Universiteit Leiden
The Netherlands

Opleiding Informatica

The Role of Instruction Method in an Educational Game

Aník van Deursen

Supervisors:
Anna van der Meulen & Giulio Barbero

BACHELOR THESIS

Leiden Institute of Advanced Computer Science (LIACS)
www.liacs.leidenuniv.nl

21/07/2024

Abstract

Since programming is required in many fields, it is important to teach elementary education students programming concepts. Because programming concepts can be difficult to grasp for novices, educational games might make it easier to learn. Instruction method is very important in the success of educational games. This study investigated the role of instruction method on the interest in computer science and the understanding of computer science concepts of primary education students in an educational game with animals. This was done by developing two versions of an educational game, which differed in instruction method: both versions contained instructional support and worked examples, however one of the version with puzzles also had a different support tactic: erroneous worked examples (in the form of programming puzzles where buggy code had to be solved). To measure the difference between the two versions of the game, three questionnaires were taken right after the students that participated in the study played the game: a previous programming experience questionnaire, a Technology Acceptance Model questionnaire and a questionnaire that measured students understanding of the programming concepts. No significant difference was found in the interest in computer science and the understanding of computer science concepts of students who played different versions of the game. Overall, the game was successful in teaching students about variables, and the students were somewhat interested in computer science after playing the game. However, there was no difference in interest and understanding of the students based on instruction method.

Contents

1 Introduction

Primary school programming education is an important topic and should be as efficient and suitable as possible. Right now, digital literacy (which includes programming) is obligatory for primary schools in the Netherlands [LK24], which makes it extra important to have a good teaching method. Previous research has shown that humans have a limited amount of processing capacity [ZLP18] [SMP98]. Instructional support is needed to ensure that not too much information has to be processed at once [ZLP18]. There are multiple forms of instructional support. Instructional text explains how problems should be solved through text [ZLP18]. Worked examples show students how to solve a problem by presenting the solution to a problem. Erroneous worked examples are worked examples with errors that the students have to try to solve [PRS10] [ZLP18].

Programming concepts are regarded as difficult to learn which reduces the motivation and interest of students in programming [PH11] [GX18]. Educational games can help to overcome these difficulties. An educational game is a technology or video game with more objectives than entertainment [PMN14]. Previous research has shown that educational games have many advantages in primary education [Man20]. Instructional support is an important consideration in educational games. While much research has been done about instructional support in educational games in general, not much research has been done about the role of instruction method on the interest in computer science and the understanding of programming concepts of primary education students in an educational game.

1.1 Rationale & Research question

Computer programming is considered to be an useful skill to have in many fields. Learning programming concepts at a young age could come in very handy later in live for students. Since educational games could be a good way to teach student programming concepts, it is important to have a fitting and efficient instruction method for the game. This instructional method needs to be investigated to maximize the interest in computer science and the understanding of computer science concepts of primary education students.

The research aim of this study is to investigate what role different instruction methods have on the interest in computer science and the understanding of programming concepts of primary school students to make learning more efficient. This is done by developing two versions of an educational game with animals with different instruction methods and testing the two versions to see the differences. In one of the versions the students only get support in the form of instructional text and examples while in the other version students also have erroneous support: the players have to solve buggy code in programming puzzles.

The research question is as follows:
RQ What role does the instruction method have on the interest in computer science and the understanding of computer science concepts of primary education students in an educational game with animals?

1.2 Contributions

The contribution of this thesis are two versions of an educational game about programming concepts for primary education students. The two versions of the game are identical, except for one thing: one version has an extra instructional support method, where the player has to solve puzzles by fixing buggy code. The other version does not have these puzzles.

1.3 Thesis overview

In section 2, an overview of the current knowledge about instructional support in educational games for primary education about programming concepts is provided. Section 3 explains how the study was performed. In section 4, the results of the study are reported. In the last section (section 5), the results are interpreted and discussed.

This bachelor thesis was supervised by Dr. Anna van der Meulen and Giulio Barbero.

2 Related Work

2.1 Instructional support in programming education

Programming education is an important topic. Right now, digital literacy (which includes programming) is obligatory for primary schools in the Netherlands [LK24]. In 2021, a study from DUO found that 27% of primary school teachers devote time to programming, moreover the teachers in last two grades devoted the most attention to programming compared to the other grades [DUO21]. Because programming is an important skill to have and in addition obligatory in the Netherlands, primary school programming education should be as efficient and suitable as possible. The cognitive load theory is a crucial theory to consider to reach this goal.

2.1.1 Cognitive Load Theory

Previous research has shown that if the limited amount of processing capacity of humans is exceeded, cognitive overload can occur. The Cognitive Load Theory (CLT) is build upon this [SMP98] [ZLP18]. According to this theory, when instructional design is constructed with cognitive load should be kept in mind [SMP98].

When learning new information, there are multiple important considerations. An important feature of new information to consider is the element interactivity. If the element interactivity is low, it is a lot easier to learn new information than when element interactivity is high, because if it is high all elements have to be understood to fully comprehend the information [PRS10]. Another important thing to keep in mind is that humans have two forms of memory working together: working memory and long-term memory. Conscious cognitive processing takes place in the working memory. The working memory can only handle a small amount of element interactivity, and that amount is usually less than what is needed to process the information [PRS10]. The working memory can thus easily become overloaded [ZLP18]. Long term memory contains schemas, which incorporate multiple elements into one element. Those schemas can be brought to the working memory. Because of this higher interactivity elements can be processed [PRS10].

There are three forms of cognitive load: intrinsic cognitive load, extraneous cognitive load and germane cognitive load. If the added load of these three forms is more than the available working memory resources, cognitive overload occurs [PRS10] [ZLP18]. Element interactivity determines how much intrinsic cognitive load is needed. Intrinsic cognitive load can only be reduced using a simpler learning task, however this might not always be possible since some information is very complex [PRS10]. Extraneous cognitive load is the load that emerges when the manner in which information is presented and the learning activities that have to be done by students are unnecessary. This is especially important when the intrinsic cognitive load is high [PRS10]. Much like extraneous cognitive load, germane cognitive load is influenced by the learning activities and the way in which information is presented. Unlike extraneous cognitive load, learning is improved by germane cognitive load [PRS10].

To avoid cognitive overflow, learning tools should be designed in a way to minimize cognitive load. This can be done by minimizing extraneous cognitive load to make more room for germane cognitive

load [ZLP18] [PRS10]. That can be achieved using through instructional support [ZLP18]. Since programming concepts are difficult to learn for novices [KP05] [SMB+13], it is extra important to have good instructional support in educational tools for students that want to learn about programming concepts.

2.1.2 Instructional support

There are multiple forms of instructional support. Instructional text explains how problems should be solved through text [ZLP18]. Intelligently-tutored problems give feedback or hints when the student makes a mistake or asks for it. Intelligently-tutored problems work better if the student has no prior knowledge about a subject [Rit07] [MvGG+14]. Worked examples show students how to solve a problem by presenting the solution to a problem, which causes less cognitive load [PRS10] [ZLP18]. Students try to solve problems by comparing the final goal of the problem to the current state of the problem [RA10]. By showing students the steps to get to the final goal, the cognitive load can be eased [AMD+12]. Worked examples are not effective when the student already has a lot of knowledge about a subject, since the examples are then redundant [CKTS01] [MvGG+14]. A problem with worked examples is that the cognitive load that is saved is not necessarily used for germane cognitive load [RA10]. The student may need more assistance for this, with for example self-explanation[AMD+12]. For computer science specifically, much research about worked examples has been done and they have been found to be extremely important in teaching student various computer science areas [PA85] [CB07] [VPL11]. Something that could be explored more is the effect of worked examples on the interest in computer science of students.

Erroneous worked examples are worked examples with errors that the students have to try to solve. This causes the student to reflect more and use more explanation than usual, and also focus more on separate steps of a solution since they have to find the error [MvGG+14] [AMD+12]. Erroneous worked examples work best when the student already has some prior knowledge about the subject [GR07]. For students with low prior knowledge the effect might be negative, since they do not know enough about the subject to immediately identify the errors [AMD+12]. This might be improved by highlighting errors [GR07]. A combination of worked examples and erroneous worked examples have been shown to be more beneficial for students that have prior knowledge compared to merely worked examples [ZLP18] [GR07]. Adams et al. [AMD+12] found that erroneous worked examples did not immediately benefit students but did later on, which suggests that erroneous examples causes a deeper learning experience. McLaren et al. [MvGG+14] found that worked examples were more efficient and causes less cognitive load compared to other supports. In the context of programming the student would have to solve bugs in code [ZLP18]. Zhi et al. [ZLP18] have found that erroneous worked examples may be more effective than worked examples and instructional text when teaching programming loops. Compared to worked examples, not much research has been done about the effect of worked examples in the context of teaching students programming concepts, and even less about what the effect of worked examples is on the interest of students of programming. This is definitely a topic that should be researched more.

2.2 Educational games

Over the years, many educational games (or serious games) have been developed. An educational game is a technology or video game with more objectives than only entertainment [PMN14]. Educational games are interactive an can enhances students interest in learning [GKV08] [CM12] and improve the effectiveness of the learning process. The students are motivated by the objective of the game and can effectively learn through the consequences of their actions and errors [Man20]. There are four main elements of digital educational games: engagement (making the player connect with the game), autonomy (the control that the player has), mastery (the repeatability of the actions of the player for full control) and progression (the reward that players get while playing the game). Progression is considered to be the main motivation of the player [CP18].

Previous researchers have reviewed the effectiveness of educational games. The first person to use the term serious games was Clark Act in 1970, who thought that they were effective for teaching and training students because they are motivating and can efficiently teach concepts of subjects [PMN14] [Abt70]. Since then, multiple sources have found contributions to measuring the effectiveness of educational games which can be divided into four perspectives: contributions from education science (play is very important in learning [Pia13]), game science (educational games increase motivation [Sta15] [JLP13] [AAA15] and games can be more effective learning tools than traditional methods [JFK10] [DJM11] [SCJ+11]), neuroscience (educational games cause greater transfer ability of skills [CSG03] [MBG14], and greater brain volume and plasticity [SK11] [KGL+13]) and information science (because data can be more easily collected in games, this data can be analysed to improve educational games) [dF18]. Another big advantage of educational games compared to more traditional methods of learning is that because students are able to achieve goals and then immediately review their results during the game, their self-confidence gets increased and they gain more trust in their decision making skills [CM12]. A last advantage is that educational games can easily be adjusted to different technological formats [PMN14]. Overall, educational games are effective learning tools. In the context of programming concepts, that are regarded as difficult to learn which reduces the motivation and interest of students in programming [PH11] [GX18], educational games can be a way to overcome these difficulties. Previous research has shown that educational games are able to teach programming concepts to students [LK15], and that when learning about computer memory educational games can promote motivation and knowledge [Pap09] [ZLP18].

There are multiple elements that contribute to making an educational game successful. The most important element is motivation. Motivation can be attributed to the narrative context of the game [Dic05] [Dic06] [Fis05] [War04], or with the rewards and goals in the game and the motivation that players get simply from playing the game [ANVA99] [DJ05] [Jen01]. Intrinsic and extrinsic motivation are an important part of motivation. Intrinsic motivation is the internal desire to learn and extrinsic motivation is could be a goal or a reward [Don07]. Another important element to making an educational game successful is that the game must have a dynamic setting. The challenges should be placed within acceptable limits. If a challenge is above the students abilities it could bring up negative feelings. If a challenge is too easy it would probably bore the student [Man20] [Don07]. Educational games can be used to teach students a wide number of subjects [Don07]. The idea is that the student will have more interest in the subject because of the pleasure

and knowledge gained during the game. Because of this, it is important to balance learning with the game play [PMN14]. Lastly, the role of a tutor is crucial. For an educational game to be successful, the student has to maintain motivation and interest, which must be done by adapting the learning experience to each students needs [KRMSA11]. Tutoring can also monitor the behavior of the student and thus avoid inappropriate behaviour [PMN14].

2.2.1 Technology acceptance model

To give insight into motivation and user experience the Technology Acceptance model (TAM) is applied. The Technology Acceptance model contains three variables: perceived usefulness (PU), perceived ease of use (PEU) and the attitude towards using the technology (AT) [Dav86] [MG14] [YvdM22], that predict whether the technology will be accepted. The relation of these variables is shown in figure 1. The perceived usefulness of a system is the degree to which people believe that the system will have an impact on their learning. The perceived ease of use is the degree to which people believe that a system is intuitive and does not require much effort to use [Dav89].

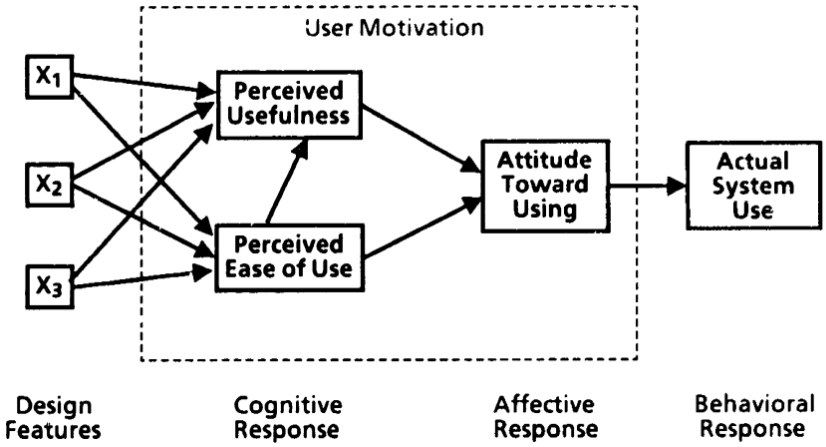

Figure 1: The Technology Acceptance Model [Dav86]

2.2.2 Primary education

For primary education specifically, educational games can improve the students learning, social interactions, behaviour, problem-solving and higher order thinking, critical ability, memory and eye-hand coordination skills. They are used to develop students cognitive skills and for increasing their motivation [Man20]. They also help students become more autonomous in their actions and decisions [ES09]. Digital education games can promote student participation and cooperation [Gro07]. They can also place the student into a flow, which would help to prevent cognitive overflow. In general, because educational games can be incorporated to support the more traditional methods of teaching, primary education is considered to be an appropriate level of education for educational games [Man20].

A lot of educational games for teaching programming concepts to primary education students have been developed, however there have been much less studies about this [GX18]. The research that

has been done about teaching programming concepts to young students through educational games has found that they have been successful [ZMB13] [KKBM12] [KS4] [KB16] [GX18].

2.2.3 Instructional support

Educational games may cause a more complex learning environment which can cause students to be overwhelmed. Because of this, instructional support is important [Wvv08] [Wv13]. Additionally, students are not able to explain themselves as well because they get a more intuitive knowledge, however it is important that they are also able to do that [Wv13]. In teaching programming, which is cognitively challenging, instructional support must be implemented [YvdM22]. While research has been done about instructional support in educational games in general, not enough research has been about instructional support in educational games for teaching and enhancing the interest of computer programming concepts of primary education students. A study that did investigate this [ZLP18] found that erroneous worked examples may be the most effective compared to worked examples and instructional text when teaching programming loops through a LOGO-like game (BOTS). However, this study did not look at the difference in interest of the students because of the instructional support. More research is required to gain a better understanding of the role of instructional support in educational games for teaching and enhancing the interest of computer programming concepts of primary education students.

3 Method

3.1 Participants

Two primary schools in the province Friesland of the Netherlands were recruited. For each school, one class was selected. School A provided 19 participants and school B 11. The students were of the ages 10 to 12. 15 of the participants are male, and 15 of the participants were female. Both classes already had previous programming lessons before the study, mainly in Scratch.

3.2 Procedure

Two versions of an educational game with animals were made (as described below in section 2.3), and three questionnaires were created to obtain an answer to the research question which were answered right after the students played the game. The first questionnaire identified the level of programming experience of the students, the second gave insight into the motivation and user experience toward the game, and the third aimed to find out how much the students learned from the game. The procedure of the research was reviewed by the Ethics Committee of the Faculty of Science. For the data collection, informed consent was obtained from both the students and their parents. The parents gave their consent through an online form and the students' assent was obtained through a physical form, and only when the parent and child both gave their consent their data was be used. In both forms the research was first explained. The children then had to write their name on the form, tick a box to give their consent and write their signature. The parents had to write down the name of their child and their own name, the date and place, and tick a box to give their consent.

The research was conducted as follows. First, a short lesson about programming and programming concepts such as variables was given. During the lesson the students were asked questions based on the previous knowledge with the information that the teachers gave beforehand, then new concepts (variables) were first introduced based on their answers and they were shown how to use this in Python. The game was also described and explained using pictures of the main screen and a puzzle (figure 2 and 3) and a video which showed how variables could be used in the game (which is described below in section 2.3) to make the students feel less overwhelmed when they first started playing the game. Afterwards, the students played the educational game for about twenty minutes. The two classes participating in the research played a different version of the game (as described below). Lastly, the students filled out the questionnaires (as described below). The research was conducted during school time and the teacher was also present. In one of the schools it lasted about an hour and fifteen minutes since there was a technical problem with the formatting of the game, and in the other school it lasted around an hour. The students were able to ask questions to the researcher when they were stuck in the game, and could also consult their fellow classmates.

3.3 Game

To investigate the role of instruction method on the interest in and understanding of computer science of the students, two versions of an educational game were created. In the game, the player has to build a zoo. The game is shown in figure 2.

Figure 2: The main screen of the puzzle version educational game. The only difference with the non-puzzle version was that there was a code button on the bottom right.

Throughout the game, the player would get multiple tutorials (which are a mixture of instructional text and worked examples) while trying to do this. An example of a part of a tutorial can be seen at the bottom of figure 2. The difference between the two versions of the game was that in one of the versions the player has to complete puzzles to progress in the game. In the other version the player progressed by buying things for the zoo. Both versions of the game had instructional support and worked examples, but the version with puzzles also had a different support tactic: erroneous worked examples. The puzzles contained buggy code that had to be solved according to the instructions that were shown above the code. An example of a puzzle is shown in figure 3. There were sixteen puzzles in total.
The player unlocked new tutorials, animals, animal enclosures and decoration by progressing in the game. New things to place in the zoo could be bought with coins (in both versions) or diamonds (in the puzzle version, where the player gets a diamond after solving a puzzle). In figure 2 a progress bar can be seen in the top right corner of the screen which was based on how many decorations, animals and animal disclosures the player has. In the non-puzzle version of the game, the player got tutorials at certain values of the progress bar, while in the puzzle version the player got tutorials after certain puzzles are completed.

When the game is first started the player got a tutorial that introduced the goal of the game and explained the game mechanics (how to use the shop to place items in the zoo) and gave the player an introduction to programming. After that, a tutorial in the puzzle version explained how the puzzles work and the player had to solve their first puzzle. Then in both versions the player got a

Figure 3: An example of a puzzle in the puzzle version of the educational game

new tutorial with an introduction to variables. The player was now free to make more puzzles (in the puzzle version) or buy new items. After a while, the player would get a new tutorial, which showed them how to use variables in the game. The name of each animal was a variable, and the player could change this name using the terminal in the game, which is shown on the right side of figure 4.
After some time, more variables were unlocked (hunger, thirst and health) which go down over time, and the player had to try to keep them at a high number because otherwise they got less coins. The player went on to expand their zoo until the next tutorial was unlocked which was about arithmetic with variables. A new variable for the animals was introduced: satisfaction, which is the hunger thirst and health of an animal divided by three. This variable could not be changed in the terminal window. The next tutorial was an introduction to if-statements, and the one after that went more in depth about if-statement conditions. The final tutorial was about else-statements. Throughout the game, the player unlocked new animals, animal enclosures and decoration to give the player motivation to keep playing. All of the code that the player learned about is in the programming language Python.

The choices that were made during the development of the game are shown and explained in table 3.3. On the left, the choices are stated and on the right the reason behind those choices is explained.

3.3.1 Technical details

The game was developed in Godot Engine, version 4. The programming language that was used to develop the game was GDScript. The programming language that was used inside of the game

Choice	Explanation
Game with animals	Because in general animals are a shared interest of children, animals were the main theme of the game.
Narrative	Narrative is an important part of motivation in educational games [Dic05] [Dic06] [Fis05] [War04]. Because of this, narrative was an important consideration when making the game. Narrative was incorporated in the game using the dialog, in which it said that the player had to safe the zoo from having to close forever by making it more succesful.
Money	Another important part of motivation in educational games are rewards [ANVA99] [DJ05] [Jen01].This was incorporated in the game by rewarding the player with money. The player got a diamond when completing a puzzle in the puzzle version, and got money based on how much animals, animal enclosures and decoration the player had in their zoo.
Progress based on puzzles in the version with puzzles	In the version with puzzles, the player would get new tutorials after they completing puzzles, because this was an easy way to track their progress in the game. This way, the player would not get tutorials while they still had to complete a lot of puzzles from different tutorials.
Progress based on progressbar in the version without puzzles	In the version without puzzles, the player would get a new tutorial after reaching a certain percentage of the progressbar to make sure that the player had a good balance of game play and educational content, since they could not get multiple tutorials right after each other this way.
Programming concepts: variables and if-statements	The programming concepts that were chosen for this game were variables and if-statements. Variables were chosen because they were the most basic programming concept that might be the easiest to grasp for beginners. If-statement were chosen for the students that were finished early with the variable part of the game, because they are a programming concept that is a bit more challenging.
Python	The programming language that the players learned to program in in the game is Python. This language was chosen because it is considered to be one of the easier programming languages.
Terminal in the game	In the game, the player could change the variables of the animals using a terminal. This was chosen because this way the children could actually do something with the programming concepts that they learned in the game.
Animal variables decrease over time	In the game, after a certain point the animals have variables that decrease over time. This was incorporated in the game to encourage the students to actually use the terminal in the game.

Table 1: Different choices that were made during the development of the game.

Figure 4: The terminal window in the educational game (non-puzzle version). The name of the giraffe is about to be changed.

to teach students about programming concepts was Python. Most of the assets that were used in the game were self made, however the UI assets were taken from https://kenney.nl/. The programming tutorials and puzzles were created for this game specifically. The game was exported for HTML5 and uploaded to the website https://itch.io/. On this website, both versions of the game can be played. The link of the version with puzzles is https://anikvd.itch.io/dierentuin and the link of the version without puzzles is https://anikvd.itch.io/dierentuin-zonder-puzzels. All of the participants of the study played the game on a Chromebook.

3.4 Measurements

To measure the students' interest in and understanding of computer science, three questionnaires were used. They were all taken online in the form of one big questionnaire right after the students played the educational game. In this questionnaire the students first had to write down their name, age, gender, grade and school.

The first part of the questionnaire inquired about the previous programming experience of the students. This part contained six questions, which asked them what programming languages they knew, whether they already had programming lessons and if yes how long and about what, to grade their programming experience on a scale of one to ten and lastly how comfortable they were with programming. All of the questions were multiple choice except for the question that asked what their previous programming lesson was about.

The next part of the questionnaire was a TAM survey which measured the students attitude towards the game, based on [LHC13]. Questions were answered on a five-point Likert scale, which ranged from neutral to fully agree. In total there were eleven questions, and two additional questions for the students that had the version of the game with puzzles. Four of the questions measured perceived usefulness (six for the puzzle version), four of the questions perceived easy of use and the last three the attitude towards the technology.

The last part of the survey measured what the students learned from the game. This was done through a short "test" consisting of five multiple choice questions. Four of the questions were about variables, and one of the question was about if-statements (which students only had to answer if they got far enough in the game to get the tutorial about if-statements).

3.5 Analysis

To process the data from the questionnaires, multiple steps were taken. First, the data was pseudonymised. Then, the answers on the programming experience, TAM and conceptual programming understanding questionnaires were converted to numerical answers. The answers to the programming experience questionnaire were converted as follows: all of the questions were converted to a score from 0 to 2 where zero indicated the least experience and two the most, except for how long the children have been programming and how comfortable they are with programming (which were scored from 1 to 4), the grade they gave to their programming experience (on a scale of 1 to 10) and the open question where the students explained what programming they had done before. The answers to the TAM questionnaire were also converted to where neutral was 1 and fully agree 5. The conceptual programming understanding questions were graded as follows: an answer was changed to a 1 if the students answer was correct and a 0 otherwise. The mean and median were calculated for all of the participants for the three different questionnaires. For the TAM questionnaire the mean was calculated for every different part: perceived usefulness (which consisted of four questions (or six for the puzzle version) about whether the students found the game useful for learning), perceived ease of use (which asked four questions about how easy it was to use the game) and the attitude towards using the technology (which consisted of three questions about whether the students liked games in general).

The data from the questionnaires was analyzed using the Statistical Package for the Social Sciences (SPSS). To determine whether the game was successful in terms of interest in computer science and teaching programming concepts, the mean of the TAM survey (and the mean of the different parts of the TAM survey: perceived usefulness, perceived ease of use and attitude towards the technology) and of the questionnaire that measured the understanding of the programming concepts was calculated. The mean, std. deviation and std. error mean of the PU, PEU, AT and the test were calculated for the group that had to solve puzzles and the group that did not. To determine whether the group with the puzzles was different from the group without puzzles in terms of interest in computer science a non-parametric test (independent-samples median test) for each of the medians of the three different parts of the TAM questionnaire (perceived usefulness, perceived ease of use and attitude towards the technology) was used. This test had to be non-parametric because the distribution was not normal. To determine whether the group without the puzzles

differed from the group with puzzles in terms of understanding of computer science concepts, an independent-samples median test was performed on the medians of the questionnaire that measured the conceptual programming understanding of the students. To measure the impact of previous programming experience, firstly the mean of the previous programming experience questionnaire was calculated overall and for the group that solved puzzles and the group that did not. Then, the Pearson correlation coefficient was calculated to test if there was a relation between the previous programming experience of the students and the answers on the questionnaire that measured conceptual programming understanding.

4 Results

In this section, the results of the analysis of the questionnaires are reported. To determine whether the group that had to solve programming puzzles differed from the group without puzzles in terms of understanding of computer science concepts and interest in computer science, the descriptive statistics were calculated, and multiple independent-samples median test were performed. This was also done for the participants of the groups together, to determine whether the educational game was effective and fun.

4.1 Previous programming experience

The first questionnaire was about the students previous programming experience. First, average scores were calculated for the total group of children of both classes together. Second, average scores were calculated for the group that did not have programming puzzles, and average scores were calculated for the group that did have programming puzzles. Third, the correlation between the previous programming experience and the answers on the questionnaire that measured conceptual programming understanding was calculated.

The average score was M = 1.6 (SD = 0.4) for the total group of children of both classes together. The average score of the group without puzzles was M = 1.6 (SD = 0.4) and the average of the group with puzzles was M = 1.6 (SD = 0.4). The students were also asked to grade their programming experience of before the programming lesson from 1 to 10, and the average score of this grade was M = 6.2 (SD = 2.7). On average, the group without puzzles gave themselves a grade of M = 7,2 (SD = 2.3) and the group with puzzles gave themselves a grade of M = 4.5 (SD = 2.5). The correlation between the previous programming experience of the students and the answers on the questionnaire that measured conceptual programming understanding was moderately positive (r = 0.50). The correlation between the grade that the students gave their previous programming experience and the answers on the questionnaire that measured conceptual programming understanding was very weakly negative (r = -0.04).

4.2 Interest in computer science

To determine whether the game boosted the interest in computer science of the students the Technology Acceptance Model was used. The scores on this questionnaire ranged from 1 to 5. First, the general interest in computer science after playing the game was calculated. Second, the difference in interest between the group that had to solve puzzles and the group that did not was explored. To do this, the average was calculated for every part of the three parts of the TAM questionnaire (perceived usefulness, perceived ease of use and attitude towards the technology) for every student, and these scores were used to calculate the descriptive statistics and to perform multiple independent-samples median tests. The TAM questionnaire asked the students eleven questions in total with two extra questions for the students that had the version of the educational game that used puzzles, and had an average score of M = 3.7 (SD = 0.6) for both of the groups combined. Four of the questions were about the perceived usefulness (six for the puzzle version), which had an average score of M = 3.3 (SD = 1.1) for the total group of children of both classes together. Four more questions were about perceived ease of use (M = 3.1, SD = 1.0) and the last

three questions were about the attitude towards using the technology (M = 4.5, SD = 0.5). The median of the perceived usefulness (PU) was Mdn = 3.5, the median of the perceive ease of use (PEU) was Mdn = 3.25 and the median of the attitude towards using the technology was Mdn = 4.7. Figure 5 shows the box plot of the different different components of the TAM (PU, PEU and AT) of the total group of children of both classes together.

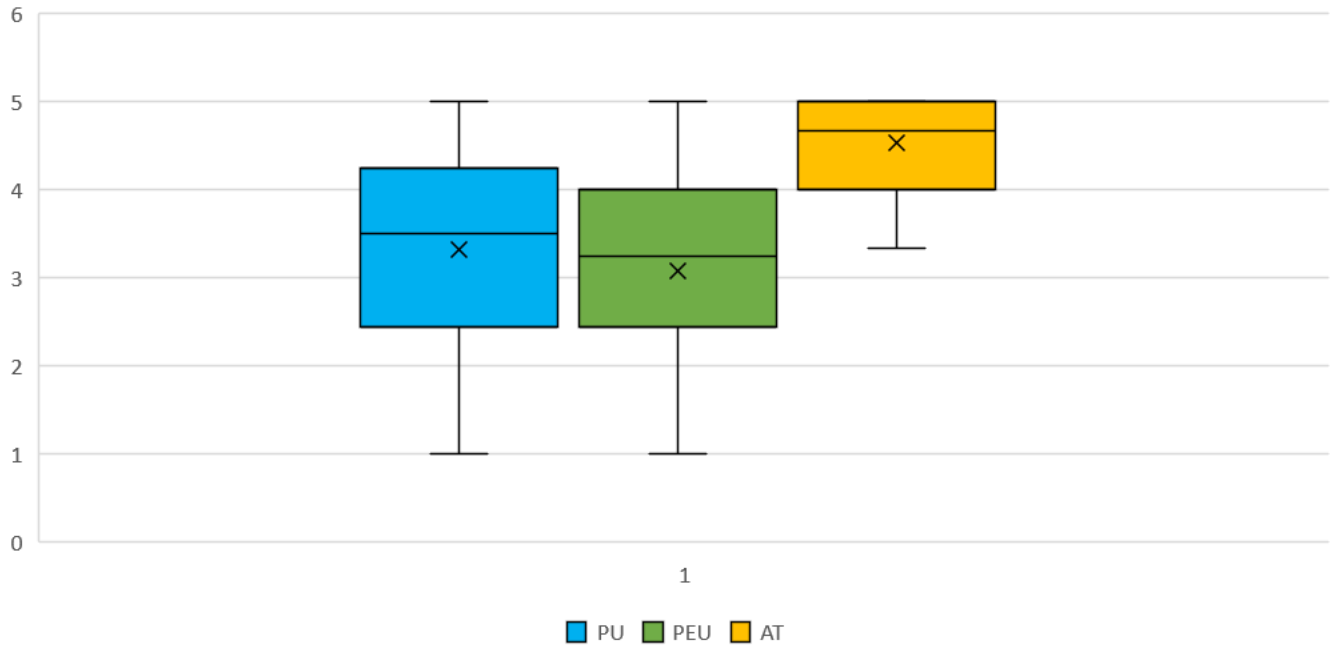

Figure 5: Box plot of the different components of the TAM (PU, PEU and AT) of the total group of children of both classes together.

In order to determine whether there was a difference in interest in computer science of the students that played the version of the game that did have puzzles and the students that played the version of the game that did not have puzzles, the descriptive statistics were calculated and multiple independent-samples median tests were performed. The mean, median, std. deviation and std. error mean of the PU, PEU and AT of the two groups can be found in table 2. In figure 6, the box plot of the different components of the TAM (PU, PEU and AT) is shown for both of the groups. For the different parts of the TAM questionnaire, an independent-samples median test was performed. No significant difference between the groups was found for the perceived usefulness (PU) ($p = 0.707$, $df = 1$), perceived ease of use (PEU) ($p = 0.442$, $df = 1$) and attitude towards using the technology (AT) ($p = 0.454$, $df = 1$).

4.3 Understanding of computer science concepts

To determine the understanding of computer science concepts of the students a questionnaire was used. First, the general understanding of computer science concepts was calculated. Second, the difference in understanding between the group that had to solve puzzles and the group that did not was explored. To do this, the average of the questions on the questionnaire was calculated for every student, and this was used to calculate the descriptive statistics and to perform an

	Mean		Median		Std. Deviation		Std. Error Mean	
	No puzzles	**Puzzles**	**No puzzles**	**Puzzles**	**No puzzles**	**Puzzles**	**No puzzles**	**Puzzles**
PU	3,3	3,4	3,5	3,7	1,0	1,3	,2	,4
PEU	3,2	2,9	3,3	3,0	1,1	0,7	,3	,2
AT	4.7	4.7	4,7	5,0	0,5	0,6	,1	,2
Understanding of concepts	0,3	0,5	0,4	0,4	0,2	0,1	,0	,0

Table 2: Descriptive statistics of the TAM and the questionnaire that measured the understanding of the programming concepts divided by the group that had puzzles and the group that had no puzzles. The mean and median of PU, PEU and AT are scored from 1-5 and the mean of test from 0-1.

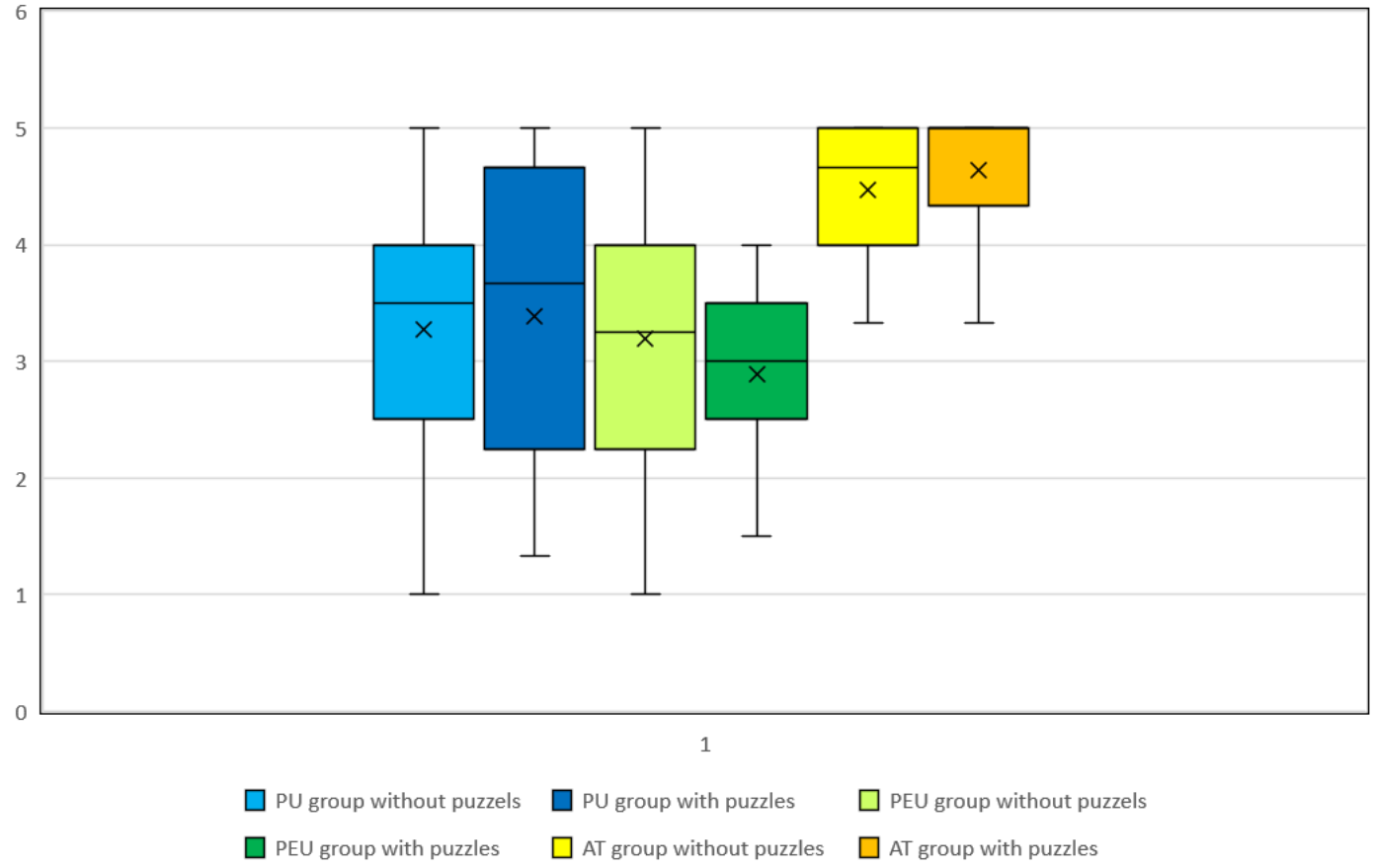

Figure 6: Box plot of the different components of the TAM (PU, PEU and AT) of the group that played the version of the game with puzzles and the group that played the version without puzzles.

independent-samples median test. The questionnaire contained five multiple choice questions about variables and if-statements. Since the questions only had one correct answer, the score of a question was either 0 (false) or 1 (true). For every student, the mean of their answers was calculated, which ranged from 0 to 1. The average score of the total group of children of both classes together on the

questionnaire was M = .4 (SD = 0.2). The median of the total group of children was Mdn = .4. Figure 7 shows the box plot of the scores of the students on the questionnaire of the total group of children of both classes together.

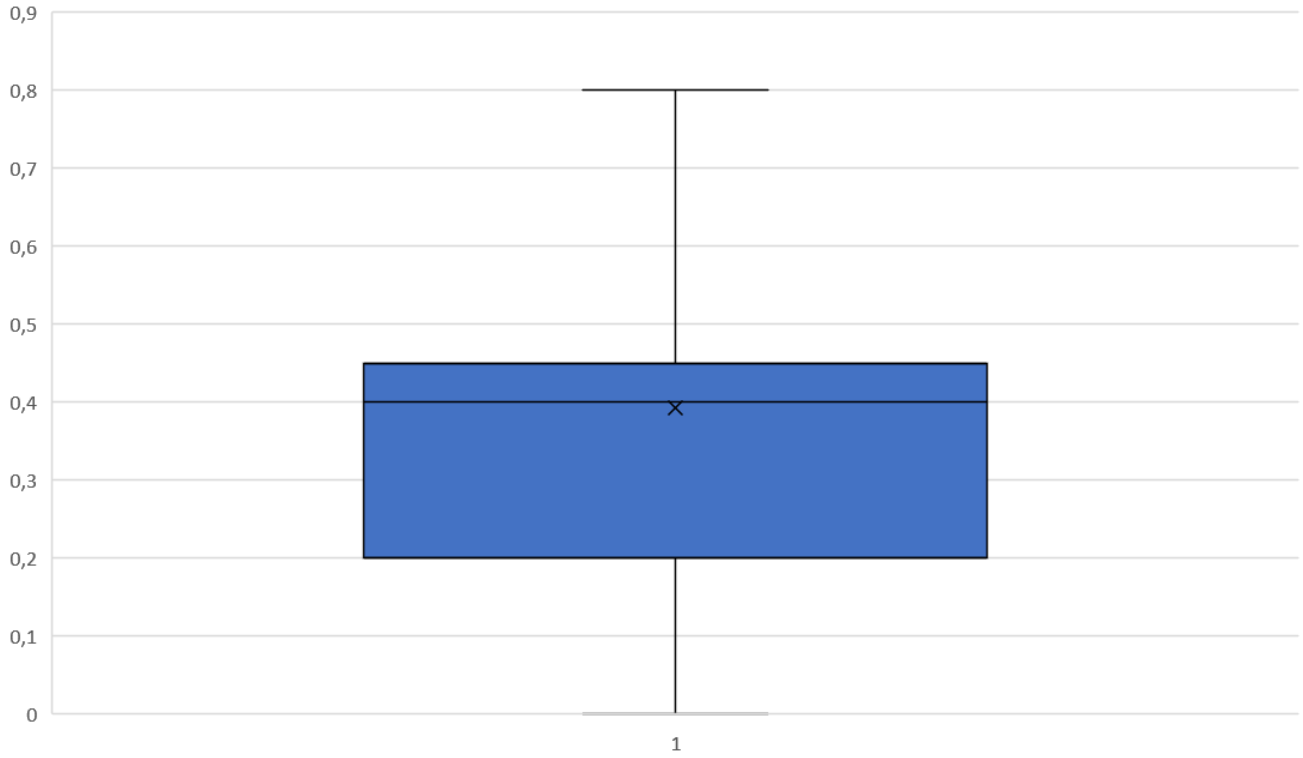

Figure 7: Box plot of the scores of the students on the questionnaire that measured the understanding of computer science concepts of the total group of children of both classes together.

In order to determine whether there was a difference in understanding of computer science concepts of the students that played the version of the game that did have puzzles and the students that played the version of the game that did not have puzzles, the descriptive statistics were calculated and an independent-samples median test was performed. The mean, median, std. deviation and std. error mean for the groups can be found in table 2. In figure 8 the box plot of the scores on the questionnaire is shown for both of the groups. An independent-samples median test was performed, which found no significant difference between the groups (p = 0.372, df = 1).

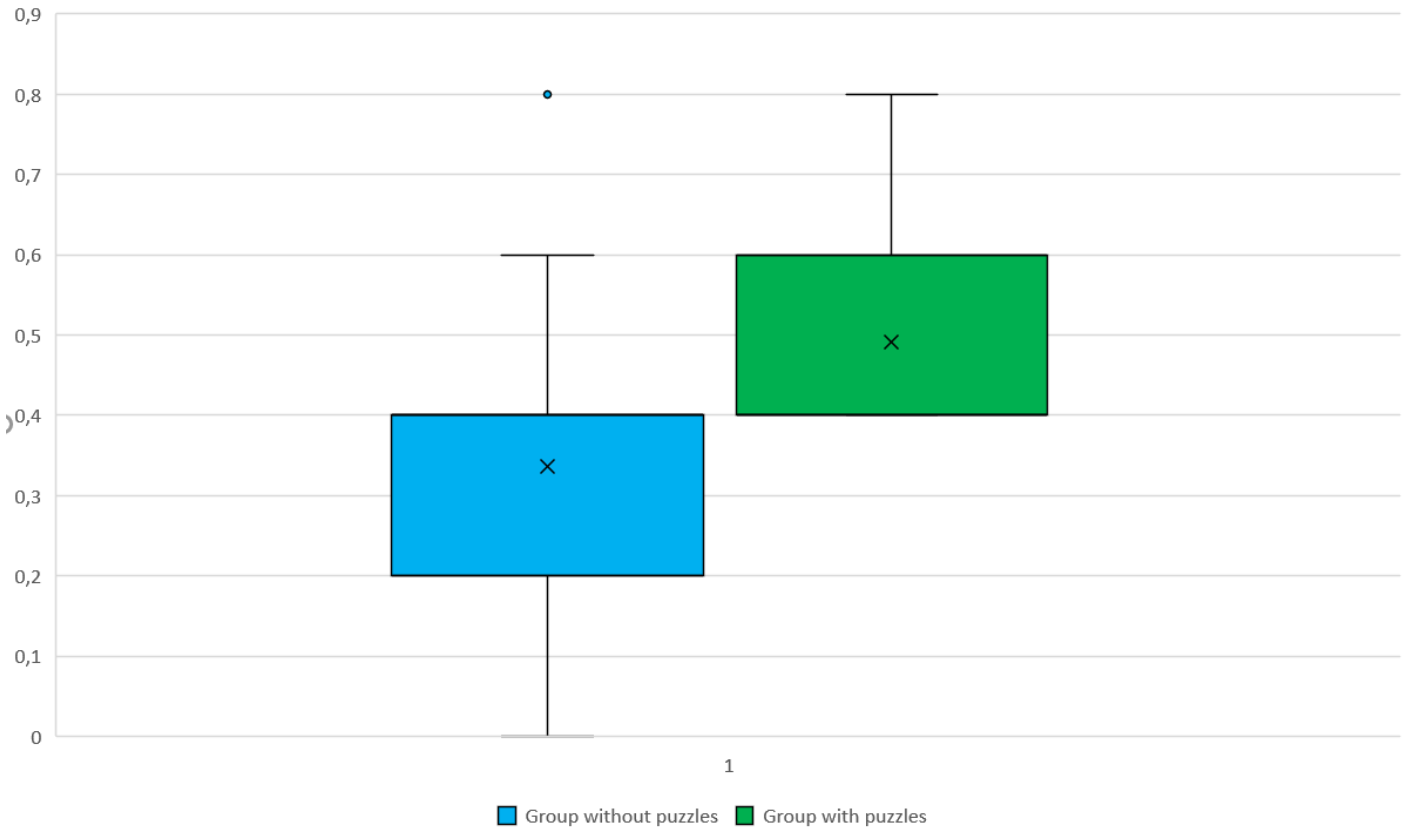

Figure 8: Box plot of the scores of the students on the questionnaire that measured the understanding of computer science concepts of the group that played the version of the game with puzzles and the group that played the version without puzzles.

5 Discussion

The goal of this study was to investigate the role of instruction method on the interest in computer science and the understanding of computer science concepts of primary education students in an educational game with animals. This role was investigated through two versions of an educational game with different instruction methods. Both of the versions of the educational game had instructional support in the form of instructional text and examples, however one of the versions also had erroneous support, where the student had to solve buggy code in puzzles. To investigate the differences in interest and understanding of computer science that students had after playing one of the two versions, three questionnaires were taken after the playing the game. The first questionnaire measured previous programming experience, the second was a TAM questionnaire and the third measured the understanding of the programming concepts of the student. The answers to these questionnaires were analyzed. The results of this analysis will be explored in this discussion and used to answer the research question. The research question was as follows:

RQ What role does the instruction method have on the interest in computer science and the understanding of computer science concepts of primary education students in an educational game with animals?

5.1 Understanding of computer science concepts

To explore the role of instruction method on the understanding of computer science concepts of students, a questionnaire about the understanding of the computer science concepts of the students was conducted after the students played the game. First, the general understanding of computer science concepts of students gets explored. After that, the difference in understanding between the group that had to solve puzzles and the group that did not is discussed.

First, concerning the general understanding of computer science concepts of students, the students who had more previous programming experience had more correct answers on the questionnaire that measured the understanding of programming concepts than the students that had less previous programming experience. The students with more programming experience might have already known more about programming concepts or had a better understanding of computer science in general than the students with less experience which may have caused them to perform better on the questionnaire. On average, the students had less than half of the questions of the questionnaire that measured understanding of computer science concepts correct. The students did good on the questions about variables, however almost no students had the question about the instruction print and if-statements correct, which brought the overall score down. A lot of students did not get to the part of the game where if-statements were explained, which explains why many students got the if-statement question wrong. The game was too long to fully play in the short time that the students had and should have focused on only variables, since variables were already hard to grasp for the elementary education students. This is in line with previous research, which found that programming concepts are difficult to learn [PH11] [GX18]. While the print instruction was discussed in the lesson about programming at the beginning of the class and explained once in the educational game, it was not explained as often as variables, which is why the students might not have understood it as well. The code terminal is a component of the game that could be used more

in a future version of the game, because it might help students gain a deeper understanding of the subject. Overall, the game was effective in teaching the students about variables since they did get a lot of questions about variables correct, and thus did improve their understanding of computer science concepts. In future research, the game should focus on variables only or the research should consist of multiple lessons that are spread across multiple weeks. In previous research, Frazer et al. found that most educational mini-games are too short to truly immerse the players [FAW07]. Because of this, it would be more beneficial to have a future game consist of multiple lessons. However, it might be harder to find schools that are willing to participate in longer research. Because of this, a shorter game that focuses on variables only instead of on multiple programming concepts might be more realistic yet still more effective in teaching than the current game.

Second, concerning the difference between the puzzle and non-puzzle group, no difference was found between the two groups. Because of this, no conclusion can be drawn about whether erroneous examples had a positive or negative impact on the understanding of programming concepts of the students. This lack of result might be because of the small group of students that the research was conducted on, however a previous study [ZLP18], where the researchers found that erroneous worked examples were more effective compared to worked examples and instructional text, also had a small group of students. Adams et al. [AMD+12] found that erroneous worked examples did not immediately benefit students but did later on, which suggests that erroneous examples causes a deeper learning experience. Perhaps the students that played the version of the game that had puzzles and thus erroneous support did benefit later on, however no questionnaire was conducted at a later time so there is no evidence to support this. Previous research also found that erroneous worked examples work best when the student already has some prior knowledge about the subject [GR07]. It is possible that the students that played the version with puzzles did not have enough previous knowledge to truly benefit from the erroneous worked examples. This might be improved in future research by highlighting errors [GR07]. Another improvement could be to have a tutor in the game or have the teacher play a more active role. Intelligently-tutored problems give feedback or hints when the student makes a mistake or asks for it. Intelligently-tutored problems work better if the student has no prior knowledge about a subject [Rit07] [MvGG+14]. Overall, future work might be able to find more of a difference between a group that has erroneous support and a group that does not if the students had more previous knowledge or if future research incorporates a tutor in the game or have the teacher play a more active role in students understanding of the programming concepts.

5.2 Interest in computer science

To investigate the interest of the students in computer science after playing the educational game, a TAM questionnaire was conducted. First, the general interest of students in computer science is discussed. After that, the difference in interest in computer science between the group that had to solve puzzles and the group that did not is discussed.

First, concerning the general interest of students in computer science, the students thought that they did somewhat learn from the game. This indicates that the game did teach them about programming concepts. There are multiple factors that could be improved in a future version of the game to make the students learn even more from the game. Firstly, the game was too long and

wanted to teach too many programming concepts. The game should have focused on one topic, variables, and made sure that the students fully understood that topic. Because of this, students might not have had a deep understanding of the concepts, which might have made them feel like they did not learn as much from the game. Secondly, the substance of the game might have been to challenging for the students. A lot of the students did not have many questions correct on the questionnaire that measured the understanding of the programming concepts of the students. This indicates that the students did not fully understand some of the concepts which might have made them feel like they did not learn a lot from the game. The students found the game somewhat easy to use. Students might have experienced some frustration while playing the game because of two reasons. Firstly, the screen of Chromebooks that the students played the game on was very small. Because of this, the formatting of the game was a bit different than intended. This was especially a problem for the group that did not have puzzles. The problem was fixed for the group that did have puzzles, however the formatting was still not ideal. This might have caused problems when playing the game for the students, and they may not have found the game very easy to use as a consequence. Secondly, before the student played the game, they already knew that it was possible to change the variables of the animals in the game since this was show in the introduction before students had to play the game. However, this was only unlocked after student made a certain amount of progress in the game, which might have caused some confusion. Further research should make sure the game is flawless to ensure that students do not experience frustrations because some things in the educational game are not intuitive or because there is a bug in the game. In general, the students really like video games. Because of this, students might be more enthusiastic while playing an educational game compared to traditional learning. This is in line with the findings of the literature [Sta15] [JLP13] [AAA15] [Pap09]. Looking at the literature, motivation can be attributed to the narrative context of the game [Dic05] [Dic06] [Fis05] [War04]. In future work, the motivation of the players of the game could be improved by enhancing this narrative, since the narrative is now only mentioned in the beginning of the game. A component of the game that could be used more is the code terminal, since this might motivate the students to be more interested in what is taught because they can actually use it to improve the zoo. It is also possible that the students would be even more interested in programming if the game was not about a zoo and had a different subject. Future research could also explore what the impact is of the theme on the interest in computer science of the students. Overall, the students found the game somewhat useful, somewhat easy to use and really like video games in general. In hindsight, there were things that could be improved about the game (as described above), however the game was in general successful in enhancing the interest of students in computer science.

Second, regarding the difference between the puzzle and non-puzzle group, no difference was found between the two groups. Because of this, no conclusion can be drawn about whether erroneous examples had a positive or negative impact on the interest in computer science of the students. Since there is not much literature available about how the interest of primary school students in programming is affected by instructional support, it is not clear whether the support does not have much effect in general or only specifically in this case. There also might be no difference because of a reason unrelated to the instructional support. Students might have found the substance of the game too challenging in general. If a challenge is above the students abilities it could bring up negative feelings [Man20] [Don07], causing them to have less interest in the game. Because not much research has been done on the role of instruction method on the interest of primary

school students in computer science, it is too soon to draw a conclusion. Further research should investigate this further.

5.3 Limitations

Originally, the two primary school classes were each supposed to have half of the class play the game with the puzzles and have the other half play the game without the puzzles. Unfortunately, because of technical difficulties with the version with puzzles with the first class, it was decided that the first class would play the version of the game without puzzles and that the second class would play the version with puzzles. Because of this, one of the groups might have had more programming experience than the other, since one school might have devoted more time to programming than the other one. This might have influenced the findings of the research: if for example the version of the game with puzzles had a more positive impact on the students, however the group that played the version of the game without puzzles had more previous programming experience, it might have seemed as if there was no difference between the two groups because the students that played the game without puzzles did equally well because they had more previous experience.

A second limitation of the study was that the students might not all have understood that the questions about their previous programming experience was about the knowledge that they had before playing the game. A lot of the students answered "zoo" or "python" to the open question about their previous programming experience. In further research, the questionnaire about the students previous programming knowledge should be answered before playing the educational game to prevent this miscommunication.

A final limitation of the study is that the research was conducted on small groups of students, which are not representative for all of the primary education students in the Netherlands. A future direction of the research could be to do a study with a lot more participants.

5.4 Conclusion

This thesis investigated the role of instruction method on the interest in computer science and the understanding of computer science concepts of primary education students in an educational game with animals. To do this, two version of an educational game were developed. Both of the versions had instructional support and worked examples, however the version with puzzles also had a different support tactic: erroneous worked examples. Overall, the understanding of computer science concepts was enhanced because of the game. The students found that they somewhat learned something from the game, found the game somewhat easy to use and really like video games in general. This study did not find a difference in effect of instructional support on the understanding of computer science concepts and interest in computer science of the primary education students. Thus, the research question can be answered as follows: instruction method did not play a role in the interest in computer science and the understanding of computer science concepts of primary education students in an educational game with animals.

References

[AAA15] Yigal Attali and Meirav Arieli-Attali. Gamification in assessment: Do points affect test performance? *Computers Education*, 83:57–63, 2015.

[Abt70] Clark C Abt. Serious games. *New York, Viking Press*, 1970.

[AMD+12] Deanne Adams, Bruce M. McLaren, Kelley Durkin, Richard E. Mayer, Bethany Rittle-Johnson, Seiji Isotani, and Martin Van Velsen. Erroneous examples versus problem solving: Can we improve how middle school students learn decimals? *Cognitive Science*, 34, 2012.

[ANVA99] Alan Amory, Kevin Naicker, Jackie Vincent, and Claudia Adams. The use of computer games as an educational tool: identification of appropriate game types and game elements. *Br. J. Educ. Technol.*, 30:311–321, 1999.

[CB07] Michael Caspersen and Jens Bennedsen. Instructional design of a programming course: A learning theoretic approach. *Third International Computing Education Research Workshop, ICER'07*, pages 111–122, 09 2007.

[CKTS01] Paul Chandler, Slava Kalyuga, Juhani E Tuovinen, and John Sweller. When problem solving is superior to studying worked examples. *Journal of Educational Psychology*, 93(3):579 – 588, 2001.

[CM12] Stelios Xinogalos Christos Malliarakis, Maya Satratzemi. Educational games for teaching computer programming. *Research on e-learning and ICT in Education Technological, Pedagogical and Instructional Perspectives*, september 2012.

[CP18] Ricardo Casañ Pitarch. An approach to digital game-based learning: Video-games principles and applications in foreign language learning. *Journal of Language Teaching and Research*, 9:1147–1159, 11 2018.

[CSG03] D. Bavelier C. S. Green. Action video game modifies visual selective attention. *Nature, 423(6939)*, page 534–537, 2003.

[Dav86] Fred D Davis. A technology acceptance model for empirically testing new end-user information systems: Theory and results. *Ph.D. Dissertation. Massachusetts Institute of Technology.*, 1986.

[Dav89] Fred Davis. Perceived usefulness, perceived ease of use, and user acceptance of information technology. *MIS Quarterly*, 13:319–, 09 1989.

[dF18] Sara de Freitas. Are games effective learning tools? a review of educational games. *Journal of Educational Technology Society*, 21(2):74–84, 2018.

[Dic05] Michele Dickey. Three-dimensional virtual worlds and distance learning: Two case studies of active worlds as a medium for distance education. *British Journal of Educational Technology*, 36:439 – 451, 05 2005.

[Dic06] Michele Dickey. Ninja looting for instructional design: the design challenges of creating a game-based learning environment. 07 2006.

[DJ05] Guillaume Denis and Pierre Jouvelot. Motivation-driven educational game design: Applying best practices to music education. pages 462–465, 06 2005.

[DJM11] D. P. Robertson D. J. Miller. Educational benefits of using game consoles in a primary classroom: A randomised controlled trial. *British Journal of Educational Technology, 42(5)*, pages 850–864, 2011.

[Don07] Mary Jo Dondlinger. Educational video game design: A review of the literature. *Journal of Applied Educational Technology*, January 2007.

[DUO21] DUO. Monitor digitale geletterdheid in het po 2020-2021. *DUO Onderwijsonderzoek & Advies*, 2020-2021.

[ES09] I. Yakin E. Sumuer. Effects of an educational game development course on preservice teachers' concerns about the use of computer games in the classroom. *Proceedings of the 9th International Educational Technology Conference*, may 2009.

[FAW07] Alex Frazer, David Argles, and Gary Wills. Is less actually more? the usefulness of educational mini-games. pages 533 – 537, 08 2007.

[Fis05] Shalom Fisch. Making educational computer games "educational". 06 2005.

[GKV08] Glenda A. Gunter, Robert F. Kenny, and Erik Henry Vick. Taking educational games seriously: using the retain model to design endogenous fantasy into standalone educational games. *Educational Technology Research and Development*, 56:511–537, 2008.

[GR07] Cornelia S. Große and Alexander Renkl. Finding and fixing errors in worked examples: Can this foster learning outcomes? *Learning and Instruction*, 17(6):612–634, 2007.

[Gro07] Begoña Gros. Digital games in education. *Journal of Research on Technology in Education*, 40(1):23–38, 2007.

[GX18] Andreas Giannakoulas and Stelios Xinogalos. A pilot study on the effectiveness and acceptance of an educational game for teaching programming concepts to primary school students. *Education and Information Technologies*, 23:1–24, 09 2018.

[Jen01] Morgan Jennings. Best practices in corporate training and the role of aesthetics: interviews with eight experts. In *Proceedings of the 2001 ACM SIGCPR Conference on Computer Personnel Research*, SIGCPR '01, page 215–219, New York, NY, USA, 2001. Association for Computing Machinery.

[JFK10] Bryan Tregunna Steve Jarvis Richard Smithies Sara de Freitas Ian Dunwell Kevin Mackway-Jones James F Knight, Simon Carley. Serious gaming technology in major incident triage training: a pragmatic controlled trial. *Resuscitation Journal, 81(9)*, september 2010.

[JLP13] B. D. Homer J. Case E. O. Hayward M. Stein K. Perlin J. L. Plass, P. A. O'Keefe. The impact of individual, competitive and collaborative mathematics game play on learning, performance, and motivation. *Journal of Educational Psychology*, 2013.

[KB16] M. Kandroudi and T. Bratitsis. Teaching programing in small ages with mobile devices through kodable and scratchjr: Case study. In *10th Pan-Hellenic and International Conference "ICT in Education", 8th Conference "Teaching of Informatics" of the Hellenic Scientific Association of ICT in Education, 25-27*, september 2016.

[KGL+13] Simone Kühn, Tobias Gleich, Robert Lorenz, Ulman Lindenberger, and J Gallinat. Playing super mario induces structural brain plasticity: Gray matter changes resulting from training with a commercial video game. *Molecular Psychiatry*, Advance online publication, 10 2013.

[KKBM12] Cagin Kazimoglu, Mary Kiernan, Liz Bacon, and Lachlan Mackinnon. Learning programming at the computational thinking level via digital game-play. *Procedia Computer Science*, 9:522–531, 12 2012.

[KP05] Caitlin Kelleher and Randy Pausch. Lowering the barriers to programming: A taxonomy of programming environments and languages for novice programmers. *ACM Comput. Surv.*, 37:83–137, 01 2005.

[KRMSA11] Michael D. Kickmeier-Rust, Elke E. Mattheiss, Christina M. Steiner, and Dietrich Albert. A psycho-pedagogical framework for multi-adaptive educational games. *Int. J. Game Based Learn.*, 1:45–58, 2011.

[KS4] S. Karadeniz, Y. Samur, and M.Y. Özden. Playing with algorithms to learn programming: A case study on 5 years old children. 9th international conference on information technology and applications. July 2014.

[LHC13] Yi Hsuan Lee, Yi Chuan Hsieh, and Yen Hsun Chen. An investigation of employees' use of e-learning systems: Applying the technology acceptance model. *Behaviour and Information Technology*, 32(2):173–189, February 2013.

[LK15] Michael Lee and Andrew Ko. Comparing the effectiveness of online learning approaches on cs1 learning outcomes. pages 237–246, 07 2015.

[LK24] Maaike Oberink Nataša Grgurina Jos Spronk Hans de Vries Martin Klein Tank Luuk Kampman, Matthijs Driebergen. *Conceptkerndoelen burgerschap en digitale geletterdheid*. SLO, March 2024.

[Man20] Dionysios Manesis. *Digital Games in Primary Education*. 02 2020.

[MBG14] Ashley F. McDermott, Daphne Bavelier, and C. Shawn Green. Memory abilities in action video game players. *Computers in Human Behavior*, 34:69–78, 2014.

[MG14] Nikola Marangunic and Andrina Granic. Technology acceptance model: a literature review from 1986 to 2013. *Springer-Verlag Berlin Heidelberg*, February 2014.

[MvGG+14] Bruce M. McLaren, Tamara van Gog, Craig Ganoe, David Yaron, and Michael Karabinos. Exploring the assistance dilemma: Comparing instructional support in examples and problems. In Stefan Trausan-Matu, Kristy Elizabeth Boyer, Martha Crosby, and Kitty Panourgia, editors, *Intelligent Tutoring Systems*, pages 354–361, Cham, 2014. Springer International Publishing.

[PA85] Peter Pirolli and John R. Anderson. The role of learning from examples in the acquisition of recursive programming skills. *Canadian Journal of Psychologyrevue Canadienne De Psychologie*, 39:240–272, 1985.

[Pap09] Marina Papastergiou. Digital game-based learning in high school computer science education: Impact on educational effectiveness and student motivation. *Computers Education*, 52(1):1–12, 2009.

[PH11] Martinha Piteira and Samir Haddad. Innovate in your program computer class: An approach based on a serious game. 07 2011.

[Pia13] J. Piaget. Play, dreams and imitation in childhood. *New York, NY: Routledge*, 25, 2013.

[PMN14] Sedano Hoyuelos Máximo Peña-Miguel Noemí. Educational games for learning. *Universal Journal of Educational Research*, 2014.

[PRS10] Fred Paas, Alexander Renkl, and John Sweller. Cognitive load theory and instructional design: Recent developments. *Educational Psychologist*, 38:1–4, 06 2010.

[RA10] Alexander Renkl and Robert K. Atkinson. *Learning from Worked-Out Examples and Problem Solving*, page 91–108. Cambridge University Press, 2010.

[Rit07] Anderson J.R. Koedinger K. Corbett A. Ritter, S. Cognitive tutor: Applied research in mathematics education. *Psychonomic Bulletin and Review*, 14:249–255, 2007.

[SCJ+11] Leon Straker, Amity Campbell, Lynn Jensen, Deborah Metcalf, Anne Smith, Rebecca Abbott, Clare Pollock, and Jan Piek. Rationale, design and methods for a randomised and controlled trial of the impact of virtual reality games on motor competence, physical activity, and mental health in children with developmental coordination disorder. *BMC public health*, 11:654, 08 2011.

[SK11] C Schilling R Lorenz C Mörsen N Seiferth T Banaschewski A Barbot G J Barker C Büchel P J Conrod J W Dalley H Flor H Garavan B Ittermann K Mann J-L Martinot T Paus M Rietschel M N Smolka A Ströhle B Walaszek G Schumann A Heinz J Gallinat S Kühn, A Romanowski. The neural basis of video gaming. *Translational Psychiatry, 1(11)*, 2011.

[SMB+13] John Stachel, Daniela Marghitu, Taha Brahim, Roderick Sims, Larry Reynolds, and Vernon Czelusniak. Managing cognitive load in introductory programming courses: A cognitive aware scaffolding tool. *Journal of Integrated Design Process Science*, 17:37–54, 01 2013.

[SMP98] John Sweller, Jeroen JG Van Merrienboer, and Fred GWC Paasn. Cognitive architecture and instructional design. *Educational psychology review 10, 3 (1998)*, 1998.

[Sta15] K. Star. Gamification, interdependence, and the moderating effect of personality on performance. 2015.

[VPL11] Arto Vihavainen, Matti Paksula, and Matti Luukkainen. Extreme apprenticeship method in teaching programming for beginners. In *Proceedings of the 42nd ACM Technical Symposium on Computer Science Education*, SIGCSE '11, page 93–98, New York, NY, USA, 2011. Association for Computing Machinery.

[War04] Atif Waraich. Using narrative as a motivating device to teach binary arithmetic and logic gates. volume 36, pages 97–101, 09 2004.

[Wv13] Pieter Wouters and Herre van Oostendorp. A meta-analytic review of the role of instructional support in game-based learning. *Computers Education*, 60(1):412–425, 2013.

[Wvv08] P.J.M. Wouters, E.D. van der Spek, and H. van Oostendorp. Serious games for crisis management: What can we learn from research on animations? In A. Maes S. Ainsworth, editor, *Exploiting the Opportunities: Learning with Textual, Graphical and Multimodal Representations*, pages 162–165. EARLI, 2008.

[YvdM22] Sabiha Yeni and Anna van der Meulen. Students' behavioral intention to use gradual programming language hedy: A technol ogy acceptance model. *Proceedings of the 27th ACM Conference on Innovation and Technology in Computer Science Education Vol. 1 (ITiCSE 2022), July 8–13, 2022, Dublin, Ireland*, 2022.

[ZLP18] Rui Zhi, Nicholas Lytle, and Thomas W. Price. Exploring instructional support design in an educational game for k-12 computing education. *SIGCSE '18: The 49th ACM Technical Symposium on Computer Science Education, Feb. 21–24, 2018, Baltimore, MD, USA*, 2018.

[ZMB13] Goran Zaharija, Sasa Mladenovic, and Ivica Boljat. Introducing basic programming concepts to elementary school children. *Procedia - Social and Behavioral Sciences*, 106:1576–1584, 12 2013.