to find solutions to improve the world and a higher percentage expressed a desire to pursue options within STEM. Furthermore, the girls in the study were more confident in creative thinking. It was no longer as scary to fail and voice their thoughts as before, which are valuable tools for tackling complex problems.

Design thinking is proven to strengthen women’s interest in tech, and media production can also be a method used [8]. Activities related to media production could, for example, be storytelling or creating a slideshow. Further, the interest in programming games was almost equal between boys and girls, but girls tend to lose their interest after
succeeding with the first easy steps [47]. An analysis of children’s Scratch programs showed that the girls’ programs were not among the most advanced ones. Instead, they were often satisfied with easier programs [20]. It was also found that games that girls have constructed differed in several ways compared to games created by boys [15]. Among others, the girls’ games were more reality-based, included less violence, and the story could be about fear or social issues. There were also several ways to win the game, and often, everyone could end up a winner.

2.11 Technology acceptance model

The present study undertakes the lens of the TAM to examine how well the developed tool is performing according to the users. An important aspect of technology acceptance that TAM considers is that it emphasizes the importance that users need to perceive a given technology product as user-friendly and useful for it to be accepted. “The degree to which a person believes that using a particular system would enhance their job performance” [14] is the definition of the first essential determinant, perceived usefulness. Perceived ease of use is the second important determinant that is defined as “the degree to which a person believes that using a particular system would be free from effort” [14].

3 METHOD

Firstly, this section presents the tested product and its development progress. Secondly, it delivers the methods for data collection and analysis.

3.1 Research framing

Storykit is a Swedish company offering other businesses a web-based digital tool for video creation. The ambition for the created videos is to convey a story to be published on different social media platforms for marketing purposes. Storykit’s tool consists of several products, and this research uses the product that creates a video with the help of AI, called Script editor. Userpilot is an external product used in the tool to take new users through simple and quick guidance.

3.2 The product

StoryCode is a step-by-step programming learning tool built together with Storykit’s Script editor (see Fig 8 in Appendix A.1). StoryCode was created during this thesis. Userpilot was used to create the tutorial that guided the users through the programming tasks. The instructions from Userpilot were in Swedish. The tested product consisted of six programming tasks and also some tasks connected to creating a video (see Appendix A.1). Inspiration for the user experience was taken from DojaCode (see Introduction) to show the students the creative side of programming. The product covers three different programming languages; CSS, HTML, and Javascript; to give the user an introduction to the programming languages and how they are connected. The product’s first task was to use CSS to change the color of the integrated development environment (IDE). The tasks were described at the top of the IDE in English. When the user started typing a color, color options were shown in a list to make it user-friendly. After completing the first task, the user went over to the next one using a continue button. The next one consisted of creating the text for the video in an HTML setting. The subject of the text could be anything. When saving the text, it was updated in the Script editor simultaneously. The third task was to add the new line tag in the text made in the previous task. That would divide the text into different sections. For the fourth task, Javascript was used, and the user wrote a variable for a pre-made for-loop. The static for-loop looked like it was looping through the slides that the video consisted of, and inside the for-loop, the user would change the variable to true, which was the fifth task. The sixth task was to enable document styling in the Script editor by setting the enableVideoStyling-variable to true. If the input from the user was incorrect, an error message appeared encouraging the user to try again along with a hint as to what was wrong. After the six
tasks, the next stage of the tutorial was to produce a storyboard out of the text made in the previous stage. The user was guided through the Script editor with the Userpilot-tool and was ending up generating a finished storyboard, which after rendering becomes a finished video.

3.3 Developing the product

When making the first design iteration, DojaCode and the company’s design interface were used for inspiration in design decisions as well as the design for an IDE and a console. Other digital learning tools were checked to learn which difficulty level suited the product’s programming tasks. One of them, Imagilabs, introduced structures such as for-loops and variables in Python for girls from 8 to 14 years old [25] and therefore it was decided to include that in the tasks. The curriculum in technology for high school students was reviewed to understand the knowledge students get from school. It was stated that block programming was used and more uncomplicated programming tasks with microbits [44]. When using microbits, the user was exposed to structures and words such as loops, arrays, and variables.

Based on that, students should be familiar with those words. The difficulty level of the tasks was set so that they were facing new programming languages and new code syntax but still recognized some parts like for-loops. The tasks were also formed to give visual feedback on what happened when they performed the task. Additionally, they were formed in such a way that they could connect the specific language to the specific kind of task. For example, changing the color is connected to design and CSS. In addition, the importance of the student feeling successful while doing the tasks was taken into account [17]. Therefore the first task was designed to be simple. For all the tasks, StoryCode included the skeleton of the code syntax, and the task was to change some words, like a variable, and not write the whole program since that would have been too difficult.

The focus of the product was to create a good user experience and get a smooth user flow. It was necessary since the target group was unfamiliar with both StoryCode and the programming languages. The first design was made using Figma (see Fig 3). After comparing the experience when using DojaCode for the first time and the first product design, it was apparent that the design was not user-friendly enough. It needed to be more interactive and distinct and have a better user interface. Therefore, efforts were made to make it look more like DojaCode during the second iteration, both visually and concerning the user flow (see Fig 4).

After the pre-study (see 3.5.2), a third iteration was done. A code issue was fixed and the number of instructions from UserPilot was reduced since it confused the pre-study users on what instructions to follow.

After the third iteration, the user tests (see 3.5.3) were done and several changes were made based on the feedback from the participants. These changes were made in consultation with a UX designer at Storykit. The main switch was to change the language in the tool to Swedish since many thought English made it harder to understand. The user flow was changed so that Userpilot instructions would not be necessary and could be removed. It was done by adding a pre-task popup with information about the coming task and an after-task popup with feedback about succeeding in the task. An information button close to the continue button was added as well. In the header, information regarding which task they were doing and how many were left to do, was added to reduce potential stress when using the tool. Also,
when trying to interact outside the intended area, a feedback message was sent that it was not possible. Other than that, several design-related changes were made such as having the same design on the call-to-action elements and the color was changed on the same. The contrast in text and components overall in the product was increased to become easier to see and the text was made bigger for the same reason.

Figure 3: The initial design made in Figma.  

Figure 4: The product design after iteration three.

3.4 Technical tools

The prototype was developed using Typescript together with the Angular framework. The design was developed for a computer interface and Google Chrome since the company’s product was optimized for those. The product tested was built upon Storykit’s Script editor. The tool creates a storyboard based on the text put in the interface. The user could choose its background images or videos themselves from different internet assets libraries such as Videoblocks, Pexels, and Unsplash.

3.5 Evaluation

The study used a mixed-matched embedded design approach, that is when quantitative data and qualitative data are collected simultaneously, but one of the designs is seen as the primary design. The other one is used to strengthen arguments found in the primary source [22]. The approach was used to get a more diverse set of data representations. The data was collected to evaluate how the students experienced a more creative programming learning tool like StoryCode. The qualitative data was collected via four focus groups with three or four students in each group recruited from a junior high school class in Sweden in the Spring of 2022. The purpose was to gain more insight into the students’ thoughts about the product. Further, a survey was conducted from the same students on the same subject to get additional quantified data regarding the usage of StoryCode. During the case study, field notes were taken from observations in the classroom.

3.5.1 Participants. Fourteen students took part in the user test and all students were in eighth grade. Six students participated in trying out the tool but were not part of the user testing. Their teacher helped to recruit them since s/he expressed an interest in the study. The teacher also helped with collecting informed consent from their parents. The study participants did not have any particular background or knowledge about programming. The number of male participants was nine, female participants were three, and others were two.

3.5.2 Pre-study. A pre-study was conducted to test the product’s interactions. Two participants took part in the study. A participant from Storykit who works as a full-stack developer and a UX designer who was not familiar with Storykit’s tools from the beginning. They did not get any instructions beforehand but were asked to give feedback in the end. The
challenges when using the tool were noticed and taken under consideration for further product improvement. First of all, an issue was detected since the result of task two did not give the feedback as intended. Secondly, the participants got confused when there were too many instructions from UserPilot, which made them not read the instructions written in the tool. Thus, several instructions were deleted in the programming stage of the tool but were kept in the video making-stage since there were no other instructions there. The only pop-ups from UserPilot during the programming stage were a welcome text, explanations on what kind of text they should write, and what happened with the program when the code was written the way it was.

3.5.3 User test. The user tests were performed during the workshop which was divided into two sessions of 50 minutes each. The first session started with an inspirational presentation about the possibilities of programming, challenges within the tech community such as gender stereotypes, and common challenges for programming beginners. Storykit’s video tool was also presented and an example video of what can be done with the video tool was shown as inspiration. The users were then placed at their seats and were asked to start using StoryCode on their computers. Notes were taken during this part of the workshop if something did not go as planned. If they had any questions, they could ask them during the tutorial. The students worked on their videos until the first session ended. The second session took place the next day and started with some more work with StoryCode. After that, some students started to fill out the survey (see Appendix A.2) with questions about previous programming knowledge, the product, and programming in general. Four groups of three to four students were gathered to discuss semi-structured interview questions further in separate focus groups, following the embedded design. The discussion was recorded. When all questions were asked, the students’ switched so that the ones that had a discussion did the survey and vice versa.

3.6 Delimitations

The product was developed to engage children in learning programming and creatively try programming. The tool was developed with English instructions. The study targeted students in a Swedish junior high school and the participants were in 8th grade. The workshop was divided into two shorter sessions because of the students’ curriculum restrictions and the time was limited as well due to the same reasons. The thesis also focused on the Swedish school and is using the TAM for structuring the discussion. Moreover, the report mainly focused on programming as a subject and not the digitalization transformation as a whole.

3.7 Methods for data analysis

Two methods were used to analyze the collected data, one for the quantitative and one for the qualitative data. Statistical analysis was practiced for the quantitative data from the survey. For the qualitative data, an inductive approach was taken and thematic analysis was used. The interviews were audio-recorded so the beginning of the data analysis began by transcribing recordings. The results of the transcribed interview data was then analyzed and divided into the themes of perceived ease of use or perceived usefulness. The themes were based on the TAM.

3.8 Ethics

The participants in this study had to have a consent form signed by their parents before participating in the study. The study data was then anonymized and can not be connected to any participant. The study also followed the general guidelines of the Swedish Ethical Review Authority [3].

4 RESULTS

First, the results from the survey are presented and second, the workshop results are given.
4.1 Survey

Based on the results from the survey, the participants were more familiar with using a computer than not familiar with how to use a computer since the mean value was 3.9 on a Likert scale. Furthermore, two out of 14 participants had the technical subject as one of their favorite subjects. Regarding their general opinion of programming, five participants mentioned in the free text question that it was fun, one said it was okay, three participants thought it was boring, and four did not think programming was their thing. Three also said that it was difficult. The knowledge in programming all participants at least had beforehand was block programming in Scratch, Python programming tasks from a math book, and programming with microbits.

Regarding the ratings about the tool, the participants did not think it was either boring or fun, with a mean value of 3.1. The mean value for how inspiring the tool was was lower, 2.9, even though the median was 3. The most popular quality was creativity, with a rating of 3.4 as the mean value and 3.5 as the median. The tool was neither educational nor the opposite, with a mean value of 3. The difference in mean value between participants that have been programming outside school and those who have not was 0.4. The students who had programmed before rated it as 3.3 in mean value in the educational category, and the rest rated 2.9. According to the participants, when it came to user-friendliness, the tool lacked that since the mean value was 2.4.

Storyboard got a mean value of 2.4 regarding how clear the instructions were, where one represented very unclear. Nine participants expressed that the tool was more straightforward the second time they used it. The rest did not agree with that as they replied that it was not easier the second time or that they did not understand the question.

![How many participants that did or did not understand the task](image)

**Figure 5:** The participants’ answers on whether they understood the task or not (see Appendix A.1).

When asked if they understood what to do on the different tasks (see Fig 5), one participant said that the four last tasks were not memorable, and two participants said they did not have enough time to do the three last tasks. The two participants that did not do the three last tasks rated the understanding of the tool’s English as two or three, where one equaled very difficult and five very easy. The three last tasks consisted of Javascript problems, the first one was CSS,
and the second and third were HTML. One person answered that no task was understandable, and the answer was copied and pasted for the last five tasks. The easiest task to understand was task two, where they were supposed to write a manuscript for a video. Task two was also the task most participants spent the most time on. The most challenging task was even among the rest of the tasks, but the percentage of participants saying that they understood (among the yes or no answers) was higher for the first task than for the three last ones. The last task had the lowest percentage of participants understanding it. The third task was the least memorable, whereas everyone remembered the second task.

Regarding whether they understood the English in the tool, a majority rated their understanding above the average, three, as the mean value was 3.8. When comparing the answers of participants who rated their understanding of the tool’s English below four with participants who rated four or above, the understanding was higher for participants rating above four for all tasks except one (see Fig 6).

When the participants rated their knowledge in programming, eight participants rated their knowledge as two and the mean value was three (see Fig 7). One equaled that they thought programming was very easy, and five equaled that it was very hard. Compared to earlier rating questions, this question was the other way around, as the previous questions had a positive rating of five and a negative rating of one. One participant rated three, two participants rated four, and three participants rated five, that programming is very difficult for them.

When answering the general question about what the participants thought about the tool, the most common answer (11 participants) was that it was difficult to understand or did not have clear instructions. Seven participants wrote that their perception of the tool was either that it was good, fun, or interesting. Further, two participants thought it was okay, whereas one person answered that the tool was boring but that something new was learned. Two participants answered that it was not user-friendly as it was too difficult to do easy interactions such as going back or saving progress. One person thought it was creative, and another thought it was too simple.

The differences that could be seen depending on gender were what prior knowledge the participants had beforehand. Two males had learned to program in their spare time. Further, the understanding of the tasks was protruding where all female and non-binary participants answered that they did not do or did not understand task 4, 5, and 6. When the female and non-binary participants were asked to rate the tool’s characteristics, they rated it higher than the mean value.
in all characteristics except for how easy it was to use. Then they rated the tool lower. None of them were interested in programming from the beginning, but three out of four thought it was fun or educational to try the tool.

4.2 Workshop

4.2.1 Observations. Several observations were made during the user test session. First of all, it appeared that all students could not understand the written instructions according to the teacher. Also, it became clear that the students needed to log in to the tool, which delayed the testing. Further, Userpilot did not seem to work as expected as it was not triggered on the students’ computers the first time. Therefore, the settings for the flow were set to be triggered every time someone entered the page, which caused triggers in the middle of the user test session, and the flow restarted. It was also noticed that several students did not read the instructions. Instead, they asked right away about the next step. The same happened if the tool did not work as expected or if they got an error message and were asked to try again. However, the participants often completed the task by themselves after being asked to read the instructions. Additionally, they did not follow the instruction to create a text with at least 15 sentences, which resulted in videos that were only one sentence long. The concentration in the classroom could have been better, and some of them collaborated with the person next to them during the user testing. Several groups were also in the same classroom and could hear each other’s arguments when filling out the form. One group did not hear the other’s arguments before the focus group discussion.

4.2.2 StoryCode. The tasks took varied amounts of time and the students left their creations after the workshop at different stages and complexity levels. When they were done with the programming tasks in StoryCode, stage one was done. Stage two was done after they had added images to their video. The number of students that finished the first stage but did not add images was eight. Six participants created a finished manuscript with text and images and two of them also created a finished manuscript but the text was around one sentence long, which made the video short. Some students did not write their text as the instructions were. Instead, they got their text from song lyrics, The Holy Bible, children’s stories, Wikipedia, and similar.

4.2.3 Focus group. The first question asked in the focus groups was “What did you think about the tool?” and the answers to that varied. One person said, “I think it was good, pretty easy to understand, and easy to work with. But when my computer died, everything disappeared, so I did not know how to save my text. Otherwise, it worked well.” In another group, one person thought that the English were a bit complicated, but it was easier to understand the second time they used the tool. The other person said it was difficult initially but that the tool became easier as learnings were made. It was hard to do easy interactions like going back, however. In the third group, one person said that it was pretty fun. Another said that it was difficult to understand the first time using the tool because of no understanding of how to move forward after task two due to inadequate instructions. That it was in English made it more challenging, said another person. The last group was on the same path; it was difficult to understand. Another participant thought it was understandable, but more thinking was needed to understand. They said that the English were okay, and one person thought whether it was in English or Swedish would not matter; The tool would have been difficult either way.

When asked what was most difficult about the tool, one person answered that creating a video was the most difficult since there were no instructions on what to do after doing the programming tasks. The answer implies that the Userpilot tool did not work in this case since there should have been instructions. In the second group, one person answered that the most challenging part was understanding what to do and when so that nothing went wrong with no possibility of going back and redoing the task. Difficulties with understanding English were also mentioned since it caused one participant not to understand the task. Lastly, it was mentioned that it was not any particular task or step that made it difficult. Instead, it was due to a lack of previous programming experience that made it challenging. The third group
agreed that unclear instructions were the hardest, and in the fourth group, a participant said that it was difficult to understand how to create a video in the end and that more time was needed.

The next question discussed what the easiest part of the tool was. The first group thought that adding pictures was done effortlessly. In the second group, writing the manuscript, dividing the text into paragraphs, and adding images were mentioned as the easiest part. The javascript instructions were the difficult ones. A person in the next group said that it was needed to click everywhere in the tool and finally find the boxes to write in, so it was not easy in the beginning until they were found. The last group also thought that the most uncomplicated step was to write the text and that it was fun to do it.

On the question regarding how they would like to learn programming, a participant in the first group answered creating games. The person thought that their tasks so far had not been as fun. On the follow-up question, if less exciting tasks have decreased their programming interest, the whole group answered yes. In the other group, a person mentioned Scratch, which made another person say there was a big gap between Scratch and StoryCode in difficulty level. In the third group discussion, Scratch was mentioned as well as creating movies and watching videos about programming. Scratch was mentioned because it was more fun than StoryCode and accessible because of block programming. The programming they had done in math classes were tasks from a book that was not that fun, even though some tasks were more fun than others. Nevertheless, some tasks felt meaningless. For example, they were calculating three times five, which could equally well be calculated with a calculator, in the participant’s opinion. One of the participants had done a website one time which was more fun because something was created from it. The last group also liked Scratch because it was easy. The instructions were clear and explained what the programming block does and why. In the technical subject, they used microbits, which also used block programming, which they thought was fun but a bit difficult. One person thought the programming book in their math class was okay and not that hard.

Regarding how this tool was compared to what they had done earlier in school, a participant in the first group expressed that even though it was difficult sometimes, it was more fun than reading what to do from a book and then getting the correct answer. The person also said that it was positive that they got to learn on their own by trying out different solutions until they got the correct answer. That way of working was more educational. All participants in the group said that the tool was more difficult than other programming tasks that they had done but still more fun. The tricky part was the code syntax since they had not seen it before. One person said that it was not necessarily harder but built in a new way, with different steps, than they are used to. In the second group, one participant thought that the tool was more complicated than what they had done since they either only had to drag and drop blocks or had clear instructions on what to do before. In StoryCode, the instructions were unclear, making it harder to understand where to click or which letters to write. The reactions in the third group were more of a mix. One person thought it was fun since it was not math and something was created. Another person thought it was more complicated than what they have done before, and another thought it was easier. One participant said that microbits were more fun than StoryCode and the last person said that microbits were more fun because one got to create things. In StoryCode, the participant thought they did not understand what was created. The last group said that the difficulty level on StoryCode was high compared to what they had done earlier.

The first group expressed that the saving function could be improved to work better in StoryCode since the text was gone when the computer died. Some things could also be clarified since two persons had difficulties knowing where to put the code in the tool, and for beginners, this tool would be challenging to use, according to the participants. In the second group, one person wanted the instructions to be in Swedish, and on the follow-up question, if the group saw any Swedish instructions, two of them did not, and the last one did not remember. They concluded StoryCode itself did not need improvements; they were just unfamiliar with programming in general. A participant in the third group said that clear instructions would be needed. Another one explained that explanations on why tasks are done in the way they
are would have been good to use the knowledge in later similar tasks or challenges. The later tasks were more about finding the code to put somewhere and then finding the box it should be in, and it would not have been possible to use that problem-solving method in another problem. In the last group, one person thought that the tool should be more straightforward on what to do and another person would have liked it to have Swedish instructions. One person thought that some things could have been deleted, for example, changing the variable from false to true since it could have been true from the beginning instead. When continuing, the person said that an explanation of why the variable was changed would have been good.

The instructions were discussed, and some were clearer than others, said a person from the first group. Also, the English ones were more unclear. On the question of what they had learned, the first group continued by saying that everything they did was something they learned since it was new to them. One person said that it was easy to understand and learn from StoryCode. Further, they answered yes to the question if they felt that they had learned something about CSS, HTML, and Javascript. The other groups answered that they guessed they had learned something and that it was not known what was used when but they could recall seeing the names in StoryCode. The last group said they did not know what it was, but they had learned a little since it was used.

5 DISCUSSION
This section will discuss the results through the TAM and highlight what the experienced perceived usefulness is and if the design was understandable enough to experience the tool as easy to use.

5.1 Perceived usefulness
A good developer has access to a toolbox with solutions to programming problems that can be used to solve new programming problems [9]. Therefore, it would have been good if the explanations of how the code syntax worked were more noticed in StoryCode. There were pop-ups with explanations in Swedish for the three last tasks. Still, even though they existed, a participant said that explanations were wanted. Since the Swedish instructions seemed to not work for everyone or were not seen, those need to be presented in another way, not using Userpilot, to make sure the instructions are visible. The reason for it is to give the participants a chance to get a deeper understanding of why and when specific code syntax is used and can bring that way of thinking into other programming problems in the future and to their toolbox.

The tool aimed to give ‘creative freedom’, meaning being able to create different videos, while having straightforward tasks. Based on the results, the latter failed in the user tests. Since creativity was the best quality and one participant expressed appreciation towards the explorative methodology in the tool, it seems like the experience is on the right path towards the goal for StoryCode; To inspire and make code fun. Although the results also show that the tool elicited an intermediate level of engagement, positioned between boring and fun.

Based on the rating of their programming skills, the participants’ pre-knowledge of programming varied since the only value not chosen was one, which equaled that programming was very easy. Information from the teacher beforehand confirmed these results, that some students have practiced programming in their free time, and therefore have found programming more uncomplicated, while others are entirely uninterested. This variation made it hard to decide the difficulty level of StoryCode to fit everyone so that they perceived the tool as useful. The result does not completely align with the learning goals of the National Agency for Education and the expected knowledge was higher than the results express. Therefore, it might have been good to create a tool for a lower knowledge level than expected to fulfill the aim of the tool, easy and useful.

According to the teaching model (see Background 2.8), a block-based tool was the recommended tool for developing most skills for the student’s education level. StoryCode could be categorized as a combination of a game platform and
a light version of a code editing environment, which are recommended tools for a higher education level. That could also explain the students’ difficulties with the tool.

The challenging part is creating a tool that levels up at the same pace as the students learn. Perhaps it went a bit too quickly in StoryCode since the understanding went from almost everyone to half of the participants between task two and task three. The time spent on each task is also a factor to consider when judging the suitability for each task. The comprehension of task two was likely affected by the large amount of time spent on that task.

The results of the student’s understanding of the tasks show that they understood the first two tasks well. However, this does not mean that they understood the connection to CSS or HTML or why it worked the way it did. The results indicate the opposite: Several students did not recognize the names of the programming languages, and they wanted to receive more info about how the code worked to understand the code. Therefore, additional information about the programming languages was added during the fourth iteration.

Furthermore, the results showed that the last four tasks were more of a challenge, and those tasks also included more complicated elements - something the National Agency for Education says should be introduced later on in education [7]. One group mentioned that everything was new for them in the group discussion, which could mean that even the expected pre-knowledge might not have existed. It should also be considered that Scratch was something they did two years ago, making it possible not to remember what was learned then. Based on this, the difficulty level might have been too difficult in combination with the tools’ other weaknesses. Programming should also be taught at a slow pace with much support otherwise, the students could give up [7, 42]. Signs of the students giving up were seen during the workshop. Since giving much support from teachers was not wholly possible due to a lack of resources during the workshop, the support from the tool was important. The results show that the support was not enough and since almost everyone understood the first tasks and not the last ones, a slower teaching pace, or more support, would have been needed.

Based on the results, it is clear that some kind of change is needed in school since many participants wished for more fun programming activities like creating games or easier tasks like using Scratch. Scratch has been used with positive results when learning programming digitally [29] and media production is another possible way of learning programming [8] and since StoryCode produces a video in the end, it can be categorized as media production. When talking about how the participants want to learn to program in school, the word fun is recurrent. The participants want to use Scratch because it is fun and easy, and they want to use microbits since it also is fun. So, fun is essential, but the perception of fun is different for everyone. Fun could be connected to the ease of understanding since when it was hard, some students quickly started to do other things during the workshop, and it was not fun anymore. That also agrees with the statement that the feeling of success is essential at the beginning of learning programming [17]. Since many participants did not understand the last tasks, which potentially could have been affecting the fun, and since the mean rating between fun and boring was 3, StoryCode needs to improve the feeling of success throughout the tool.

The perception of StoryCode is for sure affected by the challenges the participants had with the user experience. Even though most students said that it was hard to understand what to do, half of the participants commented that the tool was fun to use in the survey. The other qualities of the tool got rated around three, which could have resulted in higher ratings after implementing the suggested improvements in iteration four.

When developing the tool, inspiration was taken from DojaCode, and attempts were made to get a similar feeling when using the tool regarding user experience. In DojaCode, it was possible to test different alternatives on each task and get clear visual feedback directly. A similar task was made for the first task in StoryCode, but the visual feedback came after clicking the continue button for the rest of the tasks. The user experience in DojaCode is clear, and the tool is focused on visual feedback more than on explaining why the code worked for a specific problem. That approach
seems good if the aim is to inspire and explore different options, but those tasks might have been too easy for the target group since there was no problem-solving.

None of the female or non-binary participants had any previous knowledge, which aligns with the example from Carnegie Mellon University [21]. Even though it was not possible to take the same actions as in the study to make these participants feel more comfortable, the tool got better ratings from them. They perceived the tool as slightly more useful than the others. It seems that they valued the creative side of the tool since they still gave better ratings, even though they thought it was hard to understand and difficult. By reducing the difficulties, the tool would probably have received even better feedback and been accepted by its users according to the TAM.

5.2 Perceived ease of use

The observations from the user test workshop show that written instructions were a challenge for some students. It might have affected the result since it was extra hard for those students to understand the tasks that were only explained in the text.

As could be seen in Figure 5, the first two tasks were the easiest ones for the students to understand. For the second task, it may be because more time was spent on it since it was the most time-consuming. The last tasks were the most difficult ones to understand, and they were also the ones that contained writing code syntax. Since the programming languages were new, this could explain a lower understanding of those tasks. Also, since the instructions were hard to understand, that probably worsened the probability of understanding the task. The least memorable task was the third one, which might be because it looked similar to the second one and became mixed up.

Interestingly, five out of six tasks were rated more frequently as understandable if the English knowledge was also rated higher. That indicates that English was critical knowledge to go through the tool properly. It motivates changing to Swedish instructions. Even though English is the language used among programmers, the results show that it is a barrier for the students to understand the tool. Since the aim is not to improve their English skills but rather to give them insight into programming, the change of language is again motivated. The instructions rated 2.43 regarding how clear they were, favoring the previous statement. Also, the need for better instructions was mentioned several times in the focus groups. However, it is worth mentioning that there could be two different types of unclear instructions. It is either that it is difficult to understand English and therefore it is hard to understand (which is confirmed by the results) or that the instructions given were not enough for them to understand, which the results also indicate to be accurate. That is because participants who rated their understanding of English in the tool above three still had challenges understanding the tasks.

Furthermore, based on observations in the classroom, some participants did not read the instructions in the tool. As mentioned, they were very fast to raise their hand as soon as they met resistance, but when told to read the instructions, they often managed to complete the task by themselves. This behavior was recurrent in other situations, such as when they got a pop-up modal with feedback on what they did wrong or if the tool did not work as expected. The cause of this behavior might be due to how they are used to working with programming. In the focus group, it was mentioned that StoryCode forced them to work differently since they needed to try out different solutions until they were correct. In other words, a more explorative approach than what they are used to, as mentioned before.

The described actions, giving up when struggling without trying themselves, have similar traits to the behavior of a person with a fixed mindset. A person with a fixed mindset is someone that gives up when it is shown they cannot do something [12]. Also, a fixed mindset person is likely to either give up, avoid what they see as an impassable barrier, or show a helpless response when facing negative feedback [36], which is similar to what happened during the study when meeting resistance. Further, it is stated that “learning to program may foster a fixed mindset due to the very high number of potential error points” [13]. A growth mindset is to be preferred since it has been shown that a compelling
factor for learning is the mindset regarding one’s ability level [18]. Based on these facts, it is suggested that for a student to have a growth mindset, one needs to have strategies for getting unstuck, for example having the toolbox with solutions that was mentioned earlier [12]. It is therefore desirable that StoryCode should have offered strategies to get unstuck to avoid the fixed mindset behavior.

The act of students getting stuck implies that the user experience was not optimal, which the results agree with, and the instructions should have been more highlighted than they were. At some points, there were two instructions in different languages at the same time. One English instruction in the tool and one Swedish instruction as a pop-up. The pop-up disappeared when clicking on the correct object or button. There are several possibilities for confusion when giving information in two different ways. The pop-up is a call-to-action, which the English instructions did not have. That is why there was only one instruction at a time in the fourth iteration of the design process. Also, when the user clicked the cross on the pop-up, the flow with pop-ups was stopped. It is difficult to know how many of the participants did that, but it could explain why some students did not see the Swedish instructions. Some students experienced that the flow restarted in the middle of the tool and welcomed them again. That is also a source of confusion and another reason for deleting the Userpilot flow in the programming stage of the tool and replacing the pop-up messages with an info button.

It is a fact that the user flow through the tool needed to be improved for the fourth iteration since the user flow for the participants was not the same. Some of them tried to go back and change the text since the text was always visible in the Script editor. The Script editor was not clickable, and when there was no other way to go back, the user-friendliness was affected. Therefore, the interface was changed so that Storykit’s Script editor appeared after they were done with the programming part. In this way, the number of user flows was narrowed down, and decreased the possibility of doing wrong, which one participant feared when interacting with the tool. For the fourth iteration, task two was no longer writing an optional text. Instead, it was a pre-written text, and the task was to personalize the text by changing the names and other personal attributes. It was made so that the users would not get stuck for too long on one task since that made the participants lose focus. Also, not everyone wrote several sentences, making the created video very short, which might have been less rewarding for the participants. Furthermore, the call-to-action buttons need to be more evident. That is because when a participant asked what to do next, the answer was to click the continue button, which should be obvious from the beginning. Because of this, one unnecessary button was deleted after task three, and, as mentioned, the Userpilot instructions were deleted. The participants’ familiarity with computers could have had an impact on the user experience, but the results indicate not since participants who rated themselves as very familiar with computers also rated the tool as challenging to use.

When the participants rated themselves in programming, it showed that the mean value was three. There are some uncertainties with the result since many of the participants only had previous knowledge in block programming and Python tasks in math but rated their knowledge the same as a participant who worked as a programmer teacher, for example. It might be because the more one knows in programming, the more one discovers how much there is to know. Also, this question had its rating the other way around, which might have resulted in wrong answers, depending on if the students perceived this difference.

5.3 Method discussion

First, having the user tests split between two classes was not optimal. During the first session, they got approximately 15 minutes to explore the tool, and since task two took some time for the students, most of them only got to that task during the first session. Therefore, it was decided to let them work with the tool again during the second lesson, so they managed to get through all the programming tasks. It affected the results since they had tested the tool twice when answering questions about the tool, and the students answered that it was easier the second time they used it. Even
though they used the tool twice, it was not enough time to go through the tool correctly for everyone. There was little time left for them to think about what they did, and several students missed the instructions in the tool, which could have been caused by the unconcentrated atmosphere in the classroom and the short amount of time. An improvement would be to do the user test with fewer students in a quiet environment to ensure they are done with everything and have enough time to answer the questions afterwards. Several students did not remember the last tasks, which could have been improved with the suggested setting. That would also have been beneficial since it would have been possible to see if the Userpilot tool worked or not for everyone. It is known from observations that it worked for some students, but some students did not notice the instructions according to the results. No correlation can be made between those answers since it is not known whom it worked for or not. It might have affected the perception of the tool since it could have caused them not to understand what to do properly. A pre-study with a person in the target group would have been beneficial to avoid misunderstandings and to understand their programming and English level at that age. Also, when rating the tool in the survey, the tool was not defined, and the participants probably had both StoryCode and the Script editor in mind even though the programming stage should have been in focus.

Regarding the evaluation methods, the focus groups, combined with the survey, provided valuable insights into the tool. The ability to pose follow-up questions and obtain in-depth answers yielded valuable qualitative data. The survey results offered a better overview of the tool’s perception and may have also reflected a more honest outcome due to participant anonymity. However, it is important to note that a larger participant group would have increased result certainty. Furthermore, there is a potential for bias, attributed to the phrasing of the survey questions.

Due to the workshop layout and time constraints, the order of completing the survey and participating in the focus group varied among participants. This difference may have introduced bias, as some participants heard the opinions of others before completing the survey.

5.4 Future research and contributions

The subject, of programming education, is complex because many parts must work together for a good result. More research is therefore needed in several areas as programming in schools is relatively new in Sweden. StoryCode has contributed by exploring a creative alternative to programming education and the results made are necessary to take into account for future digital tools. Furthermore, it would be interesting to do even more research on how creativity can contribute to motivation for the programming subject. It would also be interesting to explore how digital tools can solve the problems with students who have different prior knowledge and that there are often few teachers during lessons. Especially when a lot of support in the beginning has proven to be crucial for learning.

6 CONCLUSION

In this thesis, a new learning programming tool to reduce the gender gap in STEM education was developed and tested. The aim was to fulfill the TAM suggestions of a product with a great user experience. A product that is easy to use and is seen as useful within programming education. An education that needs to suit different types of persons, especially women whose appearance is lacking in computer science.

The results show that the tool was too difficult from several perspectives, and more support should have been added making it possible for the students to get unstuck. Other improvements needed to be done are to increase the feeling of success when using the tool as well as the suggested design improvements for iteration four. However, the results suggest some positive qualities since some participants appreciated the new way of learning programming and expressed it was fun to use, but difficulties in using the tool affected the perceived usefulness and the perceived ease of use.

When there are not enough teachers in a classroom, it places high demands on digital tools. The students need to become as independent as possible and then the tools must pick up if someone needs help, they must also arouse interest
in really understanding what is being done, and work for several different levels of difficulty. The mixed opinions about the tool reflect the difficulties in meeting the requirements and the different individual needs of how to learn. At the same time, the tool must explain something as complex as programming, which has a high threshold at the beginning when the toolbox for solving programming problems is not filled. With this report, it is shown that creativity can be a way to lower the threshold, but the execution of the workshop and the lack of user tests from the target group beforehand lowered the possibility to evaluate the tool properly. By implementing the proposed changes, the results indicate that the tool would be more accepted by the users.

ACKNOWLEDGMENTS
Throughout the thesis, I have been given really valuable support from my supervisor Olga Viberg at KTH, thank you for that. I would also like to acknowledge my supervisors from Storykit AB: Mona Dadoun and Lisa Häkansson for believing in me and giving me valuable support and encouragement. Finally, I want to thank my family and friends for their support throughout this year.
A APPENDIX

A.1 Figures

Figure 8: The company’s tool: Script editor.

Figure 9: Task 1 of StoryCode.
Reduce the gender gap in computer science education using creative programming

Figure 10: Task 2 of StoryCode.

Figure 11: Task 3 of StoryCode.

Figure 12: Task 4 and 5 of StoryCode.

Figure 13: Task 6 of StoryCode.
A.2 Survey questions

(1) Vilket kön identifierar du dig som?
(2) Vad vill du studera på gymnasiet?
(3) Förklara varför du vill studera det programmet på gymnasiet
(4) Vilka ämnen är dina favoritämnen i skolan?
(5) Hur van är du vid att använda en dator? Inte van - väldigt van
(6) Hur var verket? träigt - kul
(7) Hur var verket? oinspirerande - inspirerande
(8) Hur var verket? okreativt - kreativt
(9) Hur var verket? ej lärorikt - lärorikt
(10) Hur var verket? svåranvänd - lättanvänd
(11) Hur tydliga var instruktionerna i verket? väldigt otydliga - väldigt tydliga
(12) Vad tyckte du om verket? Motivera!
(13) Upplevde du någon skillnad mellan första och andra gången som du använde verket?
(14) Vad var svårast att förstå, att programmera eller att skapa en video?
(15) Förstod du vad du skulle göra på denna uppgift? Motivera kort! Task 1
(16) Förstod du vad du skulle göra på denna uppgift? Motivera kort! Task 2
(17) Förstod du vad du skulle göra på denna uppgift? Motivera kort! Task 3
(18) Förstod du vad du skulle göra på denna uppgift? Motivera kort! Task 4 and 5
(19) Förstod du vad du skulle göra på denna uppgift? Motivera kort! Task 6
(20) Har din bild av programmering ändrats? Motivera varför/varför inte!
(21) Hur väl förstod du engelskan i verket? mycket svårt - mycket lätt
(22) Vad tar du med dig från denna workshop?
(23) Har du testat programmering förut? Om ja, vad gjorde du och vilka språk/verktyg användes?
(24) Vad upplever du är lättast och svårast med programmering?
(25) På vilken nivå skulle du säga att din kunskap inom programmering är? mycket enkelt - mycket svårt
(26) Vad tycker du om programmering i allmänhet?
Women are lacking in Science, Engineering, Technology, and Mathematics (STEM). One of the proven solutions to reduce this gap is to offer extra support to the students needing it. Further, creativity has been shown to give positive effects on females’ participation in STEM. In this thesis, a creative tool for programming education in junior high schools in Sweden was developed to give alternatives to how programming is taught and further evaluate the possible contributions it can give. The tool was tested among 14 students participating in a face-to-face workshop. To further evaluate the tool use, the study participants took part in interviews and a survey. The results show that further iterations of the tool would have been needed since the tested tool was not perceived as easy to use. Regarding perceived usefulness, the tool is on the right path since the results indicate it was fun to use but again, improvements need to be done. The contributions made are therefore an exploration of the creative alternative to programming learning in the Swedish school. Furthermore, to find out what improvements needed to be done to offer an, by users, accepted tool. Offering creative programming tools will hopefully attract a more diverse group of people to programming and decrease the gender gap within computer science.

"Keywords[swe]": människa-dator interaktion, kvinnor i tech, programmering, kreativ, videoskapande, datavetenskapauktion, grundskola