University of Tartu
Faculty of Social Sciences
Institute of Education
Curriculum of Educational Technology

Dejan Delev

AN EXPERIMENTAL STUDY TO MEASURE THE EFFECT OF AUTOTHINKING GAME ON PRIMARY SCHOOL STUDENTS' COMPUTATIONAL THINKING KNOWLEDGE

MA thesis

Supervisor: Danial Hooshyar, PhD
Senior Researcher in Learning Analytics
Institute of Education, University of Tartu

Tartu, 2020

Abstract

An Experimental Study To Measure The Effect Of AutoThinking Game On Primary School Students' Computational Thinking Knowledge

Despite being listed by several researchers as a fundamental 21st century skill, together with arithmetic and reading, CT has been seen in a negative light by students. Different approaches have been employed to try and make CT more attractive and accessible to students. Educational games are one of those approaches. Even though initial success has been recorded, many of these games do not completely succeed when it comes to acquiring CT skills. In an attempt to address these gaps and improve the existing games, an adaptive educational game, AutoThinking, was developed by a group of researchers. The game aims to teach both CT skills and concepts while also being one of the first games that employs adaptivity in both gameplay and learning. We conducted an experimental study to measure the effect of AutoThinking on Estonian primary school students' CT knowledge. The results of the study show that the game has a positive effect on children and it proved to be a more effective approach compared to a traditional technology-enhanced learning.

Keywords: computational thinking, educational game, adaptivity, primary school students

Table of contents

Introduction

Even though the concept of computational thinking (CT) was envisioned by Seymour Papert in his book Mindstorms (Papert, 1980), it was Jeanette Wing (2006) who actually coined the cutting-edge term - computational thinking. According to her, it involves "solving problems, designing systems, and understanding human behavior by drawing on the concepts fundamental to computer science" (p. 33). In recent times, CT has freed itself from the confined use in STEM (science, technology, engineering and mathematics) domains, and we can see it playing an important role in many other disciplines, such as archeology, medicine, finance etc. Wing, herself, has also stated that along with reading, writing and arithmetics, CT is a fundamental 21st century skill. Some researches (e.g Haddad and Kalaani, 2015) have pointed out that obtaining such a skill could be regarded as a factor for students' academic success and thus it should be taught in early education. Another point supporting this is the fact that many students in the future will work in a field involving or influenced by computing, and teaching CT in K-12 (kindergarten to 12 grade) would only benefit them. Therefore, countries like the United Kingdom, Sweden, Denmark and the United States have already made reformation of educational programs on different educational levels in order to integrate CT into official curricula. All these rapid changes carry certain challenges with them. There have to be enough resources to keep up teachers with experience and knowledge within the area, educate teachers fast enough, etc. At the same time, there are issues like shortage of motivation and opportunities to stimulate students' CT. During their research in 2007, Yardi and Bruckman found that students often express negative attitudes towards learning CT, and this might well hinder its proper development. Hooshyar (2019) argues that the key challenge in promoting CT skills is the lack of opportunities to improve one's CT skills. Therefore, different approaches have to be developed and applied to not only make CT more available and engaging to learners, but also assist teachers in promoting students' CT (both skills and conceptual knowledge). One prospective solution could be the game-based approach, or more commonly known as educational games. Educational games have earned recognition from researchers due to the fact that they seem to be able to capture students' attention and keep their motivation, while at the same time improving their learning achievements. Even though there are several educational games aimed at promoting students' CT, they mostly focus on reinforcing conceptual knowledge and motivation, meaning not enough attention is paid to practicing the conceptual knowledge through game-play. What is more important, the existing educational games ignore providing adaptivity. Instead of focusing more on personalization and adaptation, that would suit the individual needs of the player, most of these games follow predefined and inflexible computer-assisted instruction concepts. This could, therefore, become a huge obstacle in exploring their full educational potential. Considering the importance and relevance of CT in society, as well as the existing gaps in CT game research, some researchers, including Hooshyar et al., (2019), have recently developed an adaptive educational game (called AutoThinking) for fostering students' knowledge of CT. To measure the effectiveness of the AutoThinking game on elementary school students' knowledge of CT, this study designs and conducts a pretest–posttest experimental design. The experimental group are given the AutoThinking game, while the control groups learn with traditional technology-enhanced learning approach (conducting a lesson using a PowerPoint presentation with multimedia delivered by a teacher).

Theoretical overview

Computational thinking

As we mentioned in the introduction, the term computational thinking (CT) was first coined in a seminal article by Wing (2006). From her argument, we can say that CT characterizes a skill to analyze and then to solve various problems. At the time, this was a very refreshing perspective on people's relationship with computers and it generated a plethora of research on CT. Despite this, there has been very little unity among researchers on what the definition for CT might involve. In 2010 and 2011, the National Academy of Sciences organized two workshops in which leading researchers from education and computer sciences departments, as well as leaders from the computing industry explored "the scope and nature of computational thinking" (2010) and the "pedagogical aspects of computational thinking" (2011). In their report they made it clear that computational thinking is not the same as programming and computer literacy and expanded the term to include some key concepts from CS (Computer Science), such as problem decomposition, parallelism, debugging, search strategy, and simulation. Wing (2011) came up with a new definition, stating that "computational thinking is the thought processes involved in formulating problems and their solutions so that the solutions are represented in a form that can be effectively carried out by an information-processing agent." Soon after, Aho (2012) followed up with a simplified version by considering "computational thinking to be the thought processes involved in formulating problems so their solutions can be represented as computational steps and algorithms."

The International Society for Technology in Education (ISTE) and the Computer Science Teachers Association (CSTA), in cooperation with experts from K-12, as well as higher education, succeeded in developing an operational definition of computational thinking as a problem-solving process that includes (but is not limited to) the following characteristics:

- Formulate problems in a way that allows us to use a computer and other tools to help solve them.
- Logically organize and analyze data
- Represent data through abstractions, like models and simulations
- Automate solutions through algorithmic thinking (a string of arranged steps)
- Identify, analyze, and implement feasible solutions with the goal of pulling off the most efficient and effective blend of steps and resources
- Generalize and transfer similar problem solving process to a broad variety of problems

Despite the fact that there is no unanimous consensus on a definition of computational thinking, Grover and Pea (2013) came to a conclusion that researchers have accepted computational thinking to be a cognitive process that employs the elements of abstraction, generalization, decomposition, algorithmic thinking, and debugging. Abstraction is the ability to conceptualize and then represent an idea or a process in more general terms by foregrounding the important aspects of the idea while backgrounding less important features (Weintrop et al. 2015). According to Selby & Woolard (2013), generalization is the ability to move from specific to broader applicability, for example, understanding how to draw a square by defining internal angles, then applying the same algorithm to produce an approximation of a circle. Breaking a complex problem into a few smaller, simpler ones is how we could describe decomposition. Algorithmic thinking is a skill associated with formulating a step-by-step solution to a problem. Finally, debugging, also known as trouble-shooting, is the skill of identifying, analyzing and then corresponding to fix an error.

With broadly accepted definitions of CT, the focus has, in recent years, slowly shifted towards more pragmatic questions like how to foster and assess CT in education.

Educational game in supporting computational thinking

Educational computer games are one possible method to promote CT. In recent years, educational games have widely been used in learning, and the results have shown a positive effect on motivation promotion and knowledge construction as well as the acquisition of high-order abilities (Hwang, Chiu, & Chen, 2015). Since programming and CT share quite a lot of aspects in terms of cognitive skills, in the literature there a lot of learning environments that use programming (or coding) as a form of teaching CT to students. While many established programming languages (e.g. C++ and Java), have representation that closely depicts the way most computers think, a large chunk of these learning environments use representation that is much closer to people's language (Lye & Koh, 2014). Such examples are Scratch (Resnick et al., 2009), Snap! (Harvey & Mönig, 2010), and Blockly (Fraser, 2012). The main point of these learning environments is to make programming more accessible for learners and free young learners from the burden of learning complex syntax. They do so by adapting drag-and-drop interactions rather than writing syntax. This approach has certainly helped reduce the cognitive load on the students and ''allow students to focus on the logic and structures involved in programming rather than worrying about the mechanics of writing programs'' (Kelleher and

Pausch, 2005). However, despite showing initial success in boosting both the motivation and the learning results of the students, these learning environments do have issues. It is of absolute importance to underline that these environments cannot be considered to be educational games because they do not include some of the major components that are part of almost all decent educational games such as opportune feedback and a system that would give students rewards in order to motivate them to discover more. Most of the programming tools give feedback only when certain projects are being run, in the form of display of actions prearranged by the students (Kazimoglu et al., 2011). One other issue that bothers researchers (e.g Meerbaum-Salant, Armoni, and Ben-Ari, 2011), is the fact that these environments fail at promoting deeper learning. According to them, "concepts were only learned when students were explicitly taught the concepts while they created projects that use the concepts." The reason for this is that the environments are only partially matched with the CT skills. Thus, learners lack the opportunities to get involved in the basic thought processes of solving problems (they can't focus because of the different distractions or get overpowered by the syntax, which in this case is given in different forms).

There are also games that support learning different skills by making the most of different elements of educational games. Among those skills are CT (eg., RoboBuilder by Weintrop and Wilensky, 2012). Unlike the environments we mentioned before, these games use various motivating contexts which allow players to get involved in the thought process of developing a solution to a problem. In opposition to the visual programming and block-based environments, these educational games employ different game elements, which gives them the capacity to foster more purposeful learning. Very good examples of such games are Wu's Castle, CodeSpell, as well as MiniColon. There are a few studies (e.g., Esper et al, 2014) that report the positive effect of these games on learners' coding, as well as CT skills. The biggest issue with them, though, is that they also use text-based programming language, which again, requires learners' utmost attention to syntax details (Zhao and Shute, 2019). As such, their alignment with CT is not complete.

In 2012, Kazimoglu et al., created a game called *Program Your Robot*, aimed at promoting CT skills. The initial results showed that the game could improve learners' motivation and that "this approach could develop the problem solving abilities of students who are learning introductory programming." Similarly, Zhao and Shute (2019) designed and developed a web-based video game called *Penguin Go.* The game was designed thoughtfully to be in line with the core components of CT. Findings of their study revealed that students recorded much improved CT test scores by playing the game for less than two hours, whereas no influence was detected on their attitudes toward CT. There are a few other examples of games aimed at fostering CT, such as LightBot (Gouws, Bradshaw, & Wentworth, 2013), RunMarco (https://runmarco.allcancode.com), and games at Code.org. However, their effect on the growth of CT has not been assessed at all or only assessed to a small degree. The results are very initial due to the limited samples and the qualitative character of the assessments and definitely demand further experimentations (Giannakoulas & Xinogalos, 2018; Kazimoglu et al., 2012; Weintrop & Wilensky, 2012). In addition to the gaps emphasized so far, many educational games aimed at promoting CT, fail to address adaptivity and personalization to the player's needs, and instead follow strict computer-assisted instruction concepts. To summarize, research has shown positive results concerning the implementation of educational games to CT among learners. Nevertheless, improvements can still be made.

Taking into account the important role CT has in society and how relevant it has become in recent years, as well as the highlighted gaps in CT game research, a group of researchers, including Hooshyar et al., (2019) developed AutoThinking, an adaptive educational game, that promotes both students' CT skills, as well as their conceptual knowledge. This study attempts to assess the effectiveness of the AutoThinking game on elementary school students' knowledge of CT by designing and conducting a pretest-posttest experimental design. The experimental group of students are given the AutoThinking game, while the control group learns with traditional technology-enhanced learning approach (conducting a lesson using a PowerPoint presentation with multimedia delivered by a teacher).

In line with this aim, we designed our research questions as follows:

RQ1 - What is the effect of different learning approaches on students overall computational thinking knowledge gain?

RQ2 - What is the effect of different learning approaches on students' computational thinking conceptual knowledge gain?

RQ3 - What is the effect of different learning approaches on students' computational thinking skill gain?

Methodology

AutoThinking

AutoThinking, an intelligent adaptive game, was developed in 2019 by a group of researchers in order to promote students' CT skills and conceptual knowledge (Hooshyar et al. 2019). What makes AutoThinking special, is the fact that it is the first adaptive educational game that incorporates adaptivity in both its game-play, as well as in the learning process. With a specific target of freeing the learner from the shackles of making syntactical errors, thus reducing their cognitive load, AutoThinking employs icons instead of syntax of computer programming languages (See Fig. 1).

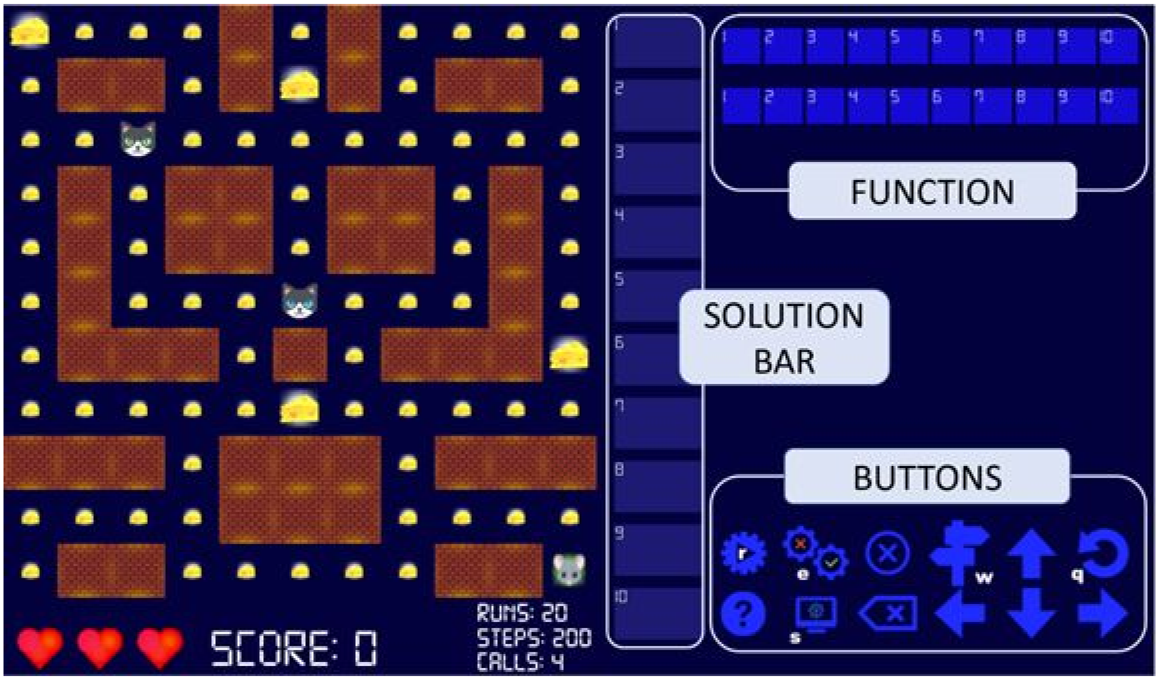

Fig. 1 - interface of AutoThinking

The game itself consists of three levels in which the player needs to apply different methods and solutions in order to win. One wins when they (the mouse) succeed in collecting all the pieces of cheese (76 in total) in the maze while trying to escape from the two cats. The player is given 20 "runs" to complete the task. If a player wants to earn more points, they have the option to do so by creating a run that involves different CT skills or concepts. For example, using the "function" bar offers the player an opportunity to pre-load various patterns, and according to the situation of the maze, even generalize. As mentioned before, the mouse's biggest enemies are two cats, and their movements need to be thoroughly assessed before running a solution. One of the cats (with blue eyes) moves intelligently i.e the same amount of tiles as the mouse, while the other moves randomly, but repetitively, the same amount of tiles as the number of commands the player enters in the "solution" bar. If required, AutoThinking presents the player with numerous sorts of feedback and hints (in the form of text, graphic or video), based on appropriateness of the solution to the state of the maze.

Description of the levels

In the following paragraph we will explain the difference between the three levels of AutoThinking and the CT skills and concepts it promotes. In the first level, which is the easiest as it is meant as introductory level, there is only one cat (green eyed) that moves randomly around the maze the amount of tiles as the number of commands the player enters in the "solution" bar. There are two big cheeses that give extra points and no adaptivity is involved. The player has four unique buttons available - arrows, help, run and delete (See Fig. 2).

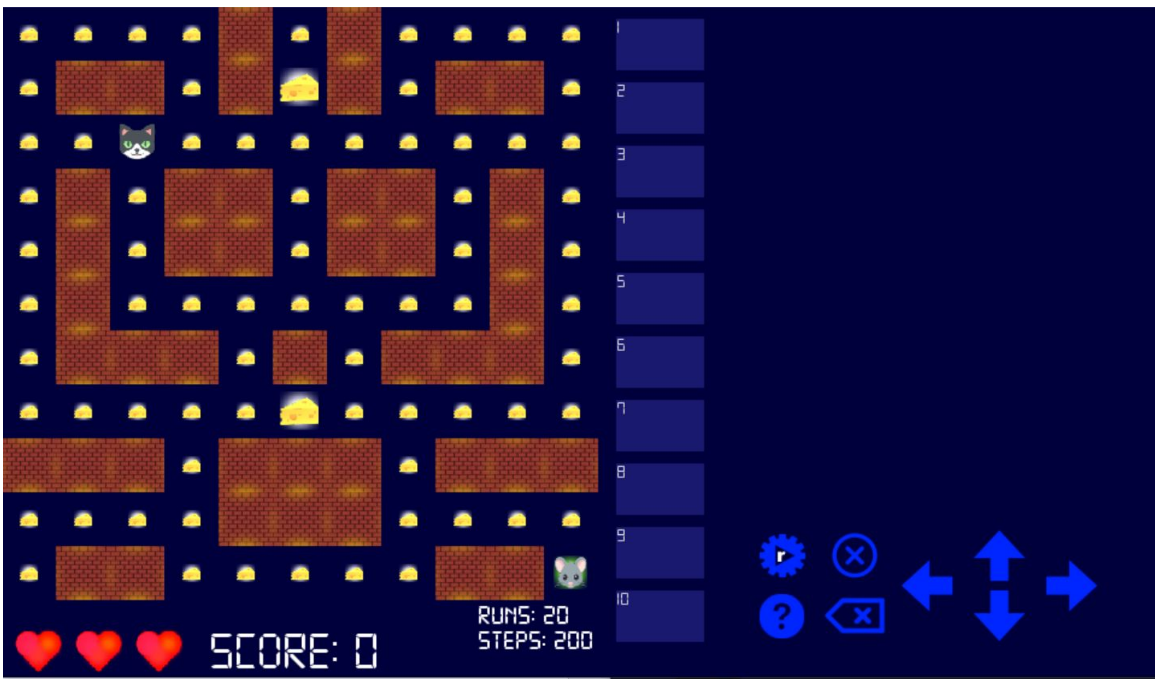

Fig. 2 - AutoThinking level 1

The CT skills level 1 targets are problem identification and decomposition, while the concept targeted is sequence. In level 2, the cat is intelligent (blue eyed) and moves the same amount of tiles as the mouse. There are also two big cheeses that offer extra points. Adaptivity is not involved in this level, but extra buttons are added - conditional, loop and simulation (see Fig. 3). CT skills targeted in level 2 are problem identification and decomposition, as well as simulation. Three CT concepts are targeted: sequence, conditional and loop. In the third, and most difficult level, the player goes against two cats, there are four big cheese out of which two are moving around the maze, and adaptivity is involved. In addition to the buttons from level 2, in this level the developers added a function bar and a debug button (see Fig. 1). CT skills targeted in this level are algorithmic thinking (problem identification and decomposition), algorithm building, debugging and simulation. CT concepts targeted remain sequence, conditional and loop.

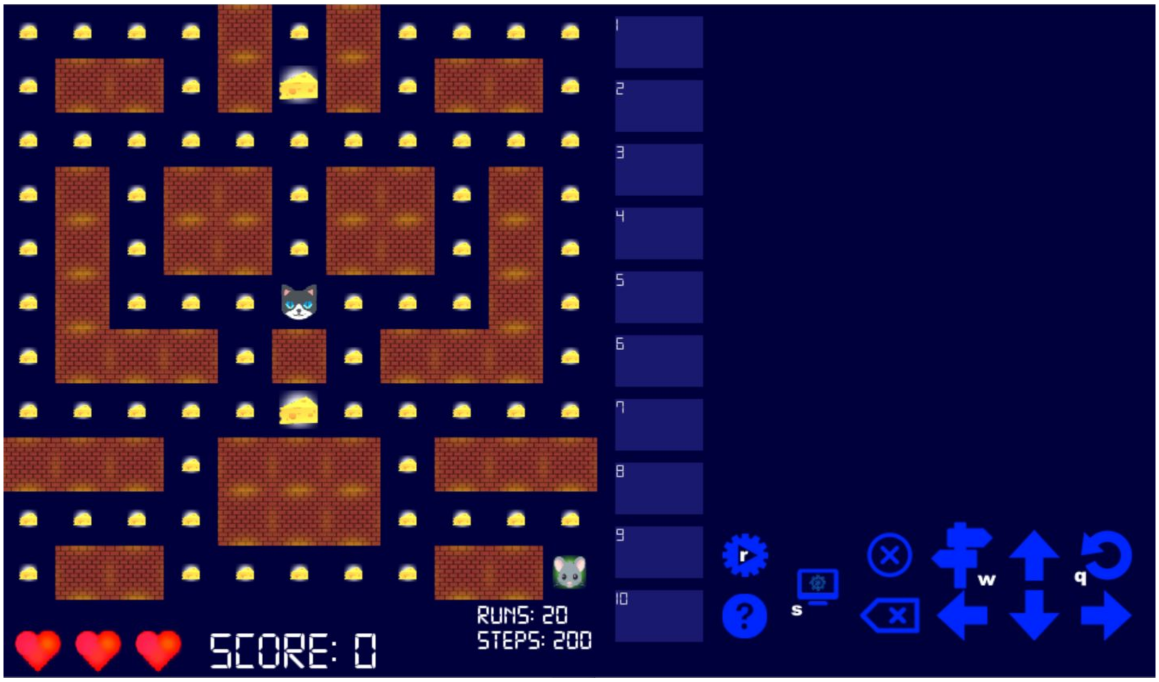

Fig. 3 - AutoThinking level 2

Link between the game and various CT skills and concepts

As mentioned in the previous paragraph, in a special way, AutoThinking fosters numerous CT skills, such as algorithmic thinking (problem identification and decomposition), algorithm building (pattern recognition and generalization), debugging and simulation. It also promotes three CT concepts: sequence, conditional and loop. We will try to present the link between different game activities and these skills and concepts.

CT skills

Algorithmic thinking (problem identification and decomposition). Similar to the very famous PacMan game, by developing solutions, the player helps the mouse eat all the cheese while trying to avoid being eaten by the cats.

Algorithm building (pattern recognition and generalization). The player uses the "function" bar to pre-load solutions that can be applied multiple times.

Debugging. The player has the opportunity to debug, or monitor the entered commands to spot possible errors in its logic.

Simulation. The simulation button gives the player a chance to practice run-time mode by simulating their solution before actually running it. In this mode cats and cheeses movements are not simulated.

CT concepts

Sequence. By entering commands in the "solution" bar, the player creates a sequence of actions or patterns and can see the execution by either pressing the "simulate" or "run" buttons.

Conditional. By making use of the conditional button, the player can make decisions based on certain conditions in the maze. For example if there are few junctions ahead, they can tell the mouse to stop at a certain junction or continue going to a different one.

Loop. This button allows the player to use a certain command or sequence of actions multiple times.

Adaptivity in AutoThinking

Once the player runs a solution, the corresponding log data is sent to the Bayesian Network (BN). The network then decides what is the cat's next move according to that solution and assesses what kind, if any, of feedback or hint to give to the player.

Adaptivity in game-play and in learning

Considering the quality of the solution offered by the player, for the intelligent cat (blue eyed), the game looks at three different states:

- Is the solution too risky? (does the player risk being caught by the cats?)
- Is it a potentially high-scoring solution?
- Were appropriate CT skills and/or concepts used?

According to this, the Bayesian Network automatically evaluates the players' abilities and regulates the intelligent cat's movements in several ways:

1. The cat moves randomly around the maze without any repetition;
2. The cat moves extremely close (up to a single tile) to the mouse, but does not catch it;
3. The cat moves in a hostile manner towards the mouse trying to find the shortest way to catch it;
4. The cat does not get closer than a specific tiles from the mouse

The cat takes into consideration all the moves from the player (from the beginning to the present solution) when switching between the four algorithms. At the same time, the other cat (green eyed) continues moving randomly with iteration around the field in line with the number of commands entered in the solution bar. In this way, AutoThinking repeatedly creates new situations that have not possibly occurred to other players.

With the aim of investigating the vast educational potential of the game, adaptivity in AutoThinking happens in two separate levels, before running the solutions and after. Before running the solution, the player has the opportunity to use the "debug" button in order to spot possible errors in its logic. Accordingly, timely feedback and/or hints are given to the player so they can make any changes, if necessary, to their solution before running it. Feedbacks and hints come in different forms (graphics, video tutorial or simply highlighting certain buttons or commands in the solution bar). After entering a string of commands in the solution bar, the player can run the solution without using the "debug" button. According to the suitability of the solution, the game provides adaptive feedback or hints which can help the player with the upcoming solutions and fixing mistakes made in previous ones. This kind of adaptivity teaches players how to find the optimum solution and at the same time promotes several CT skills and concepts.

Participants

The experiment was conducted in an elementary school in Estonia and the participants were two classes of students aged 11 and 12 years old (fifth grade). The classes were randomly assigned their groups and 17 students were part of the experimental group, while 18 were part of the control group. The experimental group used AutoThinking as a learning approach, while the control group employed a conventional technology-supported learning approach in which a teacher conducted a lesson using PowerPoint presentations together with other multimedia in order to teach CT. In Estonia classes are usually between 15 and 25 students as was the case with our experiment. It should be noted that in order to avoid the influence of different instructors on the experimental results, the control group of students was taught by the same teacher (who had taught them for at least one period). All students were of the knowledge that the participation is completely voluntary and would not affect their grades, and they are allowed to withdraw from the experiment at any time. Both students and their parents were required to sign consent forms to participate in the experiment.

Process

The experiment consisted of three different stages (see Fig. 4). In the first stage, for 30-45 minutes the students answered a pre-test and a pre-questionnaire to measure their CT knowledge and attitude towards CT. In the next 60-75 minutes both groups took part in their respective learning approach stage. The experimental group played the three levels of AutoThinking to learn CT and the teacher's role was to overlook that there aren't any technical issues and just give a basic introduction to the game. The control group had the CT skills and concepts presented by a teacher using a presentation supported by multimedia examples and by eliciting discussion with the class. Both groups covered the concepts of sequence, loop and condition, as well as the algorithmic thinking, algorithmic building, debugging and simulation skills. In the final part of the experiment (30-45 minutes), the students answered post-test and post-questionnaire.

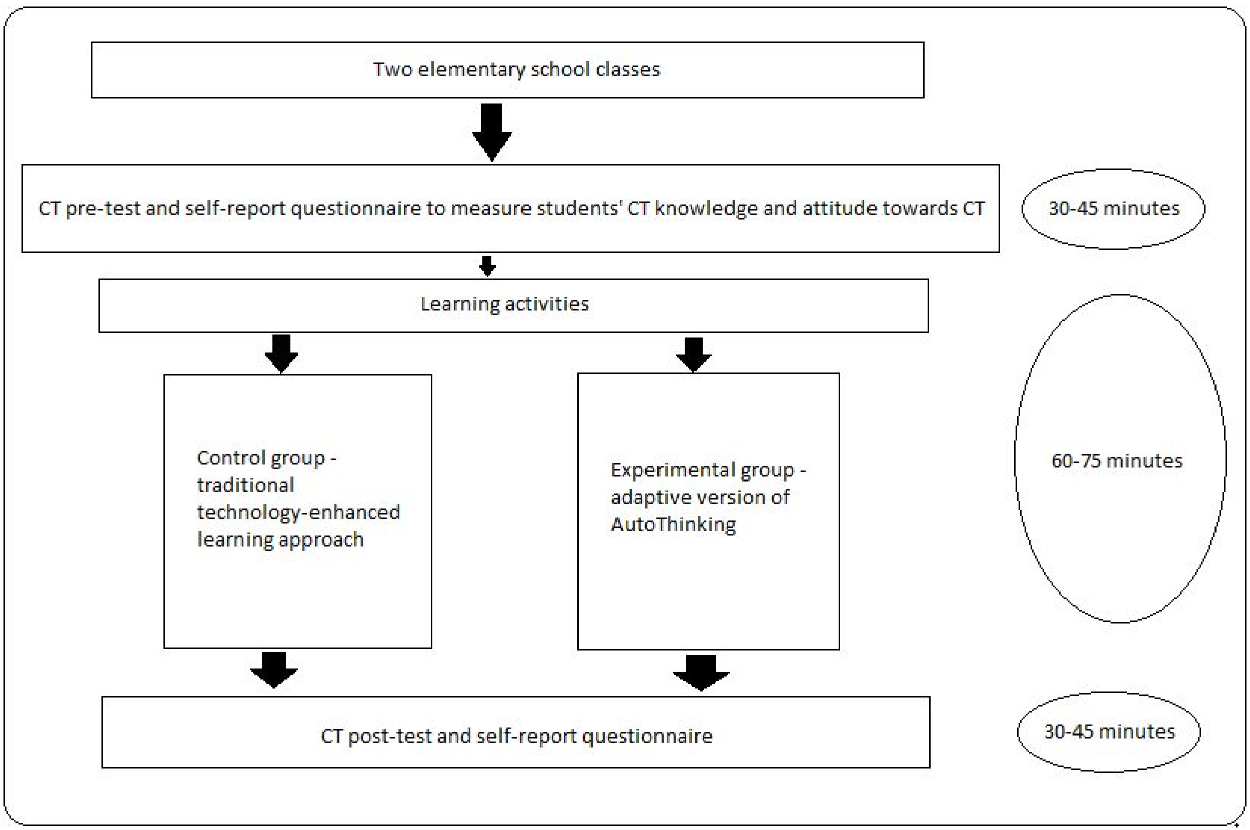

Fig. 4 An illustration of the experimental process

Instruments

For the purposes of the experiment, a pre- and post-test, and pre- and post-questionnaire were used. The tests were used to assess students' CT knowledge (both skills and concepts) before the experiment was conducted, as well as after. These tests consisted of fourteen multiple-choice questions and one fill-in-the-blank question. The questions in the tests were adapted from an approved instrument for assessing CT developed by González (2015) and were analyzed and expanded by two seasoned teachers and researchers with vast experience in teaching and researching in the field of CT and CS. With the aim of finding which skills and concepts had the biggest impact on the results, three groups of items were created (see Table 1 and 2).

What we can see in Table 1, is that there were four items that were targeting all four CT skills at the same time, five of the items were aiming at three skills simultaneously, while six items were targeting two skills. From the table we can see that algorithmic thinking, as well as simulation skills, are involved in every item. Below (Fig. 5) you can see an example of an item in which all four CT skills are targeted at the same time.

Table 1. Skills targeted at different categories of items

	Number of items	CT skills			
		Algorithmic thinking	Pattern recognition	Debugging	Simulation
4-skill	4	+	+	+	+
3-skill	5	+	+	-	+
2-skill	6	+	-	-	+

Table 2. Concepts targeted at different categories of items

	Number of items	CT concepts		
		Sequence	Conditional logic	Loop logic
3-concept	3	+	+	+
2-concept	9	+	-	+
1-concept	3	+	-	-

In the second table (Table 2), which shows us the CT concepts targeted by certain items, we can see that there are three items that target all three concepts concomitantly, nine items aim at two concepts and three items target only one. From this table we can see that sequence is the concept involved in every item. Fig. 6 is an example of an item in which all three concepts are targeted at the same time.

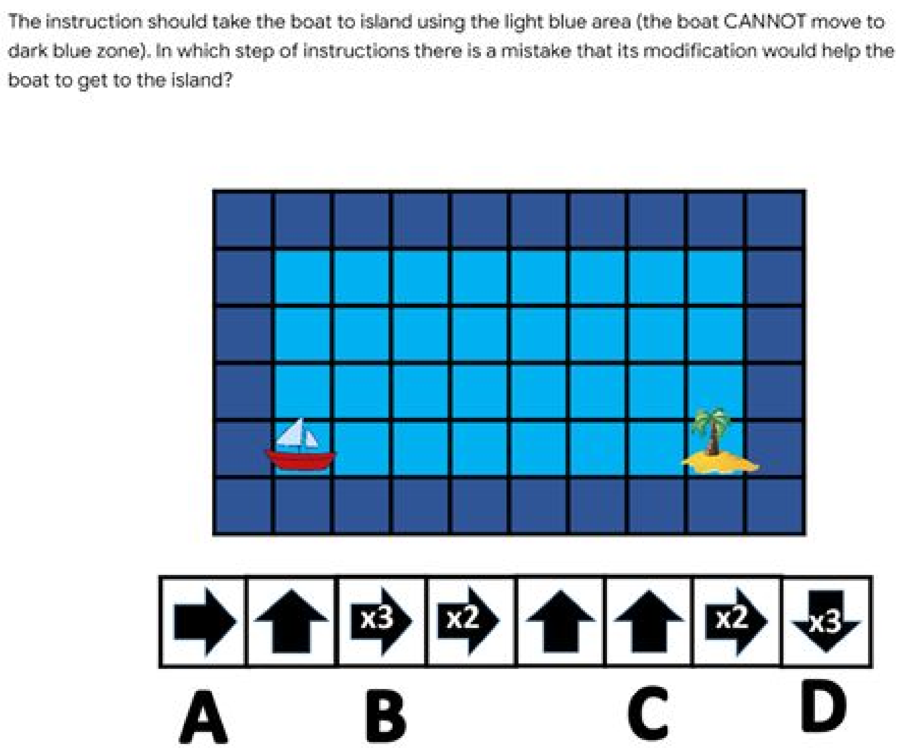

Fig. 5 An example of an item in which all four CT skills are targeted at the same time.

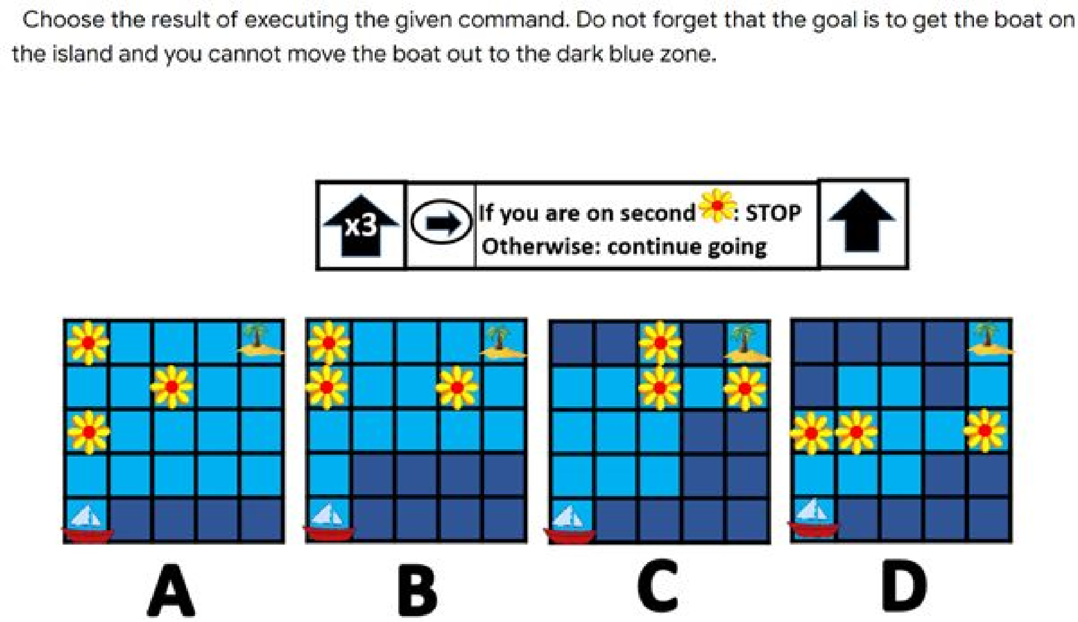

Fig. 6 An example of an item in which all three CT concepts are targeted at the same time.

Results and discussion

RQ1 - What is the effect of different learning approaches on students' overall computational thinking knowledge gain?

In order to explore the effect of different learning approaches with regard to enhancing students' overall CT knowledge, we used the pre- and post-test scores (all fifteen questions in the pre- and post-test). We analyzed data from 35 students utilizing the one-way ANCOVA with the pre-test scores, post-test scores, and the learning approach as covariate, dependent, and independent variable, respectively. One-way ANCOVA was used do to the fact that there are still some slight (insignificant) differences between the pre-test scores of both groups in terms of mean and standard deviation (experimental group: M = 12.70, SD = 1.96; control group: M = 11.05, SD = 2.04) and these kind of differences could still affect the post-test performance. This way we would obtain more precise statistical data to explain the CT knowledge gain of students, since the pre-test scores were excluded. To do so, we first checked the basic assumptions of ANCOVA which is the homogeneity of regression. Hence, we discovered that the assumption of homogeneity of regression is not violated (F=1.76, p = 0.193 > 0.05), indicating that the prior knowledge of both groups is not significantly different.

According to the ANCOVA result (see Table 3), the mean and adjusted mean scores of the experimental group are 12.70 and 12.38, respectively, whereas for the control group, the mean and adjusted mean scores are 11.05 and 11.35, respectively. Moreover, the result reveals that both groups' post-test scores are significantly different when the effect of their pre-test scores were excluded ($F(1,35) = 4.26$, $p = .04$). Talking about the effect size, if we compare the partial ETA squared value with Cohen's guidelines, we can see that the effect size for the teaching approach used is modest. It also indicates that the teaching approach can explain or predict 12% of the dependent variable (in this case students' knowledge gain).

This finding shows that AutoThinking is way more effective in improving students' overall CT knowledge in comparison to the conventional technology-enhanced learning approach.

Table 3. ANCOVA analysis of CT knowledge

Groups	N	Mean	SD	Adjusted Mean	SE	*F*	*Partial ETA Squared*
Experimental group	17	12.70	1.96	12.38	0.35	4.26*	0.12
Control group	18	11.05	2.04	11.35	0.34		

*$p < .05$

RQ2 - What is the effect of different learning approaches on students' computational thinking conceptual knowledge gain?

To investigate the effect of different learning approaches with regard to enhancing students' CT concepts, items were divided into three groups that targeted all three concepts, two concepts or one concept (see section *Instruments*). To this end, data from 35 students (17 in the experimental and 18 in the control group) was analyzed using the one-way ANCOVA with the 3-concept items in pre-test and post-test scores, and the learning approach as covariate, dependent, and independent variable, respectively. Before conducting the test, similar to the previous paragraph, we first checked the basic assumptions of ANCOVA. Consequently, we found that the assumption of homogeneity of regression is not violated (F=0.64, p = 0.43 > 0.05), which is an indication that the prior knowledge of both groups is not significantly different.

It should be noted that rather than running the ANCOVA test on all three different categories, we opted to run only one those items with the highest number of concepts (i.e. three-concept). However, we still report mean and adjusted mean for all three groups to show differences between mean of all groups.

According to the ANCOVA result, see Table 4, both the mean and adjusted mean score of the experimental group for the 3-concept items are 2.35 and 2.24, respectively, whereas the mean and adjusted mean score of the control group for the 3-concept items are 2 and 2.11, respectively. The mean and adjusted mean score of the experimental and control group for the 2-concept items are 7.47 and 7.37, and 6.50 and 6.59, respectively, while the mean and adjusted mean score of the experimental and control group for the 1-concept items are 2.88 and 2.87, and 2.55 and 2.56, respectively. Additionally, the results reveal that for the 3-concept items, both groups' post-test scores DO NOT differ significantly when the effect of their pre-test scores was excluded, ($F(1,35) = .151$, $p = .70$). Talking about the effect size in this case, if we compare the partial ETA squared value with Cohen's guidelines, we can see that the effect size for the teaching approach used is weak. It means that the teaching approach can explain or predict 0.5% of the dependent variable (in this case students' conceptual knowledge gain).

Even though in 3-concept items there are no significant differences between the groups, the adjusted means of both groups reveal that students in the experimental group show a higher improvement than the control group. Furthermore, when it comes to 2 and 1-concept items, we found similar results which showed that the experimental group outperformed the control group in acquiring CT concepts. On the whole, these findings show that AutoThinking game is more effective in boosting students' CT concepts compared to the conventional technology-enhanced learning approach.

Table 4. ANCOVA analysis of CT concepts

Groups	N	Mean	SD	Adjusted Mean	SE	*F*	*Partial ETA Squared*
Experimental group	17	2.35	.861	2.24	0.22	.151	0.005
Control group	18	2	1.3	2.11	0.22		

RQ3 - What is the effect of different learning approaches on students' computational thinking skill gain?

With the aim of exploring the possible effect of different learning approaches on students' computational thinking skill gain, we divided items into three groups: one that targeted all four skills, another one that targeted three skills and last one that targeted two skills (see section *Instruments*). Accordingly, we analyzed data from 35 students (17 in the experimental and 18 in the control group) using the one-way ANCOVA with the 4-skill items in pre-test and post-test scores, and the learning approach as covariate, dependent, and independent variable, respectively. As was the case with the previous two paragraphs, before conducting the test, we first checked the basic assumptions of ANCOVA. To do so, we first checked the basic assumptions of ANCOVA which is the homogeneity of regression. Consequently, we found that the assumption of homogeneity of regression is not violated ($F=.337$, $p = 0.566 > 0.05$), indicating that the prior knowledge of both groups is not significantly different.

It should again be noted that rather than running the ANCOVA test on all three different categories, we opted to run only one those items with the highest number of skills (i.e. four-skills). However, we still report mean and adjusted mean for all three groups to show differences between mean of all groups.

According to the ANCOVA result, see Table 5, both the mean and adjusted mean score of the experimental group for the 4-skill items are 3.35 and 3.28, respectively, whereas the mean and adjusted mean score of the control group for the 4-skill items are 2.72 and 2.79, respectively. The mean and adjusted mean score of the experimental and control group for the 3-skill items are 4.18 and 4.06, and 3.44 and 3.55, respectively, while the mean and adjusted mean score of the experimental and control group for the 2-skill items are 5.18 and 5.08, and 4.89 and 4.97, respectively. Additionally, the results show that for the 4-skill items, both groups' post-test scores are significantly different when the effect of their pre-test scores was excluded, ($F(1,35) = 7.14$, $p = .012$). Talking about the effect size, if we compare the partial ETA squared value with Cohen's guidelines, we can see that the effect size for the teaching approach used is modest. It also shows that the teaching approach can explain or predict 18% of the dependent variable (in this case students' CT skill gain).

Even more, both 2 as well as 3-skill's means and adjusted means are higher for the experimental group than the control group. Overall, these discoveries show again that the AutoThinking game is more effective compared to the conventional technology-enhanced learning approach in improving students' CT skills.

Table 5. ANCOVA analysis of CT skills

Groups	N	Mean	SD	Adjusted Mean	SE	*F*	*Partial ETA Squared*
Experimental group	17	3.35	.49	3.28	0.13	.337	0.18
Control group	18	2.72	.95	2.79	0.13		

Conclusion

In this study we evaluated the effect of different learning approaches on students' overall CT knowledge gain, their conceptual knowledge gain, as well as their CT skill gain. For that we compared the adaptive educational game AutoThinking with conventional technology-enhanced learning (lesson conducted by teacher using PowerPoint and other media). According to our findings, in regards to our first research question, the students who used the AutoThinking game showed considerably better results in gaining overall CT knowledge. Regarding the second question, according to our findings, there is not a significant difference when it comes to 3-concept items, even though the students using AutoThinking did show higher improvement than the control group students. What is more important, the students in the experimental group outperformed their peers from the control group in both 3- and 2-concept items, meaning that AutoThinking showed better results in boosting students' conceptual knowledge gain. In the third research question, we investigated the effect of AutoThinking game on students' CT skills gain and again, students who used the adaptive educational game showed better results than the ones who were exposed to traditional technology-enhanced learning approach. Overall, even though it might not be completely precise to compare our findings to other researches due to different factors (different sample size, different experimental design etc.), it is worth mentioning that our result is aligned with results from other studies on games aiming to foster students' CT knowledge (e.g Zhao and Shute, 2019). One of the main causes for the notable improvement in students' overall CT knowledge could be the game's adaptivity feature during the learning and in-play process which fully supports the player's individual needs. It is one of the first of its kind, and while more research is definitely required, AutoThinking can be used as a stepping stone for further experimentation and development.

Acknowledgments

I would like to thank my thesis supervisor Danial Hooshyar for his guidance, patience and faith in me. Thank you for always being there for me during this year, for showing me the way into science and trust in me when it seemed that I was falling behind.

Thank you to the programme director Emanuele Bardone and all the staff I had the chance to work with, it was a real pleasure and I learned a lot from everyone.

Lastly, and most importantly, thank you to my family, especially my wife, Tiiu, without whose support and encouragement this thesis would not have become reality.

Author's declaration

I hereby declare that I have written this thesis independently and that all contributions of other authors and supporters have been referenced. The thesis has been written in accordance with the requirements for graduation theses of the Institute of Education of the University of Tartu and is in compliance with good academic practices.

Dejan Delev
17.08.2020

List of references

Aho, A. (2012). Computation and Computational Thinking. *The Computer Journal*, *55*, 832–835. https://doi.org/10.1093/comjnl/bxs074

Council, N. R. (2010). *Report of a Workshop on the Scope and Nature of Computational Thinking*. https://doi.org/10.17226/12840

Council, N. R. (2011). *Report of a Workshop on the Pedagogical Aspects of Computational Thinking*. https://doi.org/10.17226/13170

Esper, S., Foster, S. R., Griswold, W. G., Herrera, C., & Snyder, W. (2014). CodeSpells: Bridging educational language features with industry-standard languages. *Proceedings of the 14th Koli Calling International Conference on Computing Education Research*, 05–14. https://doi.org/10.1145/2674683.2674684

Fraser, N. (2012, June 19). Blockly. *Blockly*. https://neil.fraser.name/news/2012/06/19/

Giannakoulas, A., & Xinogalos, S. (2018). 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. https://doi.org/10.1007/s10639-018-9702-x

Gouws, L., Bradshaw, K., & Wentworth, P. (2013). *Computational thinking in educational activities: An evaluation of the educational game Light-Bot*. 10–15. https://doi.org/10.1145/2462476.2466518

Grover, S., & Pea, R. (2013). Computational Thinking in K–12: A Review of the State of the Field. *Educational Researcher*. https://doi.org/10.3102/0013189X12463051

Haddad, R., & Kalaani, Y. (2015). *Can computational thinking predict academic performance?* 225–229. https://doi.org/10.1109/ISECon.2015.7119929

Harvey, B., & Mönig, J. (2010). *Bringing "No Ceiling" to Scratch: Can One Language Serve Kids and Computer Scientists?* 10.

Hooshyar, D., Lim, H., Pedaste, M., Yang, K., Fathi, M., & Yang, Y. (2019). AutoThinking: An Adaptive Computational Thinking Game. In L. Rønningsbakk, T.-T. Wu, F. E. Sandnes, & Y.-M. Huang (Eds.), *Innovative Technologies and Learning* (pp. 381–391). Springer International Publishing. https://doi.org/10.1007/978-3-030-35343-8_41

Hwang, G.-J., Chiu, L.-Y., & Chen, C.-H. (2014). A contextual game-based learning approach to improving students' inquiry-based learning performance in Social Studies courses. *Computers & Education*, *81*. https://doi.org/10.1016/j.compedu.2014.09.006

Kazimoglu, C., Kiernan, M., Bacon, L., & MacKinnon, L. (2011, July 1). *Understanding Computational Thinking before Programming: Developing Guidelines for the Design of Games to Learn Introductory Programming through Game-Play* [Article]. International Journal of Game-Based Learning (IJGBL). www.igi-global.com/article/international-journal-game-based-learning/56313

Kazimoglu, C., Kiernan, M., Bacon, L., & Mackinnon, L. (2012). A Serious Game for Developing Computational Thinking and Learning Introductory Computer Programming. *Procedia - Social and Behavioral Sciences*, *47*, 1991–1999. https://doi.org/10.1016/j.sbspro.2012.06.938

Kelleher, C., & Pausch, R. (n.d.). Lowering the Barriers to Programming: A Taxonomy of Programming Environments and Languages for Novice Programmers. *ACM Computing Surveys*, *37*(2), 55.

Lye, S. Y., & Koh, J. H. L. (2014). Review on teaching and learning of computational thinking through programming: What is next for K-12? *Computers in Human Behavior*, *41*, 51–61. https://doi.org/10.1016/j.chb.2014.09.012

Meerbaum-Salant, O., Armoni, M., & Ben-Ari, M. (2011). *Habits of programming in scratch*. 168–172. https://doi.org/10.1145/1999747.1999796

Papert, S. (1980). *Mindstorms: Children, computers, and powerful ideas*. Basic Books, Inc.

Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., Millner, A., Rosenbaum, E., Silver, J., Silverman, B., & Kafai, Y. (2009). Scratch: Programming for all. *Communications of the ACM*, *52*(11), 60–67. https://doi.org/10.1145/1592761.1592779

Román-González, M. (n.d.). COMPUTATIONAL THINKING TEST: DESIGN GUIDELINES AND CONTENT VALIDATION. *Proceedings of EDULEARN15 Conference*. Retrieved August 7, 2020, from https://www.academia.edu/20221156/COMPUTATIONAL_THINKING_TEST_DESIGN_GUIDELINES_AND_CONTENT_VALIDATION

Selby, C. C., Selby, C., Woollard, J., & Woollard, J. (n.d.). *Computational Thinking: The Developing Definition*. 6.

Weintrop, D., Beheshti, E., Horn, M., Orton, K., Jona, K., Trouille, L., & Wilensky, U. (2016). Defining Computational Thinking for Mathematics and Science Classrooms. *Journal of Science Education and Technology*, *25*(1), 127–147. https://doi.org/10.1007/s10956-015-9581-5

Weintrop, D., & Wilensky, U. (2012). *RoboBuilder: A Program-to-Play Constructionist Video Game*. Proceedings of the Constructionism, Athens.

Wing, J. (2006). Computational Thinking. *Communications of the ACM*, *49*, 33–35. https://doi.org/10.1145/1118178.1118215

Wing, J. M. (n.d.). *Research Notebook: Computational Thinking—What and Why?* 9.

Yardi, S., & Bruckman, A. (2007). What is computing? Bridging the gap between teenagers' perceptions and graduate students' experiences. *Proceedings of the Third International Workshop on Computing Education Research*, 39–50. https://doi.org/10.1145/1288580.1288586

Zhao, W., & Shute, V. J. (2019). Can playing a video game foster computational thinking skills? *Computers & Education*, *141*, 103633. https://doi.org/10.1016/j.compedu.2019.103633

Non-exclusive licence to reproduce thesis and make thesis public

I, Dejan Delev,

1. herewith grant the University of Tartu a free permit (non-exclusive licence) to reproduce, for the purpose of preservation, including for adding to the DSpace digital archives until the expiry of the term of copyright,

"An Experimental Study To Measure The Effect Of Autothinking Game On Primary School Students' Computational Thinking Knowledge",
supervised by Danial Hooshyar (PhD).

2. I grant the University of Tartu a permit to make the work specified in p. 1 available to the public via the web environment of the University of Tartu, including via the DSpace digital archives, under the Creative Commons licence CC BY NC ND 3.0, which allows, by giving appropriate credit to the author, to reproduce, distribute the work and communicate it to the public, and prohibits the creation of derivative works and any commercial use of the work until the expiry of the term of copyright.

3. I am aware of the fact that the author retains the rights specified in p. 1 and 2.

4. I certify that granting the non-exclusive licence does not infringe other persons' intellectual property rights or rights arising from the personal data protection legislation.

Dejan Delev

17/08/2020