Learning and Teaching Programming Skills in Finnish Primary Schools – The Potential of Games
Abstract

Teaching and learning of programming skills is becoming a mandatory part of Finnish primary school curriculum in the fall of 2016. Teaching of these skills will start from the grade one. The decision of starting teaching programming skills for 6-12 year old students is creating a new situation to the field of basic education in Finland. The current situation is creating challenges for teachers, students and for the planning of the new curriculum. It seems that currently there is not much existing official information and material for Finnish class- teachers considering teaching programming skills. In this study I am trying to map out the new situation from the point of view of teachers. The main research question of this study is: What is the role and importance of teaching and learning programming skills in Finnish primary schools?

For this thesis I have reviewed existing literature on the learning and teaching of programming for school children. As a research method I have used qualitative content analysis for analyzing the results of the interviews with teachers. I have mapped out their opinions and suggestions considering the teaching of programming skills, why it is important and how it could be taught. The role of games or game-based environments has been strongly recognized in this study. I have also tried to clarify the position of programming as a learning subject in Finnish basic education. In this study, I am concentrating on learning and teaching of programming skills at grades 1-6 at Finnish primary schools. As results, I will point out that the teaching and learning of programming skills in basic education can be beneficial for society and for the development of children’s cognitive skills, but there are also many challenges involved. In addition, I am introducing two frameworks based on the gathered data.

Keywords

Teaching coding, teaching programming, programming education, learning programming, primary school, elementary school, children, learning theories, constructivism, game based learning, computational thinking, educational games, game design, game-based learning environments

Supervisor

Ph.D., University Lecturer Tonja Molin-Juustila
Foreword

This study process has been very educational. By being educational, I mean in both the proper senses of the word. Besides this study process has educated me some basics of scientific work, it has also introduced the world of Finnish primary school education to me. It was a snowy winter day when I first time walked into a primary-school schoolyard for interviewing teachers. It was a recess-time in the school and as I walked through the crowded yard I heard the loud yelling and laughter of children. I smiled because that indeed is the sound of happiness. The Finnish education system had not managed to suppress that voice. “The future of our nation is in these children. They are the ones in which we have to concentrate”, I thought to myself. Although I have interviewed teachers within this study, along the way I tried not to forget the sound of young learners for often it remains unheard.

I would like to thank all the interviewed teachers who participated in this study. Some of you have already made an extensive career as a teacher and some are just starting to work with pedagogy. Many of the teachers who have been around longer and are oriented to ICT are experts in their own profession. Some of the teachers also seem to have a life lasting mission to work for improving computing and coding skills for children. Because of you, there are also bits and pieces of real wisdom in this study.

In addition, I would like to thank my supervisor, University Lecturer Tonja Molin-Juustila for experienced and professional guidance during the relatively short process in which this thesis had to be finished. I would also like to express my gratitude to Professor Netta Iivari for reviewing this thesis and giving vital feedback for finalizing it.

Tuomas Hiltunen

Oulu, 14.5.2016
# Contents

Abstract ................................................................................................................................. 2  
Foreword .................................................................................................................................. 3  
Contents .................................................................................................................................. 4  
1. Introduction .......................................................................................................................... 5  
2. Learning and Teaching Programming Skills ......................................................................... 8  
   2.1 Importance of Programming Skills .................................................................................. 9  
   2.2 Practises of Learning and Teaching Programming Skills ................................................ 10  
   2.3 Games and Learning Programming ................................................................................ 12  
   2.4 Learning Theories Related to Learning of Programming .................................................. 20  
      2.4.1 Constructivism ........................................................................................................ 20  
      2.4.2 Game Based Learning ............................................................................................. 21  
   2.5 Position of Programming Subject in the Curriculum ...................................................... 23  
3. Methods .............................................................................................................................. 25  
   3.1 Interviews .................................................................................................................... 25  
   3.2 Qualitative Content Analysis ......................................................................................... 28  
4. Findings ............................................................................................................................... 30  
   4.1 The Importance of Teaching Programming Skills ............................................................ 34  
   4.2 Methods and Challenges of Teaching Programming skills .............................................. 36  
   4.3 Game-based Environments and Teaching ....................................................................... 40  
   4.4 Programming and Finnish Curriculum .......................................................................... 43  
   4.5 Learning Theories in Teaching ...................................................................................... 45  
5. Results and Discussion ......................................................................................................... 47  
   5.1 Theoretical Implications ............................................................................................... 48  
   5.2 Practical Implications .................................................................................................... 50  
      5.2.1 Framework for Integrating Programming to other Subjects .................................. 52  
      5.2.2 Framework for Designing an Educational game ...................................................... 53  
6. Conclusion ........................................................................................................................... 55  
   6.1 Limitations .................................................................................................................... 55  
   6.2 Future Research ............................................................................................................. 56  
References .................................................................................................................................. 57  
Appendix A. Structure of the Interviews in Finnish ................................................................. 61  
Appendix B. Structure of the Interviews in English ................................................................. 62
1. Introduction

There are various reasons why learning to interact with computer technology and learning to use software and functions is important. As computers are domesticated in the everyday lives of people, there seems to be an urgent need for childhood education considering computer science. This early education would deepen learners' understanding in several fields related to computer science. One of the most important challenges would be developing programming skills of young children. Fortunately, this has been recognised by decision makers.

In the latest available official curriculum of Finnish basic education for primary schools 2014, learning of programming skills has been taken into consideration. However, when we are looking at the curriculum for the grades 1-3, the learning of programming skills at Finnish primary schools is not described very precisely. There is not much information available on how, why and with what kind of tools it would be implemented. According to Finnish primary-school curriculum (2014), for the grades 1-3, the aim of teaching information and communication technology (ICT) skills is to utilize pupils’ accumulated ICT-skills and knowledge. School work related to acquiring coding skills at lower grades is intelligibly based mainly on child’s play. The basic ICT skills like information retrieval are practiced and learned to use as learning tools. Pupils are also learning to evaluate the significance of information and communication technology. In schools the usage of devices, software and services are practiced as well as using of a keyboard and typing. Pupils are sharing their experiences with digital media and also the computer programming “suitable for their age”. However, the methodology and tools for teaching have not been defined or even mentioned in the curriculum, so the instructions are at quite general level on the 2014 paper. It has been mentioned though that “gamification is used for contributing the learning”. (Opetushallitus, 2014.)

When we are examining the official curriculum of Finnish primary schools (2014) further, we notice that programming skills are, after grades 1-3, mentioned next when the goals of teaching mathematics for grades 3-6 are discussed. One goal of teaching would be to encourage the pupils to create instructions as computer programs using graphical programming environment and later to teach pupils to plan and implement computer programs by using a graphical interface. In the curriculum, besides the programming is tied to mathematics, it is also tied to a children’s general skills of thinking with attributes like learning systematic and causal connections. It is said in the 6th grade mathematics criteria of evaluation. "A student is able to code a functional computer program by using a graphical interface." Algorithms are not discussed until in the evaluation criteria for mathematics for grades 7-9. At the description of wide-based knowledge for grades 7-9, it has been said interestingly: "Programming skills are practiced as a part of different school subjects." This sentence can be seen as a prelude to fast progression in the development of primary school curriculum in Finland concerning teaching programming skills. (Opetushallitus, 2014.)

In upcoming curriculum for 2016, the teaching of programming skills is becoming a mandatory part of Finnish basic education and teaching of these skills will start from the grade one. In this study I am concentrating on the teaching and learning of programming skills at grades 1-6 in Finnish primary schools. The European Code Week (European Comission, 2015) is EU Commission initiative, which aims to increase knowledge of programming in general. Code Week is part of European Comission’s digital agenda for Europe, which encourages the adoption of teaching programming skills in primary
schools and even earlier in childhood education. Related to this, actions to implement programming education have been made also in Finland: in collaboration with the Ministry of Transportation and Communications, the Board of Education in Finland have decided that teaching of computer programming will be taken place at the primary schools in the fall of 2016. Apparently, the programming does not become its own subject but the time for teaching will be taken from hour division of mathematics. Ordinary teachers will begin to teach programming skills in all Finnish classes from the first grade to ninth grade. In their guide for teachers, Liukas and Mykkänen (2014) have identified similarities between programming skills or computational thinking and general problem solving skills or thinking skills in general. (European Comission, 2015; Liukas & Mykkänen, 2014.)

I have personally been rather enthusiastic with computer games in my youth and as well I belong to the generation, where it has been possible to see the development of computer gaming from Commodore Vic-20 to these days. It is also pleasant to notice the change in how games are related and what kind of attitudes people are taking towards the computer gaming today. Gaming has been quite often considered as a waste of time in my childhood. Nothing good could come into existence by playing computer games. Therefore I find the topic of this thesis personally interesting. It suggests that people’s attitudes towards gaming have been taking quite the opposite direction or at least are starting to do it. People have started to notice that learning through games is possible and above that it can also be fun. Game based learning (GBL) for instance is slowly becoming generally an approved concept and its use is encouraged more also with teaching programming skills to children. (Egenfeldt-Nielsen, 2010; Becker, 2007.)

Motivation for this thesis originates also from my earlier experiences with the school-children and use of ubiquitous computing in the City of Oulu, Finland. During those experiences, I have made observations in comprehensive schools where children were in interaction with computers and computing environments. Also an actual decision in European educational politics (European Comission, 2015) and some personal studies in the field of educational science and in information processing encouraged me to write my thesis on the topic.

In this study I am trying to find out why and how are programming skills taught to children. I am also interested in possible new viewpoints among teachers that raise from these quite rigorous plans to teach coding skills for every child. I have studied the existing literature of the topic. Because the learning of programming skills is coming to every school and every teacher is required to start the teaching of these skills, it is legitimate to interview teachers about the matter. I have conducted 8 interviews with Finnish teachers and analyzed the results qualitatively. The intention of this study is to find out the reasons and methods for teaching computer programming to children. I also want to find out what kind of challenges does teaching programming skills to primary school children in Finland cause from the didactic (theoretical and practical application of teaching) point of view, and from learner’s aspect. In addition, I am trying to map out the position of teaching programming at Finnish primary school curriculum. The main research question and sub questions (RQ1-RQ5) can be defined accordingly:

What is the role and importance of teaching and learning programming skills in Finnish primary school?

RQ1: Why are programming skills taught for young children?

RQ2: How are programming skills taught for young children?
RQ3: What is the role of game-based environments in learning and teaching programming skills for children?

RQ4: What is the position of programming in Finnish primary schools?

RQ5: What are the challenges of teaching programming skills?

Next, in Chapter 2, I will review the found literature of the topic. I will first discuss the importance of teaching programming. Secondly I will discuss the practices of teaching programming skills to children. Third, I will illustrate the role of games in teaching programming skills. Fourth, I will introduce two main theories raised from the literature. Then in Chapter 3, I will introduce the research methods used in this study. In Chapter 4, I will analyze the gathered data. In Chapter 5, I will introduce the results of this study and discuss them. I will also introduce the theoretical and practical implications of this study. Finally, in Chapter 6 I will conclude this thesis with limitations and suggestions for the future research.
2. Learning and Teaching Programming Skills

In this chapter, I am discussing learning and teaching of programming or coding skills in previous research. As a data collection method for this literature, I reviewed articles from two major journals: Computers & Education and Computers in Human Behavior. The articles were acquired from ScienceDirect database. Both journals are relevant to both educational science and information technology or computer science. The additional criteria of choosing articles were that children had to be under 12 years old and journal volumes or issues are all published between 2008 and 2015. While reviewing, I descended from the newest issue towards older ones to include the newest possible articles in this study. From the articles I chose qualitatively several core articles, which suited the best for the topic. In addition, also various other searches and databases were used to get the information on the topic. Probably the hardest and the most important criteria were to find articles that are suitable for the right age-group, include the use of game-based environment and are discussing precisely about learning programming, not about learning mathematics or other school subjects.

In the context of primary education for children, the definition of programming is not self-evident. I will first introduce a popular way of explaining it to children. Computers are systems that require input to process the data. After processing the data, computer outputs the inputs. Outputs are processed inputs that are providing some sort of information. The input is provided for the computer by giving commands or instructions to it. That is programming. Programming can be called a form of problem-solving, where writing the code is just one part. (Wikijunior, 2016.) According to Fessakis, Gouli and Mavroudi (2013), programming requires high order thinking and algorithmic problem solving skills. With the word programming, the word coding is often used as a synonym. Coding is often used when the skill to write code is needed. Coding can be described as an implementation part of programming. According to Duncan, Bell and Tanimoto (2014), the term coding is popular, so it would be beneficial to use it instead of programming. When programming with children is discussed in previous research, children have been able to follow the guided instructions, but solving unguided challenges have required more specific instructions. Programming requires a sequence of actions planned ahead. Otherwise we are talking about direct manipulation, which is not considered as programming. (Duncan, Bell & Tanimoto, 2014.) In this study, I am using the word programming in its wider definition that includes logical problem-solving and high order thinking, but also more basic logical thinking. With children it is reasonable to use it also with direct manipulation for example when making animations with Scratch-computational learning environment. Children often start the learning activity with direct manipulation. Eventually, the goal is usually to get children to make the sequences of actions and functionalities that require logical thinking to accomplish. In this study, I will use the word “programming” from the starting point of the learning activity aimed to teach programming skills. In another words from the direct manipulation to the sequences of planned actions. The concept of computational thinking is defined in the next section (2.1).

First I will discuss the importance and practices of teaching programming skills. Secondly, I am reviewing the role of games and game-based environments in the teaching of these skills with some examples. Thirdly, I will present constructivism and game based learning as the most referred theoretical background of the reviewed literature. Finally, I am searching for the information on what would be the positioning of programming as a learning subject based on reviewed articles.
2.1 Importance of Programming Skills

In this paragraph, I am first using two references to non-scientific publications to demonstrate the arguments that are easily found in internet. Instances that are advocating the message on why children should learn to code as early as possible in childhood seem to have many arguments in favor for applying the new curriculum so that coding is taught to every child. They are stating that in the world of search engines the factual knowledge is also easily available so the information retrieval skills of children are becoming more and more important (Four reasons why kids should learn to program, 2016). Programming skills can be seen as a natural continuum for these skills. Developers of Tynker, educational programming platform are answering the question about why children should learn computer programming in their website accordingly: 1. Programming is a basic literacy in the digital age. 2. Programming can change the world. 3. “You have an idea for the next big innovation? Great. Can you bring it to life?” 4. Programming doesn’t have to be hard to learn. In his article, Wing-Kostner (2012) is pointing out the positive financial side learning of programming. The idea that has been expressed in the form of code, can potentially generate value if the idea behind it is unique.

In the previous research, the importance of teaching programming skills to young children stands out. According to Kam, Agarwal, Kumar, Lal, Mathur, Tewari and Canny (2008), the word ‘value’ can also be measured with something different than just financial benefit. One can come up with the code that regulates for example irrigation systems at desiccated locations. In that kind of case we can be talking about the value of saving human lives. Even without generating revolutionary functions with code, by learning programming at young age, people can communicate with professionals and use and understand the possibilities and limits of software better. (Kam, Agarwal, Kumar, Lal, Mathur, Tewari & Canny, 2008.)

According to Kalelioglu (2015), because children are born into a technological world where new technological skills must be adapted fast, there is a demand for changing education programmes to train computer programming for young learners. Arguments that are in favor of teaching programming skills to young children can easily be found on the internet (Duncan, Bell and Tanimoto, 2014). They are pointing out that these arguments are often “emotionally charged” and “anecdote-laden”. They are aiming to bring research based knowledge to discussion because it is not a scientific fact that involving for example 8-year old and younger children to work with computers and coding is beneficial or good for children’s development. However, they are concluding their article with the statement: “Teaching programming skills to younger students is clearly possible and it has benefits.” According to the Duncan and colleague (2014), the challenges of teaching coding to young children can be found for example from learning tools used, context, teacher training and confidence, culture and specific skills taught and engagement level of an initial learning environment (or game-based environment). There are some doubts presented that don’t necessarily support the idea of teaching coding to young children and therefore it is important to define the meaning of coding and try to answer the question what coding is? Nevertheless it is almost certain that today’s school children will be interacting with technology throughout their working lives, regardless of what career paths they choose to follow, so having an understanding of technology and its computational limitations will serve them well. (Duncan, Bell and Tanimoto, 2014.)

This perspective requires viewing the wider aspect of coding in form of developer of cognitive skills, not just strictly the algorithmic side. Cognitive skills are defined shortly by Miller & Wallis (2009) accordingly. "Cognitive control refers to the ability to coordinate thought and action and direct it toward obtaining goals.” (Miller & Wallis,
Children are not necessarily meant to be indoctrinated to “code-slaves” to multinational corporations nor that should be the agenda. A better agenda would be to develop mathematical and geometrical knowledge of children using coding as a tool, and moreover to develop the reflective thinking of children towards the better logical or problem solving skills. (Fessakis, Gouli and Mavroudi, 2013; Duncan, Bell and Tanimoto, 2014; Kalelioglu, 2015.)

Also socio-economic reasons why learning of programming is important seem to come up from reviewed articles. Kuljis & Baldwin (2000) are stating that the demand for competent programmers has risen dramatically in the past years. Participation in computing in general has become important and when increasing this participation, programming is an efficient way (Maloney et. al., 2008). However, according to Duncan et al. 2014), “The value of teaching programming to young students is generally supported for one of two reasons: enabling students to understand what programming is all about, and the general value of computational thinking (CT) which will be of use regardless of a student’s career.” (Duncan, Bell & Tanimoto, 2014)

Wing, 2014 has defined the computational thinking accordingly: “Computational thinking is the thought processes involved in formulating a problem and expressing its solution(s) in such a way that a computer—human or machine—can effectively carry out.” (Wing, 2014) Computational thinking is probably mentioned the first time in the book Mindstorms-Children, Computers and Powerful Ideas by Papert (1980). In his book, the word computational thinking is mentioned when Papert is evaluating a computer hobbyist clubs that are teaching LOGO- learning environment to children. “Their computers simply did not have the power needed for the most engaging and shareable kinds of activities. Their visions of how to integrate computational thinking into everyday life was insufficiently developed. I have no doubt that in the next few years we shall see the formation of some computational environments that deserve to be called samba schools for computation.” (Papert, 1980)

In the next two chapters, I will demonstrate that computational environments (Papert, 1980) have evolved in 36 years. In this study, I am calling them game-based learning environments. In this study, there are digital environments like Scratch involved that are defined as a visual programming language by the developers. However, using Scratch as a game-based learning tool is possible and it is one of its main purposes (Wu & Chang, 2010). The variety of computational environments used in the context of teaching is wide. From the angle of this study, the learning logic of programming in computational environment is the most important factor in them. According to Lodaya (2013), game-based learning environment can be defined quite broadly. If the digital environment is offering a chance to play or design a game in a challenging way while learning, it can be called a game-based learning environment. (Lodaya, 2013.)

2.2 Practises of Learning and Teaching Programming Skills

According to Finnish Koodi 2016 instruction (Liukas & Mykkänen, 2014) programming is usually without an argument taught using game or game-based environment as a tool or vehicle for learning. Very early first coding lessons should be implemented using a game as a tool. These very early games that aim to teach programming skills are often teaching logical thinking to children. However, the games used in classes for grades 1-2 are not necessarily computer games: Teaching the appropriate programming models or logical thinking does not always require a computer. Basic concepts and practices can be learned through play. In most of the programming teaching games. The child will learn to understand the way in which the computer is instructed in order to solve problems. The
game often seeks to ensure that certain activities are carried out in a specific order to solve the task. Teacher should learn to see what games are beneficial and useful considering later education of children. For children, the most important thing is the joy and positive relation to leaning logical thinking and programming. When a child is looking for the shortest route out of the maze, or sorts the recipes or songs alphabetically, he can also learn logical thinking and even programming skills as a byproduct. (Liukas & Mykkänen, 2014.)

While studying 6-8 year old children, the findings of Fessakis and associates (2014) are supporting the idea that children can benefit of the use of game- like computer programming environments in early childhood education. These environments are offering children ways to develop their mathematical, problem solving and social skills. Previously programming skills have been considered to be beneficial to mainly algorithmic problem solving skills but Fessakis and colleagues (2014) are highlighting that learning programming skills can benefit also the improvement of high order thinking in general. High order thinking (Clements & Nastasi, 1999) involves children with judgemental skills, problem solving situations and critical thinking. Especially critical thinking is nowadays considered important skill because of the rising amount of information which is available through internet.

According to Shabanah and Chen (2009), despite the importance of developing higher cognitive thinking (e.g. problem solving by logic- first method), the algorithmic side of programming can’t be underestimated. With older students, data structures and algorithms are important foundation topics in computer science education. Students learn to deal with algorithms in many computer science courses. In addition algorithms can be versatile: “For instance, in computer graphics, students learn objects rendering algorithms, in networking, they study algorithms that solve networks traffic congestion, and in database, they learn algorithms that search or sort data.” (Shabanah, & Chen, 2009.) Nature of programming languages is also changing. For example, Corral and Balcells (2014) have been focusing on object-oriented programming because object-oriented design seems to be natural, since in real life we think in terms of objects, which have certain properties and behaviors.

If we think about the traditional programming education which has been introduced to older generations, the code has been often introduced without any visualization. Yet for young children, abstract concepts can be hard to understand and assimilate. It can be hard to understand what particular line inside code does and what its function is. In teaching situation it is often necessary to visualize abstract concepts so that they are easily understandable by children. If a particular piece of code can be attached to something concrete like an action in the computer screen so that it visualizes directly the activity of code, the understanding of the code itself gets better (Kuljis & Baldwin, 2000). Visualization methods have been developed for a different ways, such as the especially developed programming environments as Scratch (Maloney et. al., 2009).

When after 1-2 grades we are transferring to use more and more computer games and game-based environments, it must be taken in consideration weather to teach coding through syntax or through graphical interface. There are graphical game-based environments like Alice and Scratch and there are games that are teaching coding through syntactical language (GrazyFrog Maze, Codecombat) that has to be written by player from beginning. One of the big questions is how to get student to understand the meaning of sequences of actions that they have to make while learning to code. (Fessakis, Gouli & Mavroudi, 2013.) Duncan, Bell & Tanimoto (2014) have identified challenges of
graphical game-based environments. They are also highlighting the role of teacher in learning process: By understanding thought sequences, beginner programming (for example, with Scratch) is close to direct manipulation and when thinking double clicking command buttons to execute command, it often is direct manipulation. However, there is an option of moving away from direct manipulation as long as the teacher encourages students to do so. So the meaningfulness of practices depend often on teachers’ skills to give additional orders. “A trial-and-error approach to programming, combined with a very simple repertoire of control commands, can lead to what some teachers have referred to as a “Spritefest”, where students simply “program” sprites to move around, and make do with how the program happens to work if they don’t understand how to achieve the effect they were after.” (Duncan, Bell & Tanimoto, 2014.) Even though based on the reviewed literature it remains unanswered when to transfer from graphical interfaces to pure syntax, common sense can be used to a certain point. As learning programming skills are bound to mathematical and linguistic skills of children, it can be interpreted that graphical environments are more suitable for the youngest children. They can also keep the motivation and joy of learning these skills up (Shabanah and Chen, 2009).

Logic first- approach seems to be more effective way of teaching when subjects are under 12 years old and moreover logic via graphical game-based environment. However, a lot of teaching materials are taking quite the opposite approach and teach the syntax, variables and logical operations first without giving them understandable meaning that could be obtainable though logic and computational thinking. (Wing, 2014; Fessakis, Gouli & Mavroudi, 2013.)

2.3 Games and Learning Programming

According to Huang (2014), the game playing process supports the learning process because it allows players to experience learning situations in games. It is also encouraging interactions between players and the game environment and learner can be positioned in complex learning environments. (Huang, 2014.) Because of the similarities in the learning process with game-based learning environments and more traditional games, I am not separating them in this study. I am talking about games and game-based learning environments in the same connection.

Games have been used in the teaching of various subjects. Learning for example English grammar through games can have certain benefits. To name a few advantages of using games for learning, games motivate players (to achieve goals), gives self-confidence (when winning), are fun (through enjoyment and pleasure) and encourages the players’ creativity (to solve the game). (Yip & Kwan, 2006.) It could be interpreted that learning a programming language e.g. C++ through games would have at least a few of these attributes. Also pragmatic educational theories have encouraged the role of action in the learning process for example “Learning by doing” in educational philosophy (Dewey 1957). So learning through games and learning by playing the computer and mobile games seem to be a great way for educating language skills - and possibly also programming skills for children (Kam et. al., 2008).

The research evidence found on reviewed literature supports the view that children are enjoying the engaging learning activities and have opportunities to develop mathematical concepts, problem solving skills and social skills by learning through a game-based environment. Playing is considered “fun”. In context of young, under 12-year-old students, Fessakis and colleagues (2013) are pointing out that because children are players, they also want to create games. While having fun is not typically evaluated high
in teaching context, students who have fun work harder, longer and are more willing to expand their thoughts. (Fessakis, Gouli & Mavroudi, 2013.)

Egenfeldt-Nielsen (2009) has defined three types of learning in educational games area. First, “Learning through Games”, where games are usually developed for educational purposes and only in rare occasions purely for entertainment. In this study Code Combat is an example of this type of game and will be introduced later in more detail. Second, “Learning with Games”, where games are not usually developed for educational purposes but they are used as a tool for learning by for example through assignments given by teachers. Minecraft is a typical game belonging to this category. Third, in “Learning by making Games” category the games are typically game-like development environments that are used to create games or other constructs. As an example of this type of game is Scratch Jr which will be introduced later in this study. With game examples I want to point out that learning programming can be done using games from all of the three categories. (Egenfeldt-Nielsen, 2009.)

Learning through games has also been discussed by Hsieh, Lin and Hou (2015) when studying elementary school children learning of sequence classification concepts (=problem solving). According to them, knowledge is gained through both personal and environmental experience during the game play process. Children learn to develop their own ideas, try them out, and generate concrete concepts based on their experiences. Hsieh et al. (2015), are stating that using game or game-based environment provides a mean to visualize and measure the learning process effectiveness and overall engagement in learning environment. (Hsieh et. al., 2015.) In their article about problem solving by 5–6 years old kindergarten children in a computer programming environment, Fessakis et al. (2013) are presenting an experimental game for children. The game is including the following features: “Simplified syntax, programming with the use of command symbols on tiles that children can drag and drop to compose programs, immediate execution of commands and programming paradigms adapted to metaphors.” While in Scratch, there is a stage where children place sprites, actors and objects each one of which follows its own script. (Fessakis et al. 2013.) According to Kam and associates (2008), one reason for using computer games in learning purposes differently is that because mobile phones and laptops are generalizing in poor regions more rapidly than traditional teaching with teachers and schools. Kam et al., (2008) are stating that because of the increased availability of these devices worldwide would be a good reason to use them also in educational purposes using games as instruments of learning a new language. When learning programming skills with games, Maloney, Peppler, Kafai, Resnick & Rush (2008) have recognized the need to broaden the participation in computing and observed the youth while they were using Scratch (visual, game-like programming language used for learning purposes). They are noting that youth interest in technology starts with digital media, including games, and could also serve as a more promising pathway into programming. “Scratch emphasizes media manipulation and supports programming activities that resonate with the interests of youth, such as creating animated stories, games, and interactive presentations.” (Maloney et. al., 2008.)

In their article, Shabanah and Chen (2009) give reasons, why an algorithm learning and visualization approach using serious games is reasonable. In the background there is knowledge of human ability to realize graphic representations faster than textual representations. Serious games are games that have been developed for learning or other serious purposes like training and simulation, not only for entertainment. However, serious games and proprietary games are sharing some important attributes that can justify game using and developing also for educational purposes.
As the first reason for using computer games in educational purposes Shabanah and Chen (2009) are referring to the general popularity of the games. Young generation is using many hours in playing. That is why we should be teaching them using computer games. Second, they consider the use of computer games important because they are interactive. High engagement level results improved algorithm learning. Computer games are also competitive, which is the third reason. Games themselves motivate players to spend time with them. This is called *Intrinsic Motivation*, which is improving learning. Fourth reason to use computer games is that they simplify assessment. Winning a computer game requires understanding its rules well. By winning or losing, algorithms can be taught to player. Finally, a reason for using games for educational purposes is that games are entertaining. Algorithm learning can be a hard task. A computer game can convert the learning task to be fun and interesting. (Shabanah & Chen, 2009.)

Based on their review of the previous research, Berns, Gonzalez-Bardo & Camacho (2013) are outlining the following reasons for the attractiveness of computer games. First, they are highly immersive because of the use of 3-D technics. Second, games are simulating competition and co-operation. Third, games are providing players with real-time feedback on their failure or success. Fourth, games are encouraging risk taking, exploration and problem solving. Fourth, games are fun and entertaining. Finally, they often have different difficulty levels based on the experience and skills of player. (Berns, Gonzalez-Pardo & Camacho, 2013.)

Shute, Ventura & Ke (2015) have identified a deeper psychological reason for using games in educational purposes. Besides the fact that most children seem to gravitate towards computer gaming in their daily lives, computer gaming has positive effects on visual-spatial skills, attention, openness to experience, persistence, creativity and civic engagement. (Shute et. al., 2015) Through educational games children can learn also be motivated to learn academic content and problem solving skills also in metacognitive context. (Shute et. al., 2015; Fessakis et al., 2013.) According to Flawell (1979), young children are often limited with their knowledge of their own thoughts or their level of know-how. They can’t consciously monitor their skills of learning, memory and level of comprehension. Communication, social cognition and problem solving are among other things sectors that are affected by *metacognition*. (Flawell, 1979.)

Corral and associates (2014) have noticed that difficulties of understanding for example the concepts of object oriented programming, can be overcome by a game-based approach. Abstract concepts can be illustrated better with graphical environment. The persistence, motivation and overall academic results of students were good when using game-based environment. Younger children can develop new skills via games e.g. media storytelling, visual design and audience awareness. (Corral et al., 2014; Robertson, 2012.) Hsieh and colleagues (2015) were observing elementary schools students while they were learning problem solving in general through a game in the classroom. They noticed that knowledge was gained through both personal and environmental experience during the game play process. Children learned to “develop their own ideas, try them out, and generate concrete concepts based on their experiences.” (Hsieh, Lin & Hou, 2015.)

It seems that learning programming skills through games can be intentional or unintentional process for children. According to Maloney, Peppler, Kawai, Resnick & Rush (2008) “Most youth didn’t identify scripting with Scratch as a form of programming in general. When youth were asked: “What is computer programming to you?” They responded “Computer programming?” I don’t have a clue what that is!” According to Maloney et. al., (2008) that was not necessarily a bad thing because teaching programming to youth is not turning them straight away to hackers nor programmers but
the nature of teaching is more of engaging them to technology and guide them to understand the logic behind the structures.

Hsieh et al. (2015) have found evidence that the game can consistently increase students’ engagement in the game-based learning environment. While this seems to be true at general level with elementary school children, there seems to be some differences in the learning styles and strategies of boys and girls: In problem situations, boys are more verbal than girls who tend to execute more non-verbal behaviour like expressions and body-language when confused. Boys show to be more competitive in nature. Female students can be less confident about using technology to execute learning tasks and possibly they need to give themselves more space to think. Indeed, there seems to be some embedded differences inside the young generation of learners who are also sometimes called “digital natives”. According to Prensky (2001), there have been general difficulties in understanding the learning needs of young generations. This originates from the fast development and harnessing of digital technology leading into problematic situation: “Today’s students are no longer the people our educational system was designed to teach.” (Prensky, 2001.)

Haden (2006) has been studying how to teach traditional programming skills for college students “through” game programming (see E-N, 2009, about learning through games) with promising results. Participating in the game programming course was improving students’ programming skills and in addition students thought that the class was enjoyable. The use of self-made tools like “The incredible rainbow spitting chicken” (for learning linked lists) inspired the students. Educators have frequently been concerned that computers are only for gaming, not for calculus, but graphical and interactive power of new development tools can be harnessed to educational purposes. Also Shabanah & Chen (2009) have recognized that algorithms are often hard to teach and learn. As a solution, they are using the game-based visualization approach. Game-based approach for teaching algorithms is using emotions and feels of player or student to motivate the learning process. Desire to win, compete and be entertained are motivating players to learn algorithms. (Shabanah & Chen, 2009.)

On the other hand, there is no magical recipe for teaching algorithms through games successfully. According to Shabanah (2009) researchers have not found general evidence that use of visualization systems would lead to better understanding about algorithms even though children would be interested of using the system. Also algorithm animations can have unwanted effects on students understanding about algorithms. “Studies in which students merely viewed visualizations did not demonstrate significant learning advantages over children who used conventional learning materials.” He founds two reasons for this. First developers often lack insight into pedagogical issues. In the condensed significance of the definition, the pedagogic refers to a teaching skill, growing skill or how the teaching should be arranged. System (e.g. game) developers are focusing more on developing graphics and sound. Secondly students’ interaction with an algorithm visualization system can remain superficial. Students are only observing the system instead of taking actions. As Shabanah is pointing out: “What the students do, not what they see, has the greater impact on learning.” (Shabanah, 2009.)

In the reviewed articles there are many challenges identified from game-developer’s angle. Kam et al., (2008) are noticing that it is challenging to design educational games that are in addition to being educational, also pleasurable. Designing them is hard particularly for children with limited experience and knowhow with technology. (Kam et al., 2008.) Next I will introduce some attributes of educational games presented in the
previous research and their implications. In the end, I will also introduce two popular educational games as the examples of different user interfaces and types of games defined by Egenfeld-Nielsen (2009).

Considering the attributes of educational games, Kiili (2005) is stating that games are in generally designed to generate a positive effect on players. According to Kiili (2005), games are effective to engage a “flow”-state among players. By playing game, the player is approaching the psychological state where he or she is focusing on achieving the goal so hard that surrounding environment seem to fade. Kiili is also highlighting the importance and natural character of problem solving skills among human beings and how the games are offering a good framework for offering problems to players. According to Kiili (2005), three components must be taken into consideration when designing an educational game: a person, task and artefact (see Figure 1). Some of the attributes are task- related and some artefact – related. When designing tasks and artefact, persons experience and skills must be taken in to account to get his attention. Also for example bad usability has negative impact on reaching the flow- state.

![Figure 1. Flow- state through playing a digital game. (Kiili, 2005)](image)

Through games or game-based environments, it is possible to reach flow- experience. From the point of view of this study, maybe the most important flow- consequence would be increased learning through playing a digital game. Kiili (2005) is defining three attributes for effective educational game, immediate feedback, clear goals and challenges that are matched to person’s skill level.

According to Fessakis et. al., (2013), educational games are offering players ways to interact with the game. In their article, they are introducing five types of interaction that can be listed as attributes for good educational games for children. First, *Competition*, which is maintaining the children’s interest and attention. Second, *Interferences* concerning command proposals and instructions. Third, *Collaboration* with teachers and friends when facing difficulties depending on whom the child selects as his/her collaborator. Fourth, *Moral support*, where the player approves or becomes encouraged by the game. Fifth and last type of interaction, *Dialogue* is not only a task related but also about the excitement of a “journey” in game.

Annetta (2010) has defined a framework for serious educational game design. It includes the following concepts: identity, immersion, interactivity, increasing complexity, informed teaching, instructional. The player is given an *identity* and after that, the player
becomes immersed. Then the player is forced to interact with other in-game characters. As the player learns about the game plot, he or she must answer the questions, which become increasingly more complex as the game progresses. The teacher was able to access the clicks and answers to the questions provided by players from the game’s database.” These results informed his teaching practice and allowed him to adjust the game’s scaffolds and develop other instructional activities to align with the learning objectives.” (Annetta, 2010.)

In Table 1, there are some of the most popular games or game-based learning environments available. The table includes also the games and game-like environments revealed by the analysis of interviews (4.3). The table shows the type of input of the game and difficulty level defined by Duncan, Bell & Tanimoto, (2014). The difficulty level is defined by the skill level needed for starting to use a game. It is also indicating how far the difficulty level of the game goes on the scale 0-4 (0 = the easiest, 4= the hardest difficulty level). The type of the game is either a drag & drop or writing the indicated programming language as user input while playing.

**Table 1.** Game-based learning environments for young students. (Duncan, Bell & Tanimoto, 2014)

<table>
<thead>
<tr>
<th>Name of the Game-Environment</th>
<th>Type of Interface</th>
<th>Level (0-4)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bee-Bot, Blue-Bot</td>
<td>Drag &amp; Drop</td>
<td>0</td>
</tr>
<tr>
<td>Scratch Jr</td>
<td>Drag &amp; Drop</td>
<td>0, 1</td>
</tr>
<tr>
<td>Kodable</td>
<td>Drag &amp; Drop</td>
<td>1</td>
</tr>
<tr>
<td>Lightbot</td>
<td>Drag &amp; Drop</td>
<td>1</td>
</tr>
<tr>
<td>Robo Logic</td>
<td>Drag &amp; Drop</td>
<td>1</td>
</tr>
<tr>
<td>Cargo-Bot</td>
<td>Drag &amp; Drop</td>
<td>1, 2</td>
</tr>
<tr>
<td>Kodu</td>
<td>Drag &amp; Drop</td>
<td>1, 2</td>
</tr>
<tr>
<td>Tynker</td>
<td>Drag &amp; Drop</td>
<td>1, 2, 3</td>
</tr>
<tr>
<td>Scratch</td>
<td>Drag &amp; Drop</td>
<td>1, 2, 3</td>
</tr>
<tr>
<td>Alice</td>
<td>Drag &amp; Drop</td>
<td>2, 3</td>
</tr>
<tr>
<td>KidsRuby</td>
<td>Ruby</td>
<td>3</td>
</tr>
<tr>
<td>Code Combat</td>
<td>PHP, Javascript, Python</td>
<td>2, 3, 4</td>
</tr>
<tr>
<td>Codecademy</td>
<td>Python, Ruby, Javascript, HTML, CSS, PHP</td>
<td>3, 4</td>
</tr>
<tr>
<td>Minecraft</td>
<td>Javascript</td>
<td>3, 4</td>
</tr>
</tbody>
</table>
BlueJ | Java | 3, 4
---|---|---
Simduino | Arduino C | 4

In the Table 1 we can see that the type of interface is related to the difficulty level of the learning environment. The environments aimed for younger audience are on the difficulty level 0-3. They have all drag & drop interfaces. When the difficulty level increases (2-4), the interface is starting to become controllable by writing “real” code.

Next I will introduce two examples of educational games and environments, Code Combat and Scratch Jr. They are both designed for teaching programming skills to player, but they are using different game mechanics and interfaces to obtain the goal. I will also discuss some of the differences and similarities between these two games. I introduce these two games because they both are popular and they are both using different player input type revealed in literature review: Code Combat uses ‘pure syntax’- style and Scratch Jr graphical ‘drag & drop’- style.

When considering the three types of learning in educational games defined by Egenfeld-Nielsen (2009), Code Combat could be considered as an example of Learning with Games type of learning while Strach Jr. represents more Learning by making Games- type because in Scratch Jr it is possible for the player to create his own animation with functionalities.

![Figure 2. Code Combat, an educational programming game suitable for 9+ year old children. (CodeCombat, 2016)](image)

In Code Combat, a player is solving in-game problems by writing the right code to the command prompt. Code Combat is originally open source learning IDE (Integrated Development Environment). Different programming languages like PHP, JavaScript and Python can be learned through playing an exciting role playing game. The learning curve is suitable for young learners. The game was originally developed to motivate the game inventor himself, George Steins, to code. (Beshawred, 2014.)
Scratch Jr. is a popular graphical game-like environment among children aged 5-7, where children are able to build their own interactive stories and solve problems with a graphical drag and drop method. (Scratch Jr – Home, 2016)

We can notice similarities and differences between Code Combat and Scratch Jr: in Code Combat the input type of player is writing the right syntactical code in the “intelligent” command line. Coding is “real” coding from the first level. On the other hand with Scratch Jr. player is using a graphical interface with buttons and shortcuts without getting involved with syntax written through a keyboard. In Code Combat, you have to think more about logic first before giving commands. At first levels both of the games are teaching for example “MoveTo”- method for character. The Code Combat is aimed for older players so the difficulty level gets hard faster than in Scratch Jr, which gives more freedom for a player to use his/her creativity in creating the problems. These two games seem to be good examples of how other environments use graphics- first and others logic first- methods (Fessakis et. al., 2013) It seems that Scratch Jr encourages player in using a “Trial & error”- thinking method whereas Code Combat seems to require more “Planning ahead”- thinking method (Fessakis, Gouli, & Mavroudi, 2013).

When considering the reasons for using computer games in educational purposes discussed by Shabanah & Chen (2009), both games are popular, interactive and entertaining but Code Combat seem to be more competitive. It challenges a player to solve problems and there are real win / lose situations so it seems that it is easier to simplify assessment with Code Combat more than with Scratch Jr. The reason for this could be that Scratch Jr is aimed for younger audience. Also immersion and in- game identity (Annetta, 2010) were expressed stronger in Code Combat. In this game the player will always name one’s character which makes his/her identity stronger. The music and audio is also very exciting and has immersive attributes in Code Combat. Both games aim to make the experience of learning programming as enjoyable as possible.
2.4 Learning Theories Related to Learning of Programming

In the literature review, the most often used pedagogic theories related to learning programming were constructivism and game based learning (GBL), which actually integrates several various learning theories. Next, I will shortly introduce the psychological background of learning programming as a child, and then the both learning theories. First in 2.4.1, I will introduce constructivism and then in 2.4.2, I will introduce GBL.

The aim of teaching children programming skills should not only be to teach them understanding of computer science and related fields. Programming environments can assist children to “analyse, organize, express and evaluate” their thoughts while solving a problem (Fessakis et. al., 2013). Opportunities and skills like “articulating their thoughts, watch the outcomes, clarify their thought process and receive immediate feedback” (Clements & Nastasi, 1999) are going beyond just algorithmic problem solving skills or mathematical skills. In their article Social-cognitive behaviors and higher-order thinking in educational computer environments, Clements & Nastasi, 1999 are pointing out that learning computer programming can be beneficial to not only cognitive learning of children but also in learning metacognitive skills. According to Zhang (2001), knowing more about learners’ metacognitive, strategic knowledge could help when teaching them second language skills. In his study, students’ second language reading ability and their second language proficiency levels interacted with their metacognitive awareness. According to the research results of Fessakis et. al., (2013), children’s problem solving method seem to be often trial and error- method. They are solving the problem step by step. Some of the children however use planning ahead- method. They are planning 1-3 steps or programming commands ahead before implementing the function. Some children can plan the whole problem solving procedure ahead before implementing it and this requires advanced metacognitive skills. The evidence that some of the children already have this skill and others have it partially suggests that learning problem solving skills through games truly is beneficial. (Fessakis et. al., 2013.)

2.4.1 Constructivism

Constructivist learning is founded on learners’ active participation in problem solving and critical thinking towards relevant learning activity. Learners are "constructing" their own knowledge by trying out approaches based on their earlier knowledge. They try to integrate their new ideas to prior knowledge. (Kanselaar, G., 2002.)

Boyer, Langevin, & Gaspar (2008) and Duncan, Bell & Tanimoto (2014) have noticed the connection between educational theory (constructivism) and programming. They are writing about the relationship between new constructivist apprenticeship techniques meant to improve programming pedagogy and student self-direction. They are highlighting that the nature of programming is creative and requires thinking outside the box, find alternative answers to their own in problem solving situations. This is seen as a perquisite for becoming a genuine programmer with higher cognitive skills. Fessakis, Gouli & Mavroudi (2013) identify the same beneficial connection of computer programming and higher order thinking besides just improvement in algorithmic problem solving skills. Programming education should also be meaningful compared with the competence and developmental level of learners. Haden (2006) is pointing out that if the first experience in instructional programming course is enjoyable and effective, success in later programming courses can be predicted. What it comes to the competence of children, Rushkoff (2010) has defined the dilemma accurately: “We teach kids how to use
software to write, but not how to write software. This means they have access to the capabilities given to them by others, but not the power to determine the value-creating capabilities of these technologies for themselves." (Rushkoff, 2010.)

Dalgerno (1996) have discussed about Vygotsky’s view of knowledge. In constructivism there are various models of knowledge representations that individuals construct. These representations can be equally correct. The main focus of teaching is more guiding the learner to build his/her existing mental models. The knowledge is constructed rather than transferred from teacher to learner. (Dalgero, 1996.) Majority of people have equal possibilities for knowledge construction via social interaction. According to Liu & Matthews (2005), human beings are always searching for the truth, in another words people are curious. The scientific exploration of the truth must be obtained via development of human rationality. (Liu & Matthews, 2005.) Epistemic relativism in Vygotsky’s case can be said to emerge from the metaphysical view that knowledge is a social construction. (Boghossian, 2006.)

When we are thinking about constructivism from pedagogic viewpoint and also we are interested in learning environments, Kanselaar (2002) has proposed that there are eight characteristics that can be used to define constructivist learning environments:

1. They provide multiple representations of reality.

2. Multiple representations avoid oversimplification and represent the complexity of the real world.

3. They emphasize knowledge construction instead of knowledge reproduction.

4. They emphasize authentic tasks in a meaningful context rather than abstract instruction out of context.

5. They provide learning environments such as real world settings or case based learning instead of predetermined sequences of instruction.

6. They encourage thoughtful reflection on experience.

7. They enable context and content-dependent knowledge construction.

8. They support collaborative construction of knowledge through social negotiation, not competition among learners for recognition. (Kanselaar, 2002.)

In his article, Kanselaar, (2002) is also highlighting that in education area besides individual, the social perspective is very important. There is a term socio-constructivism for describing this social approach of constructivism.

2.4.2 Game Based Learning

Because of the increasing use of technology in the learning process, the need of engaging students to subject has become important matter. This is why the concept of Game-based Learning has proved its effectiveness. The picture 3 illustrates the learning cycle in GBL. The cycle is a loop for as long as the game levels are renewing. After completing each level, player is seeking new challenges and trying to overcome them through action. Rehearsal is needed in order to make advance in the game, gain the new level and in order to evolve.
The continuous interest in game-based learning and mounting evidence for its efficiency suggest that there is a real potential in Game based learning (GBL). (Ito, 2008; Egenfeldt-Nielsen, 2010) GBL is referring of playing the type of games that has often beforehand specified learning outcome. Besides the games are widely used to attract children to learning activities and also to social interactions through multiplaying and sharing for instance. Prensky (2003) have found evidence that GBL has also physiological impacts. Our digital native children’s brains are changing to accommodate the new technologies with which they spend so much time with. (Prensky, 2003.)

Egenfeldt-Nielsen (2010) is identifying a need of improvement considering GBL and formal education. They are stating that the game-based learning should be an integrated part of the formal education. The games and game-based learning environments are available in Finland but only a certain small group of teachers with “mission” are using them in everyday teaching and are trying to integrate the games in the curriculum so that they would also be present in the teaching of other subjects than mathematics, for instance.

Egenfeldt-Nielsen, S. (2010) has also recognized that game-based learning is potentially facing a lot of challenges. The attributes for game-based learning are not taking the new innovations into account. Game-based learning seem to fit badly with current educational system. It is not clear, how it would be adapted to it. (Egenfeldt-Nielsen, 2010.) Becker (2007) notices an important didactical aspect in applying GBL among schools. By didactical he means that it is related to the art of teaching teachers how to teach. If we except teachers to use games as a tool for teaching, we have to give them a possibility to play those games themselves. According to Becker (2007), they should be encouraged to play games with critical eye. Teachers have to have skills to evaluate games and make the decision which of them are appropriate. These skills should be taught to new teachers during their education or through “pre-service teacher preparation programs”. The overall success of applying GBL will depend on how well teachers can take advantage of digital games. (Becker, 2007.)

It seems that depending (among other things) on the role of tutor person, game based learning situation has many constructivist- attributes. A constructivist learning opportunity or lesson can simulate the learner to solve a problem and then to reflect on what has happened in the prior knowledge. “In this sense, learning a programming language can be thought as a construction process, where the student gradually builds new knowledge structures that connects with his prior knowledge.” (Corral et. al., 2014) In order to solve a problem, the analogy of an abstract concept must be found from the real world, or in game based context from graphically simulated ‘real world’. (Corral et. al., 2014). According to Boyer, Langevin, & Gaspar (2008), there is a difference in teaching students to program and to teach them about programming. Problem solving
skills are developed better with the use of constructivist apprenticeship which leads to developing self-direction in problem solving skills. Fessakis et. al., (2013) are defining these skills as metacognitive skills.

### 2.5 Position of Programming Subject in the Curriculum

Next I will discuss previous research from the point of view how programming has been positioned as a learning subject. I am pointing out analogies between coding and e.g. mathematics, spoken language and music that could help the understanding about the nature of teaching programming skills to young children at schools.

Shieber (1984) writes about designing of the computer language for linguistic information by designing tools to facilitate communication of linguistic information to computers. In 1984 their aim was to encode linguistic information to computers. Computer languages are a direct result of this need for effective communication with computers. Programming language evolves from denotational semantics to linguistic constructs. For example he is adding linguistic words within computer code and calls it “syntactic sugar”. According to Shieber (1984) that allows programming language to conform to a user’s intentions. Shieber (1984) was finding answer for the current interest (1984) in computer science in declarative languages. He wanted to introduce clear semantics designed specifically for encoding linguistic information.

At present (2016) high level programming languages have evolved and deployed. A High level programming language may use natural language elements. The function has been developed simply to ease the user of the language. High level abstraction from machine language is typical feature for these languages e.g. Python, PhP and Visual Basic. (Feeley, 2015.)

Sparks & Ganscow (1993) point out that phonology, syntax, and semantic are the areas that are causing the problems in second language learning. Especially if you are a poor reader, you have difficulties with phonology and syntax. In linguistics syntax governs the structure of language. There are also syntactic rules that are common to all spoken languages like the conceptions of subject and object (Branigan, Pickering & Leland, 2000). In programming languages, the term “syntax” is including the vocabulary and grammar of the programming language. The logical meaning of syntax is called semantics. Semantics is trying to evaluate or give description to the syntactically correct sentences.

While we think about phonology as a linguistic term, which studies systematic organization of sounds in languages, and as we are trying to chase the similarities that would possibly connect linguistic code to programming code, we can give a brief thought for music. One of the characteristics of traditional music notation is that it is made humanly performable (Hudak et. al.,1996). On the other hand programming code is made computer performable but also computer code is made human understandable. Music notation is markup language for “musical code”. In present researchers are programming computers to understand musical code, even to produce algorithm based music by themselves. When trying to make a computer to “understand” music, after describing, transcribing and notating you have to digitalize. In digitalization or in AD/DA conversion lies connection to for example Hoare’s axiomatic basis for computer programming. (Hoare, 1969.)

The global progression in networking and information technologies has led to situation where information technology has become not only a local learning tool for children but
also a way to improve the level of education worldwide. In third world countries literacy and its availability for people living in the rural or isolated areas (Isolated from modern western civilization) is poor. Poor literacy can form a barrier to economic empowerment in developing world. Especially learning of language is forming a problem for example in India and technical devices like computers, low-cost laptops and smart cellphones could be of great help for improving these skills within rural children in India (Kam et al., 2008). Modern high level programming languages are often based on spoken languages. If we expand the argument of learning spoken language through games easily so that it applies also in learning programming languages, learning through games on these areas can have even wider global economic impact.

Due to Duncan, Bell & Tanimoto (2014), many concepts in programming are based on concepts from other subjects, such as mathematics and language, so one factor in considering what students can learn is how strong they will be in related subjects. For example, Scratch programming can involve working with a coordinate system and negative numbers, which may be difficult for a younger student to understand. “Many computer-based games incorporate simple computational models involving possibilities of motion and achieving states, or scoring points. The mental models required to understand these games are often abstractions that can be compared with mathematical ideas (e.g., functions: put in this, and that comes out.) This illustrates that the pedagogy can work both ways; computer programming can be used as a vehicle for learning concepts for other subjects and other school- subjects can be used as a “task- framework” for teaching computer programming. (Duncan, Bell & Tanimoto, 2014.)

In section two, I have discussed about why and how are programming skills taught according to reviewed articles. I have also introduced some of the popular games and game- like environments used in teaching according to the reviewed literature. I have also introduced the two of the most popular learning theories used in reviewed literature. Finally, I have discussed the positioning of programming as a subject by trying to find the connections of programming and other subjects. Next, I will introduce the research methods used in this study.
3. Methods

In this chapter, I will introduce the research methods of this study. The nature of this study is qualitative. As a data collection method for this study, I have used semi-structured interviews (3.1). As a data-analysis method I have used qualitative content analysis (3.2).

3.1 Interviews

In qualitative study, most researchers use different forms of interview. Through the interview, interviewer has to encourage people to speak. However the reaction of interviewees is hardly ever constant. Sometimes interviewees are talk more and sometimes lesser. Sharing of thoughts and views between the interviewer and the interviewee is important. A sort of connection or common agreement would be beneficial in interview situation. Basic division of interviews can be made in dividing questions in open questions and closed questions. Closed questions have numerical answers that are easy to measure. Open questions are much more complicated from the scientific point of view because people can be talking for hours, and the interviewer can’t always know, whether answers are relevant or not. Open questions must be formed and structured correctly so that they are suitable for the specific interview- situation in psychological, social and environmental ways. (Myers & Newman, 2007.)

Structured interviews are usually used in quantitative research because they enable the use of statistical tools and analysis. Structured interview has no room for improvisation. In unstructured or semi- structured interview you can ask probed questions and questions that seem to be relevant at the moment of interview. In group interview there are many participants. Problems that are usually seen in the interviews are artificiality (strangerness and time pressure), lack of trust (e.g. SW- secrets), lack of time, level of entry (organizational hierarchy must be taken account, you can’t start always from the bottom level), Hawthorne effect (participants can answer or behave differently when researcher is present. In knowledge constructing, knowledge does not come straight from interviews, it must be constructed and letter or transcript the raw material from people’s answers. Language must be also mutually understandable. In addition, people don’t always want to answer the questions. (Myers & Newman, 2007.)

When designing questions, the research questions must be carried in mind as well as what does the interviewer want to know and how would the interviewer answer to his/her own questions. Questions must be compact without the subjective opinions and double meanings or the hidden agenda. Negatives must be avoided as well as too technical terms. The interview guide / protocol and plan must be used, because every interview is done with the same frame and it will prevent the memory problems of the interviewer. The pilot test provides the interviewers experience beforehand before they actually are making the field study. Existing questions can be used to comparison and guide you making your own questions. Interviews must be implemented in the peaceful quiet place. Common sense can be used in selecting the place for interview. Face sheet is a way to collect starting information like name, age and sex. When interviewing, there are five question types: introducing, follow-up, probing, specifying and direct. If the person is not very talkative, you have to direct the interview in the right direction. Most interviews are recorded so that the material could be revisited and remembered later on. Not everyone however accept the use of the recorder. Transcription of interviews can take a lot of time. Data can be reused more easily because of transcription. Also only the most important issues can be transcribed if there is too much data. According to Myers & Newman
theoretical saturation happens when you can’t get any new insights out of your interviews. Between 10 and 20 interviews is where most interviewers aim. Sometimes few interviews are enough if they are deep enough and maybe selected from the bigger pool of interviews. Superficial 100 interviews can be worse than less made with consideration in mind. (Myers & Newman, 2007.)

Myers and Newman (2007) are also discussing the potential problems and pitfalls of the qualitative interview in IS research. In addition they have provided their own solution to the potential problems in qualitative interviews: Situate the researcher as actor, minimize social dissonances, represent various “voices”, everyone is an interpreter, use mirroring in questions and answers, be flexible and remember the ethics of interviewing.

In this study, I used semi-structured interviews as a data collecting method. I interviewed eight (8) elementary school teachers in the area of Oulu, Finland. The question pattern for the interviews were constructed and used, but the structure of the pattern was also sometimes deviated from planned if the interviewee told something that seemed interesting that could lead in finding out something new or the interviewee was pointing out something important about the matter. Additional questions were asked if necessary and the questions already answered e.g. during the introduction were left out of the pattern. The interview question pattern was divided in the themes based on the topics revealed by literature review and the research questions derived from it. All interviews were recorded with the portable digital recorder. After the interviews were conducted, I transcribed and coded them. After that, I translated them from Finnish to English for those parts that were used in the analysis-section of this study. (Appendix 1 and 2.)

In this study, I used snowball sampling for recruiting participants to interviews. (Fernandez, 2016) In the snowball method, the researcher is using existing members of a sample study group to recruit further participants through their acquaintances. In this case I first interviewed a known expert and asked her to recommend the next interviewee asking “Who do you think, I should interview next about this subject?” This method produced a certain expertise level among the interviewees. However, the expertise level of the interviewees varied naturally and some of the interviewees had higher expertise level due to their experience, motivation, professional orientation and interest towards the teaching of programming skills. In the context of this study, the population = elementary school teachers and snowball method is used to locate the people with expertise in that particular population or to locate the hidden population of coding experts among the teachers.

The disadvantage of using this method is that the actual widely used trends in teaching field might be left out because of the relatively high level of expertise among interviewees. However, in order to find the means to overcome challenges in adoption of teaching programming skills in Finnish elementary schools, this is a minor concern, because finding the means to overcome them often requires a high expertise level, or at least high expertise level does not form a barrier to finding them. Figure 4 shows the basic framework of acquiring participants using the snowball-method.
In Figure 6, I have presented the actual realized sampling of this study. From Fig. 6, we can see that use of the snowball method wasn’t as organized in practice as shown in Fig. 5. Practically the number of participants’ recommendations varied and some participants couldn’t come up with any particular recommendations. In the frames of the timetable, to get this study finished on time, the recommendations of the supervisor had to be used in addition to researcher’s contacts. This accelerated the interviewing process.

In conclusion, it was beneficial to use this method for recruiting interviewees. In the case of I-6 and I-7, the “double recommendation” enabled the arrangement of the interviews on the tight schedule as well as in the case of I-4 and I-5, where recommendation to participate came from authority. Overall the expertise level of the interviewees were high, which was also the intention.
3.2 Qualitative Content Analysis

The analysis of the data in this study resembles inductive content analysis. In inductive content analysis, the researcher using this qualitative method is trying to find out themes based on empirical material. This material can be in form of electric or printed documents, verbal material and recordings. When the researcher is studying this material, he/she is trying to find out themes and patterns that are revealed through the examination and comparison of the raw data. (Symon & Cassel, 1998.)

Figure 1 shows the basic progress of qualitative content analysis from theory to analysis. In inductive analysis, the result of coding is categorization of data. Categories must be defined and the detail level of categories must be decided based on the coding of the raw data. Interpretations are made only based on the information revealed from the data.

![Diagram of qualitative content analysis process]

*Figure 7: Basic proceeding of qualitative content analysis (Gläser & Laudel, 1999)*

Marsh & White (2006) have defined some characteristics of qualitative content analysis. In the inductive research approach, data gathering is guided by research questions but it is possible to rise also new themes and questions from the data processed. Multiple interpretations and different and opposite opinions and conclusions are examined from data in order to find for example alternative or diverse perspectives about matter. Significant concepts and patterns are identified through iterative reading and coding of text. Data selection can continue through the whole project. The goal of analysis is in answering the research questions but also considering “transformations” that are revealed in coding. The original data of study can be included as an appendix for confirmability, credibility and for reviewer to make his/her own interpretations. (Marsh & White, 2006.)
According to Elo & Kyngäs (2008), after choosing the inductive content analysis, the researcher must organize the qualitative data gathered. Organizing is done by open coding, coding sheets, grouping, categorization and abstraction. Based on them, models, conceptual system, conceptual map or categories can be created as a result of the study. In the open coding-state, the categories are created freely and after this stage those categories are grouped under “higher order headings”. The aim of grouping data is to decrease the number of categories. When deciding, which data is part of which category, researcher simply must not bring together the similar or related observations of data but create groups and decide based on his own interpretation in which category the groups belong to. Abstraction is a form of general description of the research topic based on categories. An abstraction process can proceed from detailed level towards the more general level accordingly: sub-category, generic category, main category. (Elo & Kyngäs, 2008.)

Figure 8. Inductive research process. (Blackstone, 2014)

In the inductive research process (Figure 8), researcher is processing data by trying to find patterns by analyzing it. When looking for patterns, researcher needs to have a certain distance to the collected data. It would be good if the data was categorized or coded. Then the researcher can try to generalize a theory based on the patterns found in analysis. While processing data, the researcher is moving from the specific level towards the general level. (Blackstone, 2014.)

However, instead of categorizing the transcription data by themes based on the empirical material only, I have analyzed the interviews based on the topics of the conversations. The data was organized by coding and categorization (Elo & Kyngäs, 2008). I have separated the main themes defined in the research questions and collected the data to coding sheets (Elo & Kyngäs, 2008). From coding sheets, I have identified patterns by comparing the data of the charted transcriptions of the interviews (Symon & Cassels, 1998). Through the patterns found in the data, I have tried to concentrate focus from the specific level towards more the general level of the results in form of the practical implications (5.2). The main objective has been to understand the big picture and transfer from organized, specific findings towards theoretical generalization (Blackstone, 2014).

The analysis of this study is based on the data-material gathered from the interviews of primary school teachers in Oulu, Finland. The interviews were conducted during January - April, 2016. The interviews were recorded and transcribed. They were also coded, translated and charted by necessary parts for analysis. The total word count of transcriptions of 8 interviews is $\approx 10000$ words. Next, in chapter 4, I will provide the results of my data analysis.
4. Findings

In this chapter, I will present my results of the data analysis. First I am describing the data. In the paragraph 4.1, I am presenting the findings of the importance of the teaching programming skills for children. In 4.2, I will focus on the methods of teaching and identified challenges of teaching. In 4.3, I will concentrate on the games and game-based environments used in teaching. In 4.4, I am introducing the opinions of the interviewees about the positioning of the programming as a subject in Finnish curriculum. Finally, in paragraph 4.5 I will introduce the opinions of learning theories revealed in the interviews.

In this study I have 7 interviews in total (I-1-I-7). One interview (I-6a, I-6b) was a pair interview with two teachers so in total I have 8 interviewees. Most of the interviewees were teachers (6/8). In addition, I-3 is a class teacher student almost graduated class teacher- student, who has carried out teacher training periods concerning his studies. Interviewee I-5 is a former class teacher, who has transferred from regular teaching duties to administrative duties. In Table 2 there is a description of the interviewees of this study. I have separated teaching experience in years of each interviewee. The orientation to coding (low, intermediate, strong) was defined by revealed knowledge of the matter. Teaching subject and specialties were asked separately. In addition, there is information about the gender of the interviewee and information of in which school he/she is working.

Table 2: Basic information of the interviewees.

<table>
<thead>
<tr>
<th>Interview</th>
<th>Teaching Experience (y)</th>
<th>Orientation to coding (strong, intermediate, low)</th>
<th>Teaching Subject and Specialties</th>
<th>Gender (F/M)</th>
<th>School and location</th>
</tr>
</thead>
<tbody>
<tr>
<td>I-1</td>
<td>53</td>
<td>Strong</td>
<td>Mathematics, Physics, Computer Science, Elementary class- teacher</td>
<td>F</td>
<td>Oulu University teacher training school, Linnanmaa, Oulu</td>
</tr>
<tr>
<td>I-2</td>
<td>10</td>
<td>Strong</td>
<td>Elementary class-teacher, Coding lessons, ICT-instructor</td>
<td>M</td>
<td>School of Kaakkuri, Oulu</td>
</tr>
<tr>
<td>I-3</td>
<td>&lt; 1</td>
<td>Low</td>
<td>Elementary class-teacher student, ICT-instructor</td>
<td>M</td>
<td>University of Lapland, Faculty of education, Rovaniemi</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td></td>
</tr>
<tr>
<td>I-4</td>
<td>22</td>
<td>Intermediate</td>
<td>Elementary class-teacher, classes 1-2</td>
<td>F</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>School of Tuira, Oulu</td>
<td></td>
<td></td>
</tr>
<tr>
<td>I-5</td>
<td>27</td>
<td>Intermediate</td>
<td>Elementary class-teacher, Headteacher</td>
<td>M</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Kiiminki River School, Kiiminki</td>
<td></td>
<td></td>
</tr>
<tr>
<td>I-6a</td>
<td>19</td>
<td>Strong</td>
<td>Elementary class-teacher, ICT-instructor</td>
<td>F</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Kiiminki River School, Kiiminki</td>
<td></td>
<td></td>
</tr>
<tr>
<td>I-6b</td>
<td>14</td>
<td>Strong</td>
<td>Elementary class-teacher, ICT-instructor</td>
<td>F</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Kiiminki River School, Kiiminki</td>
<td></td>
<td></td>
</tr>
<tr>
<td>I-7</td>
<td>32</td>
<td>Strong</td>
<td>Elementary class-teacher, Music-oriented</td>
<td>M</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Oulu University teacher training school, Koskela, Oulu</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

As shown in Table 2, the interviews were conducted among class teachers except I-3 and I-5. Half (50%) of the interviewees are also ICT-instructors of their schools. The gender distribution of the interviewees is 50% male and 50% female. In general, females seem to be more experienced in this sampling with an extremum of one female participant, who has 53 year experience in teaching. I-1 has also experience of teaching mathematics and physics to older students. Besides being a class teacher, I-7 has a strong orientation in music-teaching for children.
Chart 1. General experience of the interviewed teachers.

Chart 1 shows that overall experience level of the interviewees is high. Most of the interviewees have been teachers for well over 10 years (average = 22 years, median = 21 years).

Chart 2. Participants’ orientation to the teaching computer programming.

Based on the data revealed by interviews, I have defined three attributes for illustrating orientation-level in programming of the teachers: low, intermediate and strong (Chart 2). The attributes are defined according to the answers to the basic information question. "Do you have experience of teaching strictly programming?" Also the interview questions 4, 7, 16, 17 (Appendix B) gave a direction for categorization. The definitions are made by using the following example of the categorization: First, if the ways of teaching asked in the interview (question 4) reveals many ways of teaching. Second, if an interviewee has used game-based environments for teaching programming, or environments that have been designed to teach programming (question 7). Third, he / she has regarded the teaching of programming skills easy and non-challenging (questions 16, 17). If the example answers are as mentioned, the orientation category of the interviewee would be strong. I-1 and I-2 can be described as a reference for strong orientation. I have compared others answers to the I-1 and I-2. From there, I have dropped the level to the intermediate or low. In the x-axis, there is a count of interviews, where one distance means one interview. Chart 2 reveals that there is only one low orientation among the interviews.
belonging to the most inexperienced participant I-3. The general orientation level seems to be good. The problem with defining orientation level accurately is that the meaning of programming should be defined so that every interviewee would have the same perception of the meaning of programming. During this study process it is also revealed that in the context of teaching young children, it is not a simple task to define programming. In this study I have taken also logic and problem-solving related activities in account what it comes to the definition of programming (Chapter 2, p8).

**Chart 3.** Line numbers in text transcriptions and experience in teaching in years.

In the Chart 3, the amount of text-lines is defined on y-axis and interview-numbers are on x-axis. The orange line follows the teachers’ experience in teaching in years. In order to fit the scaling of these two variables (lines, experience), I have doubled the line width of one line of text. In reality, line count in regular text processing software is twice above-mentioned. It seems that the teaching experience of the interviewees and the amount of text in individual interview-transcriptions have a positive linear dependency ($r = 0.81$). Transcriptions are made from the audio recordings of the interviews. The more experienced participants seemed to be more talkative than less experienced.

**Chart 4.** Topics of conversation by the number of questions asked.
As we can see on the Chart 4, I have sorted the topics of the questions accordingly: personal opinions of the teachers about teaching programming skills, why programming should be taught?, how programming should be taught?, what kind of methods have teachers used in teaching, the use and impact of games and game-like environments in teaching, challenges for children, challenges for teachers, position of programming in curriculum, learning theories and finally free association, where the interviewee was able to talk about matters outside the question pattern (Appendix 1).

The graph on Chart 4 is emphasized towards games and game-like environments and teachers’ challenges because question-wise there were more questions about those topics. (Games and game-based environments 26% and challenges for teachers 19% of the asked questions.) Certain topics need more questions to get answers than others for example, because of their complexity. It is easy to map out the used methods in teaching with one question, but to map out the role of games and game-like environments in teaching is more difficult and requires more questions.

![Chart 5](image)

**Chart 5.** The length of answers in relation to the number of questions asked by themes.

On Chart 5, I have separated the long answers by word count from the interview transcriptions by topics or themes of conversation (green line). The yellow line indicates the number of the asked questions by the topic as seen in the Chart 4.

From Chart 5, we can see that by word count, the topics of conversation are following the main trend of questions. However, topics like used methods and how is programming taught and how it should be taught are provoking long discussions without many separate questions. In addition, the position of programming in curriculum seemed to interest the interviewees in general. Next I will present the found reasons for teaching programming to children.

### 4.1 The Importance of Teaching Programming Skills

In this chapter, I will introduce the opinions and interpretations of the teachers interviewed about why they think that teaching of programming skills is important and what is their opinion about teaching coding to children in general.

All of the interviewed teachers think that teaching coding to young students is a positive thing. I-2 and I-6 are stating that teaching coding itself does not contain an absolute value
but teaching coding is a good thing considering the teaching of mathematics and logical thinking. I-1 says that teaching coding is a good thing because it has wide or pervasive attributes. It should be taught because it motivates, challenges, and socializes. It would also be important to eliminate the use of digital devices for only entertainment among children.

“The world has gone digital, and we all have smart phones in our pockets. However, for example reservation for dentist is often hard to put up in a cell phone's calendar, especially for children. Using tablets, cell phones and other devices for useful purposes seem to be difficult and the children seem to use their devices only for entertainment purposes. If all of the useful purposes remain undetected entirely at home, the school should and must be given the capacity for teaching everyday survival skills for children. Finland needs hackers too, people who create something new. If you look at successful coders, so many have learned their skills from a very early age (the best coders). The same is also true for musicians and artists. Everyone do not have to become an ace coder, but nevertheless we should allow the development of these capabilities. Today, you can hardly find a profession where you could say that I do not touch computers. There are smart clothing, intelligent cars etc. If you are left outside, it is difficult to give further education of matter. The teaching of ‘computational thinking’ would be a good word to use.” (I-1)

I-2 and I-5 are highlighting the importance of programming in developing the children’s logical thinking.

“The teaching coding to kids is a really good thing! There is a value in itself, but besides of that in teaching logical thinking and mathematics. I am thinking specifically logical thinking. Programming is certainly also important in the future in many areas but especially the logical thinking in children's development is important.” (I-2)

“I like math a lot myself and I think about things logically. In my view, one of the big thing in programming is the logic. The overall understanding of the logical relationships may clarify through programming for children.” (I-5)

I-4 is bringing the economic context of teaching programming in society forth.

“Programming skills should be taught probably just for the sake of the future of jobs. Coders are required. Society changes. 10 years ago I would not have been able to answer that question (Why it is important to teach programming skills to children?), but now I realize that this life has evolved like this. Of course learning a new thing is always good. 99% of the children begin to have smartphones.” (I-4)

One challenge considering this study is coming up from the answers of this section also, the definition of programming and how wide concept it should be.

“As I said earlier, it is perhaps the vague concept of what programming means. Does the concept relate specifically to computer programming? Programming can also mean that someone knocks you with his fingers on the back, say, to a certain point and then we have to figure out what it means as a solution. These are the things that we have practicing here at school for ages. Sure programming should be taught, but children may not actually be aware, what does it actually mean to program. Of course it should be taught to children!
“Teaching programming is useful. In a certain way it has already been a part of e.g. mathematics education or when providing instructions to pupils. After all that is the basis for coding itself. Now it is brought in computer science... Yes. it is useful for the future. Coding is needed in various issues.” (I-7)

“As the children are constantly surrounded by working program-operated things, this whole life revolves around them. So it would be good that they learn to understand the laws of computing. Then later they maybe also learn to understand and write real code-language. It is tied so closely in everyday life. (I-6a, I-6b)

“When I heard that the teaching programming skills is becoming mandatory, my first reaction was suspecting. I wondered, if that is going to make sense and is it really necessary. Then I realized that my opinion was based on my own perception of the old memories of computer programming in the DOS-ages. Games like Scratch are not based on the code writing. Programming can be lighter and the more practical thing. Besides, I do not have options. Because it is coming in the curriculum, I have to accept it and use it in teaching.” (I-7)

Conducted interviews reveal that the teachers’ definitions of programming are quite versatile. It is possible that in the context of teaching programming skills for children, the definition of programming should be different from its ordinary definition. This indicates that the use of the word computational thinking (CT) instead of programming would be beneficial for that actually is what the teachers are eventually talking about according to the definition of CT in paragraph 2.1 (Papert, 1980; Wing, 2014.).

4.2 Methods and Challenges of Teaching Programming skills

In this chapter, I will present the teaching methods used by interviewed teachers. I will also point out the challenges that Interviewees have experienced in their teaching of programming skills. Also teachers’ observations of how are children relating to the teaching of programming skills is introduced.

The interviewees of I-1 and I-2 are highlighting the connection of given assignment and real world. It would be beneficial if the teacher could find an analogy or descriptive problem from ‘real life’ and process it to teaching by using computational thinking or coding.

"With 1.graders, we have started with coordination. The other pupil is a robot and the other is given tasks and guidance for driving the robot at the floor following a certain path. Then there is the captain commands-game, where you have to give a right starting mark just like in the real programming language has always when starting. In teaching programming skills for young pupils analogies can be used e.g. in music a song can be encoded or you can add programming like elements to learning a song in music class. In physical-classes you have to plan ahead, what you will wear etc. Beebot-robots are a good method for 1-4 graders (a programmable bug that moves around for 15 seconds at the time). I have also shown my teaching methods to other teachers who have crawled on the floor finding paths marked with sticky tape like the children. The early exercises are done on the floor without computers through playing.” (I-1)
Children’s skills and thinking are developing fast. It is not reasonable to use the same teaching methods with for example 1-2 and 3-6 graders. I-1, I-2 and I-6 are using first “real-life” games, where precise instructions have to be followed. After 2. grade, they are transferring to use e.g. Scratch Jr and different learning robots (Image 1).

“Personally, I have tried to sketch the kind of programming path that in the elementary education with 1-2 graders we first program "a human" by giving him oral instructions, and he should then implement them exactly as the instructions are given. After that, we have maybe started programming using the graphical interface and icons for example with different iPad applications and Scratch. Then, being gone forward we have programmed Edison- robots that make things physically. Also the Lego robots have been constructed. So we have started small and continued ahead all the time. On the side, I’m always trying to find an analogy to the everyday life in form of practical substitutes for example: “How automated sliding doors are operating automatically.” (I-2)

“With older students (grades 3-4) we used the Blue Bot- robots in a way, where the students made themselves a platform, where Blue Bot moves. So the children were inventing the game idea. We measured the step size of the robot movement, constructed a grid and depending on the school- subject, we made games where the robot was involved and it had to be programmed (with the tablet) to move in different places depending on assignment.” (I-6a, I-6b)

**Figure 9.** Interviewee I-6b is demonstrating the use of Blue-Bot robot. The robot can be programmed with PC or the tablet via Bluetooth- connection.

According to I-7, the teaching of programming skills should be more practice-oriented than theory-based. He is finding an analogy between teaching music and programming.

“One should begin teaching programming for the lower classes when there is no theory- burden stressing students, so the prejudices towards coding doesn’t necessarily form. One can see it also in other materials. If you first take the theory and start the teaching of children with theory base, your teaching can end
up poorly. We can take for example music teaching using technology. If we use the music technology and software in the class, I will not start by explaining the operating principle first, but from the fact that how we benefit as easily as possible from this tool. Afterwards I tell how it is operating. For example with Garageband, it is the kind of programming. You make different adjustments with the controls to it, you record tracks and you mix them. Then you add effects. When I ask students afterwards, what did you do? Did you understand what you did? Even though the composing is primarily creative operation, the pupils see that the song which has now notated, has been captured from inside them and they are manually themselves able to deal with it using e.g. Sibelius-software. They are also manipulating synthesizer parameters. What else is that but coding? They are learning how to use the software flexibly and in many different ways. Scratch and others use the same ideology.” (I-7)

In Table 3, there are teachers’ views and experiences of how have children related to the teaching of programming skills. Based on the results, children’s relation to learning of programming skills is usually very positive. Programming or coding is generating enthusiastic and joyful feelings and interest in children. In general, they seem to be very motivated. Using of the game-based environments allows learners to be self-directed.

Table 3. How do children relate to the teaching of programming? Teachers’ observations.

<table>
<thead>
<tr>
<th>Interview</th>
<th>Teachers observation</th>
</tr>
</thead>
<tbody>
<tr>
<td>I-1</td>
<td>“Always enthusiastic. Never bored. They want to share their results with classmates. They say that coding is the best class.”</td>
</tr>
<tr>
<td>I-2</td>
<td>“No problems. They are very interested. Even the optional classes are full.”</td>
</tr>
<tr>
<td>I-3</td>
<td>“Enthusiastic. With joy and energy.”</td>
</tr>
<tr>
<td>I-4</td>
<td>“With great joy when tablets are taken in use.”</td>
</tr>
<tr>
<td>I-5</td>
<td>“Their relation to it is very positive.”</td>
</tr>
<tr>
<td>I-6a</td>
<td>“Children like to play for hours. The downside is inequality. There are teachers who don’t teach programming or let children play at all.”</td>
</tr>
<tr>
<td>I-6b</td>
<td>“Nice and enthusiastic. I have also noticed self-directed learning among children. However, If a child lacks logical thinking skills or sustainability, it can be very hard. There are always hard cases.”</td>
</tr>
<tr>
<td>I-7</td>
<td>“Mostly positive relation. There are also those who relate to it negatively. If they have practiced the same things at home, their relation is usually positive.”</td>
</tr>
</tbody>
</table>

Even though children’s relation is identified as “positive”, some interviewees pointed out challenges in the context of this theme. One of the identified challenges (Table 3, I-6b) is also related to the self-direction of the learner. I-1 is also giving the answer to this particular challenge. The role of teacher seem to be important despite the self-direction:
“In self-direction there is danger that children may only focus on the graphical appearance of the task (When using e.g. Scratch Jr. to make an animation.). The functionality itself may remain unimplemented. In this case, teacher must give this style of additional task: "Make the cat move."” (I-1)

According to the interviewees, the one challenge in learning of programming skills is the difference between children’s skills through orientation. Some children are using the same games at home that are used in school teaching. Besides, while children have different levels of programming skills, so do teachers.

“A fear that arises, or the greatest challenge, is not the students but the teachers. We have teachers who are not so enthusiastic or interested in the matter, and then we have those who are. The gap between them is growing. Which will then mean from the student's point of view that there is a great difference in whose class you are ending up? There will be inequality in that situation. In the old days, all learned things the same way and learned how to read by using the same means.” (I-6)

When mapping out the challenges that teachers are identifying in the teaching of programming skills, a few matters arise up more than others: “For many teachers, learning the whole new matter is difficult and very few are willing to learn new things in their own time.” (I-1). The interviewee of I-2 is pointing out interestingly that “It is going to be hard to get every teacher to teach programming. On the other hand we are not confronting any resistance towards it either!” The Interviewee of I-2 is training other teachers to teach coding skills and according to him, it can be hard for some of the colleagues. For those teachers, who are oriented to ICT-area, starting to teach programming seems not to be very challenging. As the biggest challenges they are mentioning lack of machines (computers and tablets), a need for continuous training for the teachers and difficulties of evaluation because there is no official curriculum available yet. As positive sides they are mentioning the amount of possibilities offered by games and personal satisfaction when the new method is successfully implemented at the classroom. (I-1, I-2, I-4, I-6, I-8.) On the other hand, there are lesser ICT-oriented teachers too. “I think that learning of programming skills is easier for the pupils and they learn it easier than teachers” (I-5). In some of the interviews, it comes up that some teachers may be afraid of the situation where students know things better and have better skills than them. I-1 says that they should notice that it doesn’t matter with programming. You can get feedback and instructions from the game. Students are self-directed and they are teaching each other while learning to program through game-like environment. Teaching programming is also often related to technology. According to I-8, there are also challenges in technical infrastructure.

“Technical underdevelopment has often been an obstacle. Trainee-teacher lessons have been cancelled, because the machines or the internet connection doesn’t work. In the context of programming teaching, these have been greater obstacles than the students' willingness to learn new things.” (I-7)

In the past, children have not been allowed to use smartphones or tablets at school time. It has been a problem of defining the allowed use-context for smartphones. In the upcoming new situation, where programming skills are taught using tablets and maybe smartphones also, the rules and practices have to be reconsidered (I-8). The situation is also highlighting the challenge to restrict the pure entertainment use of the computational devices (I-1).
Next I will discuss the role of games and game-like environments in the teaching of programming skills for children at primary schools.

4.3 Game-based Environments and Teaching

In this section, I will discuss of what kind of pedagogic relation the interviewed teachers have towards using game-based environments to support the teaching of programming skills and what kind of games, and game-based environments they use / have used. The information presented in the Table 3 is following the information presented in Table 1, except the gender and school location have been replaced by information on classes taught at the moment and used games and game-based environments. If the classes taught wasn’t specified in the interviews, I have marked 1-6 as an indicator. The student and the headmaster didn’t have classes to teach at the moment.

Table 4: Use of games and game-based environments among interviewed teachers.

<table>
<thead>
<tr>
<th>Interview</th>
<th>Teaching Experience (y)</th>
<th>Orientation to teaching of programming</th>
<th>Teaching Subject and Specialities</th>
<th>Classes</th>
<th>Games / Environments used</th>
</tr>
</thead>
<tbody>
<tr>
<td>I-1</td>
<td>52</td>
<td>Strong</td>
<td>Mathematics, Physics, Computer Science, Elementary</td>
<td>1-2</td>
<td>Racket, Scratch, Scratch Jr, self-made learning games</td>
</tr>
<tr>
<td>I-2</td>
<td>10</td>
<td>Strong</td>
<td>Elementary, Coding lessons, ICT-instructor</td>
<td>3-4</td>
<td>Beebot, Tynker, Scratch, Scratch Jr</td>
</tr>
<tr>
<td>I-3</td>
<td>&lt; 1</td>
<td>Low</td>
<td>Elementary, teacher student, ICT-oriented study path</td>
<td>-</td>
<td>Scratch, Minecraft</td>
</tr>
<tr>
<td>I-4</td>
<td>22</td>
<td>Intermediate</td>
<td>Elementary class-teacher, classes 1-2</td>
<td>1-2</td>
<td>BeeBot, Scratch Jr, Ekapeli, Opinajassa</td>
</tr>
<tr>
<td>I-5</td>
<td>27</td>
<td>Intermediate</td>
<td>Elementary class-teacher, Headteacher</td>
<td>-</td>
<td>Kodu, Perunakellari</td>
</tr>
<tr>
<td>I-6a</td>
<td>19</td>
<td>Strong</td>
<td>Elementary class-teacher, ICT-instructor</td>
<td>1-6</td>
<td>BeeBot, BlueBot, Scratch Jr, Scratch, Kodu</td>
</tr>
</tbody>
</table>
Among the interviewees, Scratch and Scratch Jr. seems to be the mostly mentioned game-like learning environments. In addition for example I-3 and I-8 mentioned Minecraft. Minecraft is a popular 3D computer game where player can create constructs and tools using cubes in open ‘sandbox’ environment. Though Minecraft is not originally engineered to be an educational game, it can be used to create logical exercises for children. Game creators have also published a Minecraft- education edition which is meant to use in classrooms as a tool for learning. They are also co-operating with the Hour of Code which is a global movement for teaching coding for children worldwide. At least Interviewee -1 has been using the Hour of Code in her teaching for many years.

“Hour of Code is an international website. I have used the sites educational games for a long time. It has been existed before the koodi-2016 website. When I years ago used it in teaching, I noticed that every single student was enthusiastic about it, despite the fact that they have been struggling with mathematics. The Hour of Code always gave feedback after assignment "Great you solved the task with eight rows of code. This task can also be solved with five lines of code." The Hour of Code challenged and motivated children. Nowadays, they have managed to gamificite the learning of coding.” (I-1)
Chart 6. The most popular games and game-based environments used by teachers in teaching programming skills to children.

In the x-axis of Chart 6, there are the interview-indicator and the orientation to programming-level of interviewee as defined in Chart 2. The one individual colored block is presenting one game or environment that the participant has been using. The explanation for the colored blocks is presented on the right side of the chart. In Chart 6 we can see that interviewees with strong orientation to programming tend to use more different games and environments in their teaching. The I-5 and I-7 named some additional games like Matikainen and Perunakellari, which I have left outside of the chart because they are schoolbook publishers’ games that are used purely in teaching for specific subject, not programming skills.

All of the interviewees stated that games and game-based environments are a good thing in teaching. For example, they were described as the excellent motivators and tools for drawing attention. In addition games appear to increase the level of engagement.

“Children have strength and endurance to play for hours. If I give them a math problem: Calculate 6 math- assignments quickly, it is sometimes incredibly difficult. If the same thing would be expressed through the game, they would often do it in an instant. Gamification is something that I often do with exercises in classroom. In my opinion, the whole evaluation system of elementary schools should be ‘gamificate’: Instead of the grades you would have to complete certain levels in the semester to pass.” (I-1)

According to I-2 it would be good not to talk about programming or coding to the youngest pupils. I-2 is starting teaching programming skills to children with self-made animations, where the functionality is added little by little as the skills of children are developing.

For example, I have given a simple task to do your animation where two people meet and make friends or something. The students have then created it themselves. With young pupils and Scratch, I am not talking about coding to them, but perhaps more of making an animation. The coding is learned on the side, bringing in new elements little by little. I would not bring the syntax and loops directly to the teaching, but as a tools to allow functions to work. Tynker is also one environment that I've used. It is pretty much the same as Scratch. (I-2)

The teachers interviewed are stating that currently there is enough variety in educational games and new games are developed constantly. The teachers are satisfied in progression happening within the educational game-development area. However, Interviewees I-1 and I-2 are pointing out that there still remains a lot of work to be done in educational game development. “At first you couldn’t share your work with Scratch. Nowadays it is possible” (I-1). Also the extensiveness and diversity of certain games when used constantly is not self-evident: “Scratch Jr can turn out quickly to be too easy or narrow but you can easily jump to Scratch from Scratch Jr“(I-2).

However, there appears to be minor challenges in learning through games and game-like environments as mentioned earlier at the paragraph 4.2:

“When you allow the students to go to a self-directed mode, for example doing a task: Code a psalm or a religious hymn so we can sing it later. There is an idea that we will somehow code a working program. However, in this case the
game-environment included the ability to change drawings of the characters and backgrounds so many of the students mostly focused on the graphical side and the appearance. Functionality wasn’t implemented. The coding part was left undone. In this case, teacher must give this style of task: “Make the cat move.” It is a risk if the original task is too creative. In this case adding an extra task worked. If you want children to be creative, the programming may not be the best way to obtain that, but this also depends on your objectives. However, I am dismayed when I compare today’s and 80’s possibilities with game environments. Still, LOGO- language was inspiring for children already in the 80’s.” (I-1)

Next I will introduce the participants’ opinions about the position of programming in schools as a learning subject or discipline.

### 4.4 Programming and Finnish Curriculum

In this paragraph there are interviewees’ opinions on position of the programming in Finnish elementary school curriculum, teaching material and available support for teaching it. Based on data, there are two different views existing considering the positioning of programming.

Two of eight (25%) interviewees stated that programming could be its own learning subject. According to I-7 that would need a lot of more training for the teachers, even for those who are highly oriented to ICT. Still I-7 wasn’t against the integration either because both ways could work. Even I-2 is for the integration, he sees positive sides in programming being also a separate subject. According to I-2 it could be placed also as a separate optional subject in hour division. According to I-8, the younger the students are, the more useless it is to separate programming as its own subject. According to I-5 computer programming doesn’t contain a value itself but when it’s included in the teaching of natural sciences it would have a purpose.

> “Good question. No. It should not be its own subject in my opinion, because it should be included to the disciplines to be taught, for example the natural sciences. Coding is not the actual purpose in itself. I understand its position in the subject like ADP (automated data processing). At upper school, ADP is the teaching of the use of programs and typing. The coding is not either one.”(I-5)

According to I-1, there is a natural connection between mathematics and programming. Also I-2 sees programming as a part of mathematics, if its position have to be determined to be a part of a particular learning subject. Currently, in the upcoming curriculum of Finnish basic education for primary schools (2016), the programming is planned to be taught as a part of mathematics and it would take its place in the hour division of mathematics (Liukas & Mykkänen, 2016).

> “I am not sure should coding become its own subject at schools. Coding itself is creating a need for mathematics. When you are coding, you start to think e.g. certain angle- calculation mathematically. When a child himself has a problem, he searches the answer using algorithms. At first I guess there was an idea that coding would be taught as a part of every subject. Computational thinking would be learned in everywhere and it would be a small portion of every subject. Mathematics- teachers may be a little disappointed that it will take a slice of the mathematics teaching, but yes, I think that coding increases motivation for learning mathematics. The goals of wide based thinking should be learned with every school- subject. At language point of view learning to write correctly
would be analogically good with coding. Overall it would maybe be better if the
coding would be integrated into the other school- subjects.” (I-1)

According to I-2, if the programming would be a separate subject, the “context” of
learning would disappear. The “context” of learning programming skills has been raised
from the interviews multiple times and later in the paragraph 6.1, I will call it the “real
world substitute”. In general, most of the interviewees think that the integration of
programming to the other subjects would be a good thing. I-1, I-2, I-6a, I-6b had been
using the teaching of programming skills integrated into other learning subjects like
sports, handcrafting, native language studies and mathematics.

“I have been thinking I wouldn’t place it as a separate subject. On the other
hand being separate subject would have certain positive angles: In that case
everyone would certainly participate in learning it. Then again, there is a danger
that the basic idea of why to learn coding would disappear. In my opinion, it
would be much better if it’s integrated into other subjects. Also with that
approach there is danger that it is forgotten why the coding part is integrated in
to certain exercises. It should be in everywhere but if it belongs to somewhere
and time would have to be sacrificed from some subject, the subject should be
mathematics.” (I-2)

“In our work, we have tried to bring coding reasonably into the different school
subjects. On the other hand, we have just calmly trained computer
programming.” (I-6a, I-6b)

According to I-3, separate programming classes could exist, if they would be optional
classes. I-7 has positive experiences of using smartphones and language learning
software. He is finding an analogy between learning programming language and learning
conventional language.

“In my opinion it could be a good thing if there were an optional separate
programming classes where the teachers would be professionals of the subject.
That would equalize the teaching. Programming basics could be taught in
mathematics to all.” (I-3)

“In the native language teaching. There could be a huge advantage from it!
There were immigrants learning the Finnish language from zero skill-level with
the help of software applications for smartphones. That resulted in a huge
success. Maybe the similar thing could work also with teaching programming
languages.”(I-7)

Next I will discuss the available help considering the teaching of programming.
According to the most of the interviewed teachers, the teaching of programming skills is
not hard or challenging for them personally. However, the level of orientation and
experience is relatively high among the interviewees in this study. According to I-2: “It
is not challenging for me, but it is challenging for many of my colleagues.” (I-2)

“Everyone should start it (teaching coding) the next semester, and there are new
projects coming up that are made by good guys for example in relation to
Scratch Jr. Resources have been made to support the official curriculum.
Teachers are experiencing an unfortunate fact that they will have to use their
own time to learn new things.” (I-1)

I-2 is one of the active developers of guide- websites and information packages for
teachers considering teaching of the programming skills.
“It is available (the help) if you know how to look for it but it could be arranged better. We have now the Oulu ICT- learning path under construction, which will have instructions and tips included for teachers. It is a pretty good idea if we get it well done. At the moment, teachers have to dig the materials themselves out. We are just developing there a collection of material. There are also a couple of Facebook- groups who are really active for example with the IPad users and referees of applications. People, who are interested of tablets quickly answer questions there. Currently help is available, if people know where to ask. In the future, when more channels are opening, people get help easier.” (I-2)

According to I-4, courses are arranged where teachers are educated of the matter. Due to I-7, the planning of educating the teachers has been discussed with the colleague in their school.

“I am the ADP teacher in our school and I asked the ICT- trainers to keep concrete evening course, where the whole coding brightened up to the teaching staff. Usually older teachers may have been afraid of the word coding saying ‘This is horrible! I don’t want this in our school!’ There were also teachers in their 60’s, soon to be retired. They stayed in the class the whole time and they quite eagerly were along and said at the end: ‘Now this same thing has always been made but now they have invented a new name to it!’”. (I-4)

“It seems that currently the help, instructions and the teaching material are available, if the teachers know where to ask for them. They are created and maintained by active social networks of teachers who are interested in programming and ICT- related teaching in general. Next, I will introduce the participants’ opinions of learning theories.

4.5 Learning Theories in Teaching

In this paragraph, I will present the interviewees’ opinions, what would be the most usable learning theories in teaching programming skills for young students and what kind of connections they see between learning theories and practice. It seems that some teachers are considering learning theories more than others. Some theories mentioned by more ‘theory-oriented teachers’ are nevertheless the same theories that are revealed by the literature review, GBL and constructivism. (Boyer et. al., 2008; Duncan et. al., 2014; Egenfeldt-Nielsen, 2009; Fessakis et. al.,2013.)

The I-1 seems to be the most theory-oriented teacher of the participants. She is mentioning GBL, constructivism, socio-constructivism and discovery learning.

“GBL especially. Coding is also a good example of where the student is the owner of his own learning. The learner will be capable of self- guiding his/her studies and progression. A student can start the learning process with clear goals e.g. certain kind of picture must be visible after completing a task. It is constructivism oriented, but also socio-constructivism oriented learning. Very much of coding with children is made together. Children can share their products in the classroom with other students and discuss them. Constructivism itself is individualism- based but when we are working here at the classroom, we have an invigorating atmosphere of doing something together. Everyone
wants to show their results to each other, pupils are giving advice to each other and they want also their outputs displayed. They are sharing their own expertise! Immediate learning through shared expertise is socio-constructivism. The other suitable pedagogical model in this context would be discovery learning. (I-1)

I-2, I-5 and I-7 seem to have more practical attitude towards teaching. I-2, I-6a and I-6b are talking about learning by doing by John Dewey. I-1 has also used learning by doing through an exercise, where children were coding learning- games themselves.

“Well, I have applied the system of a 'learning by doing’- method. I personally don’t use so much time to the introduction when giving coding tasks to children. Instead I give the pupils more time to invent and do more stuff themselves. They can add small elements into their exercises and it has worked well. In addition it is exiting for children.” (I-2)

“The other learning theories do not become my mind other than Pavlov’s. If we think GBL, the game serves as a good motivator. Sometimes the learning by computer has caused success in the studying. There could be an advantage even in the quite normal teaching of this self-direction. A student learns by himself using computer. Does this method teach better than the teachers?” (I-5)

I-7 is critical towards constructivism and finds reasons to renew the curriculum in shorter time periods. According to I-7 the suitable theories for teaching programming skills to children are just starting to evolve.

“I am critical towards constructivism. Every class-teacher student is first talking about it and when he quits training period, he usually doesn’t flag for it anymore. The theory concerning teaching and learning programming skills will possibly develop when more experience is obtained of the matter. The new curriculum will probably be re-developed quite quickly, maybe after a few years. Now it should be renewed really because after a year 2000, information technology has been developing so extensively.” (I-7)

In this section, I have presented the most important acquired results from the data and analyzed it. In the next section I am trying to find the answers to the research questions defined in this study (Section 1) based on the data acquired from the literature review and data-analysis. Next I will introduce the results of this study and discuss them.
5. Results and Discussion

In this section, I will first introduce the answers to the research questions based on the interviews and secondly, the found connections to the reviewed literature in Theoretical Implications (5.1). In the paragraph 5.2, I am introducing some practical implications of this study.

Related to discussions about why programming should be taught for children, many positive attributes emerged. Teaching programming seems to increase the motivation of children. Teaching programming also challenges and socializes children. They want to share their accomplishments acquired through programming. The teaching of programming skills could also prevent the use of devices (like smartphones) for only entertainment purposes. The teaching of programming can also develop everyday skills, and logical thinking of children. Overall, teaching programming can develop computational thinking. The need for programmers and SW-workers in future society was also highlighted in the conversations as well as the digitalization of services. Children also have to deal with extensive amount of information in the age of search engines, where the skills of information retrieval are crucial.

Regarding the used teaching methods, teachers have started with coordination exercises while playing child’s games. For example one is a “robot” and one is giving commands or finding paths according to orders. Then they have started to use learning robots like BlueBot. With computers, they have often started with Scratch Jr. and moved using for example Scratch, Racket, Kodu and Tynker when the skills of children have developed enough. Most of the teaching methods are related to games or game-based environments.

The discussion about games and game-based environments indicates that children are enthusiastic towards gaming and using game-based environments. Gamification is an excellent motivator for children. Games and game-based environments seem also to encourage learners’ self-direction because the instructions and feedback, as well as motivation to progress, is provided by the game. There are many game-based learning environments available for learning programming and sometimes it is hard to make the right choice for teaching. In some occasions, children have also made their own learning games for example for mathematics, where they have been using the game-based environment for learning programming as a tool for creating a game. All of the teachers, who have been teaching programming skills to children at primary schools, have been using at least several games or game-like environments as a teaching tool. Some teachers would develop the used games and environments by adding more action and competitive elements to them. The rise of 3D-environments can be interpreted based on discussions for example in form of Minecraft Edu.

When discussed about the position of programming in Finnish primary school curriculum, it seems that teachers don’t like the idea of programming being its own learning subject in the hour division of Finnish primary schools. They would generally like to teach it integrated into other learning subjects. There is a need for mathematics recognized in the conversations. Programming seems to create a need for mathematical thinking. According to interviewees, if the programming had to be a part of another specific learning subject, it would be the mathematics. In the end it seems that it can be taught as a part of any subject.

Many challenges of teaching programming arise from the discussions. The perception of the definition of programming, what exactly is programming called seems to be
incoherent among the teachers. In this study, the participants are at relatively high expertise level and still only a half (4/8) of them had experience in programming with generally known programming languages. The concept of computational thinking emerged in some discussions. That could be the more correct word for the programming or coding in the context of children. Also challenges like how to provide enough and equal amount of training and know-how for teachers and how to get them to train in their own time came up in the conversations. The overall teachers’ consciousness level of programming should be increased. One big challenge seems to be how to provide an adequate training for teachers and clear methods for teaching. There are also economic challenges like how to get enough devices for children and working technological infrastructure for the teaching programming.

When thinking about the role and importance of teaching and learning programming skills in Finnish primary schools, it seems that teachers would like to connect the learning and teaching programming skills to the wide-based learning of school children. The teaching of these skills would be more of a tool than an end in itself. The development of computational devices and networking has been fast in the 21st century. It indicates that it is not only important to teach and learn these skills at early age. It is a necessary requirement for surviving in the digitalized world.

5.1 Theoretical Implications

In this chapter, I am introducing the found connections of the results to the reviewed literature.

Interviews reveal that learning programming motivates children to develop their logical thinking. In the literature, the findings of Clements & Nastasi, (1999); Fessakis, Gouli and Mavroudi, (2013); Duncan, Bell and Tanimoto, (2014) and Kalelioglu, (2015), are supporting the idea that the logical and critical thinking is evolved through the development of better problem solving skills of children, moreover through the development of cognitive and metacognitive skills. In general, high order thinking is evolved while learning programming with games or game-based environments. With games and environments, children are motivated and enthusiastic. According to interviewees, children have the endurance to play for hours. This observation has similarities to the Flow- state (Kiili, 2005) in the learning context. Increased learning is one of the Flow- consequences as well as time-distortion. In the discussions, sharing and collaboration among children came up. Fessakis et. al., (2013) have defined some attributes of educational games. They are mentioning collaboration with teachers and friends when facing difficulties depending on who the child selects as his/her collaborator (Fessakis et. al., 2013). The world also needs programmers because of the fast growth of computing in society. In the future, it seems likely that the need for programmers will grow. Also even if the aim is not to become a programmer, knowing how to program will be beneficial because of the amount of devices and software around us. Learning programming skills teach us to use devices and software flexibly. (Boyder et al., 2008; Duncan et. al., 2014; Rushkoff, 2010.)

Many of the game-based learning environments mentioned by teachers also came up from the literature. For example Scratch Jr, Scratch, Tynker, BeeBot and Minecraft are all included in the article of Duncan, Bell and Tanimoto, 2014. Learning by making games-method (Egenfeldt-Nielsen, 2009), was used by several teachers as well as learning through games- method, which seems to be the most popular way of using the educational game-based learning environments in teaching programming to children. According to discussions, learning programming through game-based environments is encouraging
self-direction of learner. Despite of self-direction, additional instructions are often needed with children. In the reviewed literature, the downside of this self-direction has also been recognized. If children are left totally without guidance, there is a danger that the learning session is turning into “spritefest” (Duncan, Bell & Tanimoto, 2014). Children are concentrating for instance on the visual appearance of animation more than its functionality. According to Fessakis and colleague (2013), they don’t necessarily understand the meaning of sequences of actions that they have to make while learning to code (Fessakis, Gouli & Mavroudi, 2013).

Related to discussions about challenges of teaching programming in Finnish primary schools, providing an adequate training for teachers and clear methods for teaching emerged strongly. Also there was a concern of how to get enough devices for children and working technological infrastructure for the learning. According to Becker (2007), one challenge of teaching with game-based environments is how to get teachers to play the same games that are used in teaching (Becker, 2007). According to Prensky (2001), today’s children live in different culture than older teachers (Prensky, 2001). Therefore it is possible that the successful adoption of GBL in general will not take place until the generation of teachers is changing.

The computational thinking was mentioned in a few interviews. Teachers who mentioned it would like to call teaching of programming teaching of CT. They find the concept of CT more versatile than the word programming or coding. Wing (2014) has defined CT accordingly: “Computational thinking is the thought processes involved in formulating a problem and expressing its solution(s) in such a way that a computer—human or machine—can effectively carry out.” (Wing, 2014) Especially the part “human or machine” is standing up when compared with the discussions. With children, the teachers often start the learning of programming by “programming” humans. In the reviewed articles, Also Duncan and colleague have recognized the value of learning CT (Duncan, Bell & Tanimoto, 2014).

Through thinking what actually programming means, we are at the core of information processing science. We can speculate, how programming should be taught. Logic first-approach seems to be more effective way of the teaching when subjects are under 12 years old, and moreover logic via graphical game-based environment. (Berns, Gonzalez-Pardo & Camacho, 2013; Fessakis et. al., 2013; Maloney et. al., 2008.) However, a lot of teaching materials are taking quite the opposite approach and teach the syntax, variables and logical operations first without giving them understandable meaning that could be obtainable through logic and computational thinking. According to Vuorinen (2015), the programming part is the most routine-like part of doing. Instead of teaching programming for children, the name of the whole subject should be different at elementary schools: computational thinking. Besides programming, computational thinking would include means to understand, how computers and internet work and why the code is needed. The computer-like thinking teaches us means to understanding what a computer is and what kind of problems can be solved using them. Emphasis on teaching the programming skills reveals how little we understand the actual nature of programming. Vuorinen is highlighting that the true nature of computers can’t be revealed by using programming language but changing the way of thinking should be changed. Learning to program is answering one particular question. Computational thinking provides an answer to several others. According to Vuorinen (2015), it is concentrating on what kind of problems can be solved with computers and what computers are good at. Algorithms are just part of the implementation process of problem solving. (Vuorinen, 2015.)
From the point of view of this study, teaching coding for children should include the understanding of the metacognitive development of the subjects. Every child has tools for metacognitive skills as Fessakis and colleague (2013) have denoted. The kind of approach to code teaching should be implemented that notices the role of the metacognition in the context of learning. Learners view and motivation towards programming is not to be suppressed by presenting too hard and complex algorithms in mathematical manner, especially as early as in elementary education. It is hard to say, is the pedagogical answer to the meaningful learning of programming skills in early childhood education intuitive problem solving or is it conscious, constructed thinking? In the biological development of the child, language comes before conscious mathematical thinking in form of speech. It could be possible that the problem solving benefits of the already developed language skills of children instead of mathematics in programming-context. This is not against the mathematics itself, quite the opposite. Mathematics could be its own subject and algorithms could be adopted to it in later education. First approach in code learning could be problem solving chain which encumbers the cognitive load of the subject in the developmental way. A young student can make a pretty house with a pretty background for the drewed cat in game-based learning environment. Young student may be happy with that wanting to share the result with others. There is nothing wrong with that, but the constitutive power of teaching programming skills for young children seem to arise from the following example- situation. The pedagogical problems start from the point where you have to make the cat move and possibly catch some mice. The ideal result for learning this would be that student eventually realizes that if I can make the cat chase mice, I could maybe make the vacuum cleaner- robot to catch dust also. That could be called as the subject’s autonomic invention of the “real world substitute”, introduced in this study.

In the field of computer games, simulations are evolved. For example flight pilots and solders are trained through simulation games. They are trained to handle certain tasks in their professions. Metacognitive skills last for life, or at least to the certain point in life. For example dementia- research has shown that when the cognitive skills of people are starting to fade (e.g. Alzheimer disease), then machines are transferred to help people with their everyday lives. To make this possible and to improve the role of machines in the future, we have to learn how to persuade the machines. Eventually, that is all up to the cognitive and metacognitive skills of autonomic subject. Developing these skills at young age and harnessing them to computational thinking would create a solid constitution for developing future information technology.

5.2 Practical Implications

In this chapter, I will introduce three viewpoints of the emerging of programming in the primary schools. First, I try to clarify what concepts are involved, when talking about programming. Secondly, I will introduce two different frameworks I have prepared to illustrate my current thinking about how programming could be positioned as a learning subject in the school curriculum (5.1.1) and how games or game-based environments should be designed for the purposes of learning programming (5.1.2).

The used terminology between the participants and the stakeholders of this study in general is quite inconsistent. European Commission is talking about coding and programming as synonyms. The Board of Education of Finland is talking about information and communication technique and programming. Teachers and children usually talk about coding and researchers are talking about computational thinking among other psychologic concepts. Figure 10 is constructed based on the results and the definitions of concepts used in this study. With cognitive skills, I am referring to the definition of Miller & Wallis (2009). "Cognitive control refers to the ability to coordinate
thought and action and direct it toward obtaining goals.” Obtaining goals is one of the key elements in learning. Cognitive skills are also mentioned in the literature by for example Fessakis et. al., (2013). The results of the discussions with teachers (I-1, I-5) revealed that only surviving in the digital world is requiring a lot of skills related to computing. Starting to teach these skills in primary schools, could increase the future success of the children.

![Diagram](image_url)

**Figure 10.** Concepts of programming and benefits of learning cognitive skills in ICT-context.

In primary schools, children’s cognitive skills and ICT skills are developed. Through computational thinking, children will get improved understanding about computers, smartphones and tablets and their operating principles. They will learn, what is possible to do with them and what isn’t. The knowledge of what kind of programs can be implemented using certain tools is partly acquired in this stage (CT). Through learning logical programming (or logical thinking) with for example game-based environment, a child will get better ability to use and survive in the world of digitalized services and products. A child will learn how to use search engines effectively, where to find reliable information and possibly ability construct his/her own websites. When the skills in logical and computational thinking are at high enough level, children will learn “real” programming languages and acquire the ability to write their own programs (syntactical coding, Figure 10), creating their own applications and games. That will probably lead to increased possibilities in the future employment market (Figure 10).
5.2.1 Framework for Integrating Programming to other Subjects

This framework is my personal speculation based on literature review and analysed results. Categories of qualitative research are defined in the framework (Elo & Kyngäs, 2008). Background categories: Socio-constructivism, GBL, Computational thinking. Main category: Teaching of programming skills (through game-based environment), Generic category: Positive effects on the learning of subject through programming (motivation, flow-state, sharing) Sub category: e.g. using syntactic precision to learn how to write or using imaginary “add ingredient”- method for creating a recipe.

Figure 11. A framework for integrating programming into the other elementary school subjects in order to develop computational thinking through game based learning environments in Finnish elementary schools.

With pedagogical theories like GBL and socio-constructivism in the background, programming skills could be taught through game-based environments. The use of games and game-based environments in teaching seems to create motivation and positive social outcomes for the learner. These attributes that stimulate learners (motivation, flow-state and sharing) would be transferred to the learning of school subjects like mathematics, language learning and music. The teaching of those subjects could be approached by using the methods of teaching programming skills or to be precise, computational thinking through game-based environments. Therefore it might not be necessary to create a separate new subject to Finnish elementary schools that would be called Programming or Coding. In result of the integration, the hour division would not have to change as much as in the case of programming as a separate subject and in the case of being a part of mathematics. Programming wouldn’t steal a portion of any other subject. The key element would be to use programming as a tool for solving a learning subject-specific problems. (Figure 11. Sub-category) Imaginary learning subject specific problem solving situation in music class could be following: If you hear a long sound, write a whole note. If you hear a short sound, write e.g. a quarter note. If the digital learning environment shows you the sound graph in simplified form and includes functionality for learning programming, the student can formulate conditional expression which generates the markings of the right notes according to the sound graph. The digital
learning environment could be “gamificated” by developers for the realization of the motivation, sharing and flow-state.

Despite this framework is constructed based on the purely inductive grouping of gathered data, the comments from the interviewed teachers with high expertise level and long experience indicate that this kind of integration would be beneficial and desired among the topic-oriented teachers.

5.2.2 Framework for Designing an Educational game

From game-developers’ point of view, this study has revealed several pairs of contrast when thinking on how to design an educational game for learning coding skills. A framework below has been constructed based on these contrasts. The meanings of the contrast-pairs are not necessarily opposite to each other. Therefore I have marked an OR-operator which allows also both of the contrast pairs to be true. The true meaning of this is that the game can have attributes from both sides as in reality they often have. For example, many games are partly open “sandbox”-type games but also have partly scripted parts. Graphical and text-elements are often combined in interfaces. (Figure 9)

The section of the framework that is typical for especially educational games is “learning logic”-part. Based on the literature, we can approach the learning process of coding from two angles. The game can be using “planning ahead”-method, which is appealing to the players cognitive and metacognitive skills through its logic. On the other hand, the game can encourage player to use “trial and error”-method in playing, which based on the results of this study, doesn’t seem to lead into as good results in young students learning of programming skills. Also it seems that instead of teaching pure syntax to children, it would be more effective to start with computational logic. Whether to use “drag & drop” or “code writing” as an input seems to depend on the development stage of the children. (Fessakis et. al., 2013.)

![Figure 12. A framework for designing an educational game for learning programming.](image-url)
In Figure 9, the box outside the diagram, “real world substitute”, is referring to the result revealed in the expert-interviews (I-1, I-2, I-5, I-6ab). It seems that it would be beneficial to associate the coding tasks given to children through game-like environment into the real world so the coding itself wouldn’t be a separate task from reality. If the programming task remains as a too abstract concept for children, the purpose and motivation of doing it decreases. For example, I-2 used the functioning of automated sliding doors as an example of this connection to the real world (Figure 9, real world substitute). Children should think that we are now making automated sliding doors working. They don’t need to know the fact that they are actually taught logical operations through this imaginary exercise. While they are doing it through game-based environment, they are motivated and moreover the potential talent is harnessed for the educational purposes. Ultimately, the goal is to get the learning in school interesting.

In the next section, I am concluding this study with, limitations and suggestions for future research.
6. Conclusion

In the previous section, I have answered the research questions of this study. In this final chapter, I will conclude this study by first summarizing the results. Then, I will discuss the limitations of this study (6.1), and finally some topics for future research (6.2).

It seems to be beneficial for children’s cognitive development to start teaching programming in primary school. It also seems to be beneficial when thinking the future society, digitalization and need for programmers. Programming is first taught with logical “in the real life” games and with digital game-based learning environments. First with drag & drop interfaces and later using writing syntax as an input. Games and game-based environments can motivate, challenge and socialize children. They seem also to encourage learners’ self-direction. Games and game-like environments are mostly positively related by the children and by the topic-oriented teachers. Programming is taught as an integrated part of the other school subjects by those teachers who are oriented to ICT and have enthusiasm to teach computational thinking for children. There are also challenges involved in teaching programming like how to provide enough and equal amount of training and know-how for teachers and how to get them to train programming in their own time. An economic challenge is how to get enough devices for children and working technological infrastructure for the learning.

This study implicates that games and game-based environments are becoming a crucial part of teaching programming skills for young children in Finnish primary schools. The proper way of starting the teaching of programming would be with child’s games that are requiring the using of problem solving and logical reasoning and then taking the students’ skills into consideration, gradually transfer to the more complex issues and game-based environments as well as programming languages. This work implicates also that the educational field in Finland has not prepared very well for the upcoming change in the future curriculum. This situation is causing some uncertainty in the educational field. This situation has also opened and will open up new financial possibilities e.g. for the arrangers of coding-courses for children. This thesis might also be a useful guide for teachers, who are struggling with the starting of the teaching of programming skills. The framework (5.2.1) can be used for understanding the “big picture”. When designing an educational game, the framework (5.2.2) includes some attributes revealed by the data of this study that are worth considering in educational game developing.

6.1 Limitations

The biggest limitation to this work was the tight schedule. The last interview was conducted 19.4.2016 and the paper had to be returned for supervisor’s inspection 20.4.2016. The main causes for this were the tight schedules of the interviewees. The teachers with no orientation or experience, or teachers with negative attitudes towards programming and ICT-related matters were not interviewed within this study. That leaves one angle out of discourse. However, the decision of adopting the teaching of programming to schools has already been made, so interviewing non-oriented teachers would hardly be fruitful except on finding the ways of getting them more involved and motivated towards the topic. The topic of the thesis could have also been bordered narrower so there would have been more time for the analysis and categorization of the results.

During the thesis process, I have tried to keep up the strong connection to information processing by concentrating on the game and game-based environment part of the study.
as well as in the actual methods for learning programming. What it comes to literature review, there could have been more teachers’ opinions represented in it. Currently, it is directed towards why, how and with what kinds of games are programming skills for children taught. In reviewed articles from Computers in human Behavior and Computers & Education- journals, these observations are often made by one researcher from computer or information processing department and one from the department of psychology or educational science. In that sense, this study can be defined as multidisciplinary with a strong emphasis on information processing science.

Within this study, there are no solutions presented for all of the identified challenges. Only for those discussed in the interviewees and in the literature enough are speculated profoundly. Eventually, the main objective of this study has been to identify challenges, not to provide absolute solutions.

### 6.2 Future Research

One purpose of this study has been to find challenges. With research questions, I have found some of the general level challenges in adopting programming to the Finnish curriculum and discussed the possible solutions. Solutions to these challenges aren’t easy to be defined conclusively with information provided by the data of this study. One suggestion for the future research could be to find common methods and instructions for teachers on how to teach programming for children at their different development levels. Also children’s views of the matter could be mapped by observing and interviewing children. It was revealed in the interviews that children may sometimes have better skills considering programming than teachers (I-1, I-4, I-5, I-6ab). This may frighten many teachers (I-1, I-2). Also it was revealed by I-1 that social exclusion of young students is a considerable problem in the educational field. One suggestion for the future research would be to find out could the use of games and game-based environments provide enough positive outcomes like sharing and socialization to prevent the social exclusion of children.

In the future research, the theoretical framework of this study could also be deepened. For example it could include wide-based learning, cognitive load- theory and deeper insight into socio-constructivism. It would also be beneficial to have a more complete framework of designing games for learning programming. The framework could contain more specific information on how to teach functions, variables, iteration, indexed data structures and conditional execution. The framework could be done keeping the young learner in mind. It could be used for designing the game-based environments that would take also the learning theories into account.

One suggestion for the future research would be to map out, how the teachers are talking about programming and what kind of issues they are associating with it. It could be possible that the emergence of programming in primary school curriculum will be eventually referred to differently.
References


Duncan, C., Bell, T., & Tanimoto, S. (2014, November). Should your 8-year-old learn coding?. In *Proceedings of the 9th Workshop in Primary and Secondary Computing Education* (pp. 60-69). ACM.


Appendix A. Structure of the Interviews in Finnish

Question pattern:

1. Mitä mieltä olet ohjelman opettamisesta lapsille?
2. Miksi ohjelmointia pitäisi opettaa lapsille, miksi se on tärkeää?
3. Mitä huonoja puolia näet ohjelman oppitaulun peruskoulun opetussuunnitelmaan?
4. Millä tavalla olet opettanut ohjelman lapsille?
5. Millä tavalla haluaisit opettaa ohjelman lapsille?
6. Millä tavalla ohjelman pitäisi opettaa mielestäsi eri ikäryhmille?
7. Oletko käyttänyt pelejä ohjelman opetuksen tukena?
8. Jos olet, millaisia pelejä?
9. Mitä ajattelet pelien tai pelitaitoisten avulla oppimisesta?
10. Onko ohjelman oppimispelejä / pelitaitoista mielestäsi saatavilla tarpeeksi?
11. Millaiset pelit ovat mielestäsi tarkoituksenmukaisia?
12. Mitä hyväksi / huonoa peleissä on mielestäsi?
13. Mitä kehitteit sinut peleissä?
14. Mitä haasteita näet ohjelman oppitaulussa lapsen kannalta?
15. Miten lapset ovat suhtautuneet ohjelman opetuksen?
16. Oletko käyttänyt ohjelman oppitaulun oppitaulun avulla oppimisesta?
17. Miten haastavaksi olet kokenut ohjelman opetuksen?
18. Miksi olet kokenut ohjelman haastavaksi?
19. Mitä mieltä olet tällä hetkellä ohjelman oppitaulun tukesi tarjolla olevasta opetussuunnitelmasta?
20. Koetko saavasi tarvittaessa apua ohjelman oppitaulun tapausten asioissa?
21. Minkä aineen ohjelman pitäisi mielestäsi opettaa?
22. Pitäisikö ohjelman olla mielestäsi oma aineensä?
23. (Onko mielestäsi hyvä asia, että se on sijoitettu matematiikan tuntijakoon?)
24. Mistä oppimisteorioista lähtöisin ohjelman opetusta?
25. Onko oppimisteorioilla merkitystä ohjelman oppitauksessa?
26. Mitä merkitystä näkisit oppimisteorioilla olevan ohjelman oppitauksen kannalta?
27. Haluatko kertoa jotain aiheeseen liittyvästä kysymystä ulkopuolelta?
28. Olisiko sinulla suositusta, ketä kannattaisi haastatella tänään asian tiimoilta?
Appendix B. Structure of the Interviews in English

Question pattern:

1. What do you think about the teaching of the programming to the children?
2. Why the programming should be taught to children. Why it is important?
3. What disadvantages do you see in the joining of the programming teaching to the curriculum of the comprehensive school?
4. In which way have you taught programming in to the children?
5. In which way would you like to teach programming to the children?
6. In which way should the programming be taught in to different age groups, in your opinion?
7. Have you used games as support of the teaching of the programming?
8. If you have, what kind of games?
9. What do you think of with the help of the games or game environments from the learning?
10. Are there enough game environments for learning programming available, in your opinion?
11. What kind of games are suitable, in your opinion?
12. What is good or bad in the games in your opinion?
13. What would you develop in games?
14. What challenges do you see in the learning of the programming from the point of view of the child?
15. How have the children reacted to the teaching of the programming?
16. Have you regarded the teaching of the programming as easy or challenging?
17. How challenging have you regarded the teaching of the programming?
18. Why have you regarded it as challenging?
19. What do you think of the currently available teaching material for supporting a curriculum?
20. Do you experience that you are getting, if necessary, help in the educational matters of the programming?
21. In which school-subject should the programming be taught in your opinion?
22. Should the programming be its own school-subject, in your opinion?
23. (Is it a good matter that it has been placed in the hour division of the mathematics, in your opinion?)
24. From which learning theories should the programming be taught?
25. Do the learning theories have significance in the teaching of the programming?
26. What significance would you see in learning theories from the point of view of the teaching of the programming?
27. Do you want to tell something from outside the questions, connected to the subject?
28. Would you have a recommendation who would be worth interviewing about this matter?