CODING WITH MINECRAFT: THE DEVELOPMENT OF MIDDLE SCHOOL STUDENTS' COMPUTATIONAL THINKING

EMİNE KUTAY

BOĞAZİÇİ UNIVERSITY

2020

CODING WITH MINECRAFT: THE DEVELOPMENT OF MIDDLE SCHOOL STUDENTS' COMPUTATIONAL THINKING

Thesis submitted to the

Institute for Graduate Studies in Social Sciences

in partial fulfillment of the requirements for the degree of

Master of Arts

in

Educational Technology

by

Emine Kutay

Boğaziçi University

2020

DECLARATION OF ORIGINALITY

I, Emine Kutay, certify that

- I am the sole author of this thesis and that I have fully acknowledged and documented in my thesis all sources of ideas and words, including digital resources, which have been produced or published by another person or institution;
- this thesis contains no material that has been submitted or accepted for a degree or diploma in any other educational institution;
- this is a true copy of the thesis approved by my advisor and thesis committee at Boğaziçi University, including final revisions required by them.

Signature……..

Date……..

ABSTRACT

Coding with Minecraft:

The Development of Middle School Students' Computational Thinking

The purpose of this study is to examine the role of Minecraft-based coding activities on computational thinking (CT) of middle school students. In the study, CT is conceptualized so that it encapsulates not only the knowledge of CT concepts but also the use of CT practices. In this regard, the study employed one group pre-test post-test design, supported with qualitative data. Data were collected using a combination of CT concept knowledge tests, the Minecraft-based computational artifacts, and artifact-based interviews. The participants were 5th-grade middle school students (n = 11 female and n = 9 male) at a public school in Istanbul who had no formal coding experiences prior to the study. The Minecraft-based coding activities were designed and implemented as an instructional program to last six weeks. Students were pre-tested before attending in the program. After participating in weekly lessons consisting of Minecraft-based coding activities, students were post-tested and interviewed. The results showed a statistically significant increase in students' knowledge of CT concepts after participating in the Minecraft-based coding activities. The analysis also showed that there were no statistically significant differences between students' scores in terms of gender. Based on the analysis of Minecraft projects, the concepts of variables, operators, and conditionals appeared to be the least used concepts, while students mostly made use of loops and events in their computational artifacts. In terms of the qualitative analysis of artifact-based interviews, students displayed CT practices of experimenting and iterating, testing and debugging, reusing and remixing, and abstracting and modularizing.

ÖZET

Minecraft ile Kodlama:
Ortaokul Öğrencilerinin Bilgisayarca Düşünmelerinin Gelişimi

Bu çalışmanın amacı, Minecraft tabanlı kodlama etkinliklerinin ortaokul öğrencilerinin bilgisayarca düşünmeleri üzerindeki rolünü incelemektir. Çalışmada, bilgisayarca düşünme hem bilgisayar kavramları bilgisi hem de bilgisayar bilimi pratiklerini içerecek şekilde kavramsallaştırmaktadır. Bu bağlamda çalışma, nitel verilerle desteklenen, tek grup ön-test son-test deneysel öncesi araştırma deseninde tasarlanmıştır. Veri toplama araçlarını, bilgisayar kavramları bilgisi testleri, öğrencilerin çalışma sonunda ürettikleri Minecraft kodlama projeleri ve öğrencilerle gerçekleştirilen bire bir görüşmeler oluşturmaktadır. Katılımcılar, İstanbul'da bir devlet ortaokulunda eğitim gören ve daha önce herhangi bir bilgisayar programlama deneyimi olmayan beşinci sınıf (n = 20) öğrencilerinden oluşmaktadır (11 kız ve 9 erkek). Minecraft oyun ortamında hazırlanan kodlama etkinlikleri altı hafta süren bir öğretim programı olarak hazırlanmış ve uygulanmıştır. Öğretim programı başlamadan önce öğrencilerin bilgisayarca kavramlar bilgisini ölçen bir ön test uygulanmıştır. Öğrenciler haftalık Minecraft tabanlı kodlama etkinliklerinden oluşan derslere katıldıktan sonra öğrencilere bilgisayarca kavramlar son testi uygulanmış ve öğrencilerle görüşmeler yapılmıştır. Bilgisayarca kavram testi puanlarının analizi, Minecraft tabanlı kodlama etkinliklerinin bilgisayarca kavram bilgilerinin geliştirilmesi açısından etkili olduğunu göstermiştir. Kız ve erkek öğrencilerin test skorları arasında anlamlı bir fark olmadığı görülmüş, Minecraft tabanlı etkinliklerin hem kız hem de erkek öğrenciler için eşit derecede etkili olduğu sonucuna varılmıştır. Öğrencilerin Minecraft projeleri incelendiğinde, değişken, matematiksel operatörler ve koşul yapılarının en az kullanılan kavramlar olduğu ortaya çıkarken;

projelerde en fazla döngü ve olay yapılarının kullanıldığı görülmüştür. Görüşme verilerinin analizi, öğrencilerin bir kodlama projesi geliştirirken çoğunlukla test ve hata ayıklama gibi bilgisayar bilimi pratiklerinden yararlandıklarını gösterirken; en az, yeniden kullanma pratiğini işe koştuklarını ortaya koymuştur.

ACKNOWLEDGEMENTS

I am pleased to acknowledge the substantial contributions of those who helped me with my thesis. Foremost, I would like to express my sincere gratitude to my advisor, Assoc. Prof. Diler Öner for her continuous patience, motivation, encouragement, and immense knowledge during my research study. Her dedicated support and guidance helped me in all the time of research and writing of this thesis. It was a great pleasure for me to conduct this thesis under her supervision.

Besides my advisor, I would like to thank to each of the members of my thesis committee, Assist. Prof. Mutlu Şen-Akbulut and Assist. Prof. Rıdvan Ata, for their insightful comments, feedbacks, and suggestions. I would also like to thank Tufan Adıgüzel and Neslihan Er Amuce, who I consulted their content knowledge and expert opinions while adapting the instruments used in the study. In addition, Assist. Prof. Senem Yıldız and Assist. Prof. Nazik Dinçtopal-Deniz Kaya, thank you for their contributions to translating the instruments used in this study from English to Turkish.

I also thank Erhan Bingöl for his permission to conduct this research in the school he is principal and Umut Görkem Sevinç, who volunteered to participate in this study with their students. I also would like to thank Ömer Sarpdağ from Microsoft, who provided me with the Minecraft accounts to carry out this study.

I have to express my immense gratitude to my father, İsmail Kutay and my brother Emrecan Kutay who have always been sources of continuous support for me throughout my entire life. Special thanks to my mother, Saliha Kutay who provide me with encouragement, love and care. She stood by me all throughout this tough

process and had faith in my ability to succeed. I could not complete this thesis without her support.

Lastly, I am very grateful to my precious love, my fiancé Tolga Korur, for his great care, patience, tolerance and continuous support. He has always been there for me during this tough and tiring process.

Thank you all.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

In recent years, in all over the world, many attempts have been made to integrate computational thinking (CT) into K12 education (Grover & Pea, 2013). This form of thinking is considered the sine qua non of the 21st century because, with great technological advances, future professions need people who can think computationally (Wing, 2006). It is fundamental for all people because computational models that are formed by thinking computationally are used in many professional areas from science and engineering to art and history (Wing, 2006); thus, it should be a part of education, starting from early childhood (Wing, 2008). The basis of this way of thinking is based on computer science and programming (Bocconi et al, 2016). Hence, CT is developed as students engage in computer programming activities (Grover & Pea, 2013) and create computational artifacts (Brennan & Resnick, 2012).

With the integration of computers in education in the 1900s, the seeds of CT began to be laid. In 1980, Seymour Papert stated that experiences with computer programming (or coding) in K-12 education teach children to think about their thinking. Although Papert emphasized the importance of computer programming for children, until the 2000s, developing thinking skills through coding was not accentuated in K-12 education (Lye & Koh, 2014). The concept of CT began to gain importance with Jeannette Wing in 2000s. According to Wing (2006), CT is an ability to think such as a computer scientist and is needed to be gained by everyone, not solely by computer scientists. After her initial description of CT, many attempts

have been made to understand what CT is and how to develop this form of thinking. With Wing's explanations, coding in K-12 education became educators' agenda again (Grover & Pea, 2013) as CT contains several computer science and coding concepts such as testing, debugging, and abstraction (Brennan & Resnick, 2012; Wing, 2008; Wing, 2006).

Over the past several years, many researchers offered various CT models to explain CT and its components (e.g. ISTE & CSTA, 2011; Kazimoglu, Kiernan, Bacon, & MacKinnon, 2012; Lee et al., 2011). Despite several CT definitions and models, there is little consensus about what computational thinking includes (Brennan & Resnick, 2012). In this study, CT is defined based on the framework proposed by Brennan and Resnick (2012). According to this framework, CT includes three dimensions: computational thinking concepts (the programming concepts used when coding such as conditionals and loops etc.), computational thinking practices (the practices that designers develop while creating a computational artifact), and computational thinking perspectives (the designers' perspectives on how they understand the technological world and interact with others in that world). The component of CT perspectives was not included in this study because understanding CT perspectives is not possible by directly asking children according to Brennan and Resnick (2012). Therefore, this study focused on the components of CT concepts and practices.

Recently, many studies have been carried out in developing CT. As students are interacting with rich computational tools and construct computational artifacts through programming, their CT can be fostered (Kafai & Burke, 2013). Several ways to develop CT and teach basic programming involve visual block-based programming environments (Lye & Koh, 2014), the use of robotics (Kwon, Kim,

Shim, & Lee, 2012; Sullivan, Kazakoff, & Bers, 2013), and unplugged computing activities (Curzon, McOwan, Plant, & Meagher, 2014; Brackmann, Román-González, Robles, Moreno-León, Casali, & Baroni, 2017). In addition, relatively less research focused on the use of simulations and video games to develop coding skills (Lee et al, 2011).

In the age of technology, educators and researchers argue that video games can be a powerful medium for teaching and learning activities (Bourgonjon, Valcke, Soetaert, & Schellens, 2010; Gee, 2005; Squire, 2013). The attempts to use video games for educational purposes have increased in recent years (Cipollone, Shifter, & Moffat, 2014; Saez-Lopez, Miller, Vasquez-Cano, & Dominguez-Garrido, 2015) because many researchers emphasized the positive effects of video games on children's motivation and engagement (Gee, 2005; Steinkuehler, Squire, & Barab, 2012).

Minecraft is one of the most successful video games played by children (Sarkar, 2017). According to Gee (2003), being productive and interactive, and having a sense of control and ownership are important characteristics of video games that foster learning. In this regard, Minecraft offers players an open-world that allows them to explore and create their unique worlds by setting their own goals in the game (Kuhn, 2018).

In 2012, Minecraft: Education Edition (M: EE) was released for educational purposes. It has been used in many disciplines such as mathematics, chemistry, language and literacy, and art (Nebel, Schneider, & Rey, 2016; Voogt, Fisser, Good, Mishra, & Yadav, 2015). Since 2016, the MakeCode coding plugin was integrated into Minecraft and some computer science lesson resources and curriculums were shared with teachers. Although the importance of coding in K-12 education has been

emphasized in the last century (Voogt et al., 2015), students have not always had a positive attitude towards coding (Başer, 2013). However, a tool such as Minecraft can be very effective to teach CT skills. Nonetheless, there is little research -if none- in the literature on the use of Minecraft on the development of CT skills.

In a few studies, gender issues in coding have been addressed, especially for children (Papavlasopolou, Sharma, Giannakos, & Jaccheri, 2017). Girls are considered an underrepresented population in computer science areas (Cheng, 2019; Luo, Antonenko, & Davis, 2020). Despite efforts, there is an indispensable gender gap starting from elementary school (Papavlasopolou, Sharma, & Giannakos, 2020). To understand that how boys and girls show performance in coding activities, for example how they use CT concepts and employ CT practices while coding, is important to design appropriate activities for both groups (Papavlasopolou et al., 2017). The findings on gender differences in coding are sparse and not consistent. Therefore, more studies are needed to focus on the role of gender on the development of CT, so that coding activities that engage both groups can be identified.

While the discussions about what the CT is and how it is developed still continue, how exactly this way of thinking should be measured is ambiguous, too. In the related literature, various assessment methods are suggested ranging from multiple choice tests (e.g. Grover, 2014; Roman-Gonzalez, Perez-Gonzalez, & Jimenez-Fernandez, 2016) to the evaluations of computational artifacts (e.g. Moreno-Leon, Robles, & Roman-Gonzalez, 2016; Brennan & Resnick, 2012; Werner, Denner, & Campe, 2012). Since CT is a form of thinking which have many dimensions, researchers suggested that more than one assessment form such as observations, conducting tests assessing conceptual knowledge, examining students'

artifacts, and interviewing students about their products, should be used to assess the development of CT (e.g. Grover, 2014; Brennan & Resnick, 2012).

Based on the CT framework used in the study, three approaches were used to assess the development of CT: tests to evaluate students' knowledge of CT concepts, artifact-based interviews, and a design-based scenario approach. A CT concept test was used to measure the knowledge of CT concepts before and after the intervention. Based on this test, students' learning gains were calculated and analyzed. The second approach for assessing the development of CT is artifact-based interviews. This approach is aimed to develop a better understanding of how students develop CT practices while developing a computational artifact. Thus, the main focus of artifact-based interviews approach is understanding the process of developing a computational artifact in terms of CT concepts and practices. The last approach is design-based scenarios approach. The purpose of design-based scenarios approach is the same as artifact-based interviews approach, but in design-based scenarios approach, students are expected to understand a project designed by someone else, to fix bugs in that project and to add something new to this project. Thus, we can better understand how students develop CT practices of testing, debugging, and reusing and remixing. In the light of these approaches, using various data collection means increases the validity of findings by triangulating the data (Meerbaum-Salant, Armoni, & Ben-Ari, 2013).

In summary, since CT is an important thinking skill for everyone, the importance given to coding has also increased. Even if there is no common definition of CT in the literature, its importance is evident. In addition, children develop their CT skills engaging in coding through several tools. Video games encourage the CT development of students to facilitate coding through motivational and engaging

factors of video games (Kazimoglu et al., 2012). However, there are few studies in the literature investigating the use of video games in learning coding (Guzdial, 2008). Additionally, measurement and assessment methods play an important role to understand the programming process of students. But in the current literature, there is a need on how to measure the development of CT (Grover et al., 2017). Even if there are some assessment methods suggested by researchers, these methods are not generalizable to each programming environment. Therefore, there should be assessment methods which are independent of coding learning environment (Chen et al., 2016).

1.1 Purpose of the study

The purpose of this study is to examine how middle school students develop CT concepts and practices when they are engaged in coding activities with Minecraft. In addition, the study focuses on the gender differences in CT concept development.

1.2 Research questions

To reach the purpose of this study, the following research questions and sub-questions are asked:

1. Is coding with Minecraft activities effective in developing CT concepts and practices?
    a) Is there a statistically significant difference between pre-test and post-test scores of students as measured by the CT concept knowledge test?
    b) What are the number and type of CT concepts displayed in the final Minecraft-based computational artifacts?

c) What are the number and type of CT practices displayed by students in the interviews?

2 Do girls and boys differ in the development of CT concepts?

a) Is there a statistically significant difference between learning gains of female and male students as measured by the CT concept knowledge test?

b) Is there any difference between boys and girls in terms of number and type of CT concepts displayed in the final Minecraft-based computational artifacts?

1.3 Research hypotheses

H1: Post-test scores of students measured by the CT concept knowledge test will be higher than the pre-test scores of students.

H2: There will not be any statistically significant difference between the learning gains of boys and girls, as measured by CT concept knowledge test.

H3: Students will use all the CT concepts (sequences, parallelism, loops, events, conditionals, operators, & data) in their final Minecraft-based computational artifact.

H4: There will not be any statistically significant difference between female and male students in terms of using CT concepts in their final Minecraft-based computational artifact.

1.4 Organization of the sections

Chapter 2 covers a literature review of various computational thinking explanations, approaches to develop CT and to assess it, the relationship between constructionism and programming, and also the relationship between constructionism and open-ended video games. The research methodology chapter, Chapter 3 consists of the research

design, the context and participants, data collection process and instruments, and lastly the data analysis procedure and Chapter 4 focuses on the results of data analysis. Lastly, the discussion of the findings, the limitations of the study and recommendations for future research are introduced in Chapter 5.

CHAPTER 2

LITERATURE REVIEW

In line with the research purpose of the study -examining how middle school students develop CT concepts and practices when they are engaged in coding activities with Minecraft- the relevant literature was reviewed. Accordingly, this section is organized under three main titles: The definitions of CT, approaches to develop CT, and approaches to assess CT. Then, the purpose of the study and research questions were put forward.

2.1 Computational thinking

Computational thinking became a universally accepted thinking skill with Jeannette Wing publishing a short article in 2006. Since then, CT has started being considered an important thinking skill in K-12 education (Lye & Koh, 2014). According to Wing (2006), CT is a thinking process involving problem-solving needed to be acquired by all students (Wing, 2006). After Wing's definition, various definitions of CT have been put forward (Hu, 2011; Weese, 2017).

Some researchers and communities argue that the foundations of CT are related to the practices of computer science (Bocconi et al., 2016) so, it has a reciprocal relationship with computing and computer programming (e.g. Guzdial, 2008; ISTE & CSTA, 2011), whereas some advocate that it should be thought separately from computer science and computer programming and integrated into other domains, such as mathematics, science, and engineering (Denning, 2009). However, looking at the essence of the term CT, it is mostly associated with computing and computer programming (e.g. Papert, 1980; Wing, 2006).

Although the importance of CT has been recognized in the 2000s, the idea of CT in the education context has emerged in various forms and under different names (Weintrop, Holbert, Horn, & Wilensky, 2016). Seymour Papert (1980) who was the first to come up with the idea of "computing" used the term "procedural thinking" through computer programming. He suggested that computer programming provide children to explore their thinking. Similarly, diSessa (2000) called the notion of CT as "computational literacy". He proposed that computing is a powerful medium to explore and students use computing to create something in the age of computers.

Wing (2008) revised and expanded her first definition as CT sharing with other kinds of thinking such as analytical and mathematical thinking, and scientific and engineering thinking. Wing (2011) later defined CT as "a thought process consisting of generating problems and solving them, so the solutions of the problems are represented in a form that can be effectively carried out by an information-processing agent" (p.1). Royal Society (2011) states that CT is a process of understanding and interpreting events involving computing by using tools and applications of computer science. Weintrop et al. (2016) view CT as an ability to concretize ideas by using computational tools and benefiting from the knowledge and abilities obtained from the discipline of computer science.

According to Gülbahar, Kert, and Kalelioğlu (2018), the most important common aspects of CT explanations are that CT is based on the problem-solving process even though the term of CT is defined in different ways in different studies. Some researchers combined other kinds of skills and thinking when defining CT, not only problem-solving. For example, Ater-Kranov, Bryant, Orr, Wallace, and Zhang (2010) argued that critical thinking and problem-solving are the most accepted two abilities in the literature about CT. Also, CT contains the main five skills: problem-

solving, building algorithms, debugging, simulation, and socializing (Kazimoglu et al., 2012). Similarly, ISTE and CSTA (2011) explains CT as a process that includes defining problems and designing solutions to the problems with the help of computers and other tools. Additionally, Grover and Pea (2013) identified various accepted CT elements from the literature: abstractions and pattern generalization, systematic processing of information, symbol systems and representations, algorithmic notions of the flow of control, structured problem decomposition (modularizing), iterative, recursive, and parallel thinking, conditional logic, efficiency and performance constraints, and debugging and systematic error detection.

Since in the literature, there are many CT definitions and explanations and differs from each other, studies classifying the dimensions of CT in a single framework have started to be needed (Demir & Seferoğlu, 2016). Regarding this issue, Brennan and Resnick (2012) designed a CT framework with three dimensions by evaluating students' artifacts created Scratch programming environment and interviewing with them. The three dimensions of their framework are computational concepts, computational practices, and computational perspectives. The components of each dimension are shown in Table 1. This framework is appropriate for regarding CT for in the context of coding in K-12 education (Lye & Koh, 2014).

Table 1. CT Framework

Dimensions	Definitions	Components
Computational concepts	concepts that are used when coding	Sequences, loops, events, parallelism, conditionals, operators, data
Computational practices	practices that users develop while creating an artifact using CT concepts	Experimenting & iterating, testing & debugging, reusing & remixing, abstracting & modularizing
Computational perspectives	perspectives that users' understanding of the digital world around them and their relations with others in this world	Expressing, connecting, questioning

Source: [Brennan & Resnick, 2012]

Brennan and Resnick (2012) proposed this framework by examining students' projects, observing students, and interviewing with them about their computational artifacts over several years. Though the framework was developed in the context of the Scratch environment, it can be adaptable in other coding environments to observe programming experiences of K-12 students (Lye & Koh, 2014). The framework is a multi-dimensional framework which contains common programming concepts and required practices when building an artifact through coding. It is suitable in order to examine CT development in programming environments (Lye & Koh, 2014). That's why, this CT framework was selected for this study.

2.1.1 The CT framework

2.1.1.1 Knowledge of CT concepts

When learners engage in computer programming, they use a bunch of computational concepts. These concepts are used in other programming environments and languages. CT concepts consist of seven major programming elements: sequences, loops, parallelism, events, conditionals, operators, and data.

Sequences: This concept refers to the fact that the commands executed by a computer are in the order they are written. It is a key concept of programming because computers need a set of the logical order of steps to carry out programs. Computers need commands in the correct order to perform the task.

Loops: In programming, there are some tasks in which you want to do the same thing many times. For example, to create a program that draws a square with an edge of 10 cm, the computer is given the following commands: Draw a 10 cm line, then turn right. Since the square has four edges, these two commands must be repeated 4 times. Loops are used for such repetitive tasks.

Parallelism: Parallelism allows the execution of multiple sequences of instruction at the same time. For example, when playing music in the background, the character in the program may say something.

Events: These are actions that trigger other actions. For example, when the game plays, the player says hello.

Conditionals: Conditionals are an important part of the decision-making process for computers. They are basic "if-then” logic statements. Passwords are given as an example to conditionals. If users enter their passwords correctly, then they can log in to the system.

Operators: Programmers write the computer code using mathematical (subtraction, division etc.), logical (true or false), and string (length of a string) expressions through operators.

Data: Data can be thought of a container that stores information in a computer program. There are different types of data related to the information it has: strings, integers, and arrays. Integers keep numeric values while strings keep a group of characters which can be a word or a sentence. Also, array variables hold a list of related variables, both strings and integers.

2.1.1.2 Computational thinking practices

Computational practices refer to thinking about how learners are thinking and learning while creating a computational artifact. CT practices represent the process of construction of an artifact and consist of four components: experimenting and iterating, testing and debugging, reusing and remixing, and abstracting and modularizing.

Experimenting and iterating

Designing a project is an ongoing process. A designer develops her project constantly adding new features and elements. That is, when experimenting and iterating, designers try out new ideas on their own or with the help of others to develop their projects. There are some strategies (indicators) suggested by Brennan and Resnick to understand how students exhibit experimenting and iterating. The strategies for experimenting and iterating practice are:

- Building a project step by step
- Trying things out as they go
- Manage revisions based on what happens

- Trying different ways to do things or try new things

Testing and debugging

When designing a computer program everything in the first attempt does not work as desired. Designers develop some strategies to find out why the program is not working. They identify the problem, develop a solution and test the solution. This process continues until they debug. In this process, they can re-write scripts, find example scripts that work, or get help from someone about the error. Designers use testing and debugging CT practice if they demonstrate the following strategies while building a computational artifact through coding:

- Observing what happened when they run their project
- Describing what is different from what they want
- Reading through the scripts to investigate the cause of the problem
- Making changes and test to see what happens
- Considering other ways to solve the problem

Reusing and remixing

In programming, reusing and remixing is an important practice for programmers to read someone else's code and use it in their own programs. It supports code-reading practices and how to give credit to others. For reusing and remixing practice, designers use in the following strategies:

- Finding ideas and inspiration by trying other project and reading the scripts
- Selecting a piece of another project and adapting it for their projects
- Modifying an existing project to improve it
- Giving credit to people whole work they build on or are inspirited by

Abstracting and modularizing

This form of practice is about “building something large by putting together collections of smaller parts” (Brennan & Resnick, 2012, p.9). Designers make plans by deciding on project scripts (codes) and sprites based on things what they want to happen. They display the CT practice of abstracting and modularizing if they can mention about how they decided what sprites and codes are needed for their projects based on their decisions to add items or rules to their projects and how they organized these sprites and codes in ways.

2.1.1.3 Computational thinking perspectives

Computational thinking perspectives, that is, expressing, connecting, and questioning focus on how learners develop their point of views, providing an opportunity to see what they can do while engaging in interactive media (Brennan & Resnick, 2012). CT helps designers express themselves by designing something, rather than only consume technology. Designers who demonstrate the CT perspective of expressing can realize that technological tools are a means of creation. She can say “I can create with computational tools.” The CT perspective of connecting is about realizing the power of creating with and for others. When students recognize that having access to new people through online forums is important, then they can realize that they can do more than they can on their own. They can say “I can do different things when I have access to others.” The CT perspective of questioning involves students questioning the world and on how to make sense of the world by creating computational artifacts.

2.2 Approaches to develop computational thinking

The first definitions of CT emphasized the importance of computational tools- in particular computers- and CT has been associated with programming (e.g. Papert, 1980; Wing, 2006). Therefore, it can be said that computational thinking and coding are related (Bocconi et al., 2016). In this context, coding is seen as a powerful tool to enhance CT (Grover & Pea, 2013), and the development of children's CT is promoted by engaging in design-based learning activities, particularly programming computational artifacts (Brennan & Resnick 2012). Considering the positive impact of coding on CT, many countries have aimed to develop CT by adding programming and Computer Science courses in the K-12 curriculum (Demir & Seferoğlu, 2016).

The notion of engaging children with programming dates back to the time of LOGO programming designed by Seymour Papert and colleagues (Relkin, 2018). LOGO was accepted as the first programming language for children (Fessakis, Gouli, & Mavroudi, 2013; Kazakoff et al., 2012). Papert conducted his studies on teaching LOGO programming to children regarding Piaget's cognitive development in children. According to him, computers are "objects to think with" (Papert, 1980). In other words, children construct their own knowledge when designing, creating (Papert, 1980), and tinkering with computers (Resnick & Rosenbaum, 2013).

Teaching programming to children did not get the attention of schools in those years (Lye & Koh, 2014) because computer programming was considered difficult to teach and learn (Başer, 2013; Tan, Guo, Zheg, & Zhong, 2014); however, the importance of programming in K-12 education again began to be emphasized with the explanations of computational thinking by Jeannette Wing in 2006 (Grover & Pea, 2013) and integrating coding into K-12 education is highly considered more important than ever before (Heintz, Mannila, & Farnqvist, 2016). After LOGO

programming, various programming tools and environments have been developed taking it as a reference. LOGO and LOGO-like programming environments and tools were called “low floor, high ceiling” (Grover & Pea, 2013) which means that the CT tools should have a low threshold for a beginner programmer to easily create working programs (low floor), as well as be enough to meet the needs of proficient ones (high ceiling) (Repenning, Webb, & Ioannidou, 2010). Repenning et al. (2010) claim that CT tools should provide a learning environment supporting transfer, equity, and sustainability. Currently, several computationally rich environments that have these characteristics are designed with the aim of both teaching coding to K-12 students and developing their CT skills (Grover & Pea, 2013).

There are various ways to develop CT and teach basic programming such as of visual block-based programming environments (Lye & Koh, 2014), the use of robotics (Kwon et al., 2012; Sullivan et al., 2013), and unplugged activities (Curzon et al., 2014; Brackmann et al., 2017).

Robotics kits are seen as effective tools for computer programming (Sullivan et al., 2013). They are generally used in preschool and elementary school level because children at these ages are not in the stage of concretization, and so such tools help both concretize computational concepts and motivate them to programming (Demir & Seferoglu, 2017). Students understand the logic of programming while interacting with robotics (Sullivan & Heffernan, 2016). Sullivan and Heffernan (2016) reviewed the literature on programming to children through robotics kits from 1999 to 2004. According to their findings, when children learn programming concepts with robotics, their problem-solving and CT skills are supported.

Oluk and Korkmaz (2016) conducted a study to examine the relationship between students’ computer programming skills in Scratch and their CT levels. They

studied with 31 5th grade students. Students have learned basic programming for six weeks. At the end of the treatment, their computer programming abilities were measured using the Dr. Scratch tool, which helps scoring Scratch projects in terms of CT and gives feedback to improve the computer programming skills, and also students' CT was assessed through Computational Thinking Levels Scale. The researchers found that there is a positive correlation engaging in computer programming through Scratch and developing CT. That is, when students' programming skills increase, their CT levels also develop, or vice versa.

Lee et al. (2011) investigated approaches to teach coding to improve students' CT. For this purpose, they examined three school programs to illustrate how children engage in and develop CT in these three domains: Modeling and simulation, robotics, and game design and development. They suggested that CT skills are developed when children are involved in these rich computational environments. They proposed a three-stage learning progress, Use-Modify-Create, regarding practices in these domains to support CT of children. According to this progression, children firstly use someone else's program or game. Then, children need some modifications to make it more personal. After they acquire some skills and feel confident in themselves, they create their own projects. Based on this approach, children develop the CT practices of reusing and remixing, experimenting and iterating, and modularizing and abstracting.

Werner, Denner, Campe, and Kawamoto (2012) carried out a study aiming at developing CT of 325 middle school students and to assess students' CT performance. They used the Alice programming tool which provides students to create stories in a 3D environment. They used Use-Modify-Create process developed by Lee et al. (2011). Firstly, students were asked to work with some structured

instructional exercises, then they designed their own games. The intervention lasted 20 hours during a semester. Werner and his colleagues (2011) assessed students' CT performance by developing an assessment tool, Fairy Assessment, for Alice programming environments. After analysis of students' works based on Fairy Assessment, it was found that students' scores were high on the tasks that measure comprehension and design, and they have lower scores on the task that requires complex problem-solving abilities. The researchers advocate that students perform well on tasks when they understand the context of the story and instructions in the Alice program. In this regard, they suggested that each task should be designed to help students think algorithmically and to measure specific CT components.

2.2.1 The use of games to develop CT

Games are powerful tools to develop CT because of their motivational and engaging aspects (Kafai & Burke, 2016; Weintrop et al., 2016). There are two popular approaches aimed to develop CT skills and teach basic programming using games (Kazimoglu et al., 2012).

2.2.1.1 Game development approach

One approach for teaching computer programming to children is to create computer games. According to this approach, computational thinking is promoted in computer game design and development (Lee et al., 2011). Scratch (Resnick et al., 2009; Brennan & Resnick, 2012), Alice (Conway et al., 2000; Cooper, Dan, & Pausch, 2000), Stencly (Liu et al., 2014), and Greenfoot (Henrikson & Kölling, 2004) are some environments and tools supporting this approach. The purpose of this approach is to support learning CT concepts such as sequences and loops through developing

games (Denner, Werner, & Ortiz, 2012), rather than teaching how to design and develop a game (Demir & Seferoğlu, 2017), to sharpen computational thinking skills (Kafai & Burke, 2013) and to develop computational practices and perspectives in terms of programming and technology (Brennan & Resnick, 2012) to prepare individuals to skills that will be necessary for future jobs (Bocconi et al., 2016).

2.2.1.2 Game-play approach

The second approach of developing computational thinking is to learn programming concepts by playing games (Demir & Seferoğlu, 2017). People are motivated while playing games (Demir & Seferoğlu, 2017). Utilizing this motivation factor, the games can be used as a constructive learning environment to teach introductory programming concepts (e.g. sequences and events) and develop CT practices of testing and debugging (Kazimoglu et al., 2012). The idea behind this approach is that as users play games, they need to use some programming concepts to complete the levels in the game or develop solutions using CT practices (Kazimoglu et al., 2012). Some of the environments and tools that support this approach are Code.org, Kodable, and LightBot (Demir & Seferoğlu, 2017). Also, Weintrop and Wilensky (2014) called this approach as program-to-play. They claimed that not only this method gives novice programmers to embody their ideas by engaging in computationally rich environments but also helps develop computational literacy skills of young learners, so contributes to raise individuals for the community which are computationally literate.

Weintrop et al. (2016) aimed to explore students' CT practices when they create a video game to play through coding. The researchers created a video game called RoboBuilder and students, aged from 13 to 14 years, played this game. Then, data were collected through interviews with each student, observation, and

examining students' computational artifacts. They asked students to build a strategy and then, to apply it to see whether it works or not. After that, students were asked how they develop their strategy. Thus, researchers had a chance to observe students' CT practices. They found that constructionist video games support to the development of CT practices.

2.2.1.3 Open-ended video games: Sandbox games

Today's students who are growing up with computers, the Internet, and by playing video games (Frand, 2000), called the Net Generation, need digital manipulatives that motivate and encourage them to explore and learn (Annetta, 2008) rather than outdated teaching practices (Prensky, 2000). Video games, popular with the youth today, are one of the best objects to motivate and engage the Net Generation for learning (Shaffer et al., 2005; Tüzün, 2007; Holbert, Penney, & Wilensky, 2010; Kafai & Burke, 2016; Steinkuehler et al., 2012). In this age of technology, educators, policymakers, and researchers argue that video games can be powerful tools for teaching and learning practices (e.g. Gee, 2005; Bourgonjon et al., 2010; Squire, 2013) by taking advantage of the motivational features of video games (Weintrop et al., 2016). The important point here is to ensure that students create computational artifacts within game environments, rather than merely playing the games.

Originated by Seymour Papert, constructionism states that children construct their knowledge by "constructing a public entity" (Papert & Harel, 1991). From a constructionist perspective, learning occurs when students engage with the materials, build something using them, discuss what they create, and reflect on it (Egenfeldt-Nielsen, 2006). Constructionism also highlights the social facets of learning (Kafai, 2006), which means that although children construct artifacts with personal

knowledge, they also share it with others (Kafai & Burke, 2016; Weintrop et al., 2016). Video games, with their personal, social, and cultural aspects, are viewed as natural contexts, which can potentially make the learning environments more constructionist (Kafai & Burke, 2015).

Especially open-ended video games, which are also called sandbox games, promote the constructive process by providing a virtual world and opportunities to explore that world and to create something personal and shareable (Kafai & Burke, 2016). They provide players with virtual worlds that do not have a defined goal and one single correct solution (Squire, 2008). The virtual worlds of video games provide players social practices, situated understanding and develop shared values and collaboration and cooperation skills (Shaffer et al., 2005). That is, video games can provide children with a constructionist learning environment which Papert refers (Kafai, 2005). In this regard, Minecraft, today's one of the most favorable sandbox games among the children (Kafai & Burke, 2016), offers players an open-ended world that enables them to explore and create their unique personal worlds by setting their own goals in the game (Kuhn, 2018).

2.2.1.3.1 Minecraft

Minecraft which is the best-selling sandbox game of recent times (Kafai & Burke, 2016) was sold 180 million copies and has 112 million active users in 2019. Based on this data, it is accepted as one of the most successful video games played by children all around the world (Peckham, 2016; Sarkar, 2017). In such an enormous Minecraft playing community, the players' age range varies from 6-15 (Lane & Yi, 2017). Minecraft is also appealing for both girls and boys (Overby & Jones, 2015; Petrov, 2014).

Although Minecraft was originally produced solely for commercial purposes, it then began to be used in education since 2012 because of its noteworthy impact on children (Lane & Yi, 2017). It is described as a sandbox game which offers players an open world to explore and create their unique worlds by setting their own goals in the game (Kuhn, 2018). In Minecraft, there is no defined goal for players, and they are free to determine their own goals (Ekaputra, Lim, & Eng, 2013). They can create their own virtual worlds by constructing structures or objects through blocks representing the real world.

Minecraft has been related with constructivism approach (Lane & Yi, 2017) which suggests that learners construct their knowledge through creating to and interacting with artifacts (Papert & Harel, 1991). Therefore, Minecraft can be an appropriate learning environment to foster constructivist learning (Brand et al., 2014).

Research on using Minecraft in education is an emerging area hence, it needs more research (Lane & Yi, 2017). But there are some preliminary studies which found that Minecraft-based activities support students to work collaboratively (Karsenti, Bugmann, & Gros, 2017), creativity (Brand, de Byl, Knight, & Hooper, 2014) and engagement (Lane & Yi, 2017). Minecraft used in several different disciplines such as biology, physics, mathematics, and history, with various age groups (Petrov, 2014).

Short (2011) suggested some Minecraft activities to teach scientific concepts in biology, ecology, physics, chemistry, and geography. Similarly, Pusey and Pusey (2015) carried out a study to examine the use of video games in education. The researchers used Minecraft to teach Earth Science to 8th-grade students for five weeks. They observed students and found that their motivation and engagement were

increased after Minecraft-based science activities. Also, according to students' survey results, it was reported that students enjoyed the use of Minecraft in the lessons and their interest in science was increased.

Zorn, Wingrave, Charbonneau, and LaViola Jr. (2013) investigated how Minecraft would encourage students' interest in computer programming. They developed a coding plugin- named CodeBlocks- which offers a programming environment to control a robot in the game (see Figure 1). The researchers studied with students who have little or no experience with coding and they attended in a 20 minutes tutorial section which composed of programming concepts and then completed four tasks using CodeBlocks. It was found that students' interest in coding and robot programming was increased and their perceptions to coding were positively changed.

Figure 1. An example program created with CodeBlocks
Source: [Zorn et al., 2013]

Although there are some studies focusing on using Minecraft to teach coding in the literature, these are very limited and not considering the development of CT skills.

2.3 The role of gender in CT and coding

The American Association of University Women (AAUW) (2010) indicated that female students are minorities in the fields of computing, engineering, and science areas. However, recently, the gender issues in computer programming have been taken into account (Cheng, 2019). Educators, researchers, and policymakers suggest various coding experiences to bring more girls to computing area and to minimize the gender gap (Kelleher, Pausch, & Kiesler, 2007). Physical computing such as robotics (Cheng, 2019) and designing games can provide effective learning environments to increase their interest in computer programming (Çakır, Gass, Foster, & Lee, 2017). As Minecraft is a popular video game for both girls and boys (Petrov, 2014), it can provide a learning environment which is motivating for both groups to engage in coding.

In recent years, the number of studies investigating gender differences in various aspects of programming has increased; however, the results of these studies are contradictory (Cheng, 2019). While some studies found that gender did not affect achievement in programming. For example, Atmatzidou and Demetriadis (2016) examined the development of CT skills of high school students through robotics. They define CT skills as consisting of abstraction, generalization, algorithm, modularity, and decomposition. They found that both male and female students reached the same level of CT skill at the end of the study. However, their results showed that more training time was required for female students to be in the same CT skill level.

Zhong, Wang, Chen, and Li (2015) developed a framework consisting of several computational tasks to assess CT and compared the achievement scores of boys and girls to analyze whether the CT assessment they developed had equal

difficulty level in terms of gender. They reported that there was not any statistically significant difference between the scores of boys and girls. Similarly, in a study aimed to examine the effects of a visual programming curriculum on the development of middle school students' CT skills by Witherspoon et al. (2017), any statistically significant difference was found in the scores of girls and boys.

In a study by Allsop (2019), the gender-based comparison findings showed that there were no differences between the projects of girls and boys in terms of using the CT concepts in two different programming environments, Scratch and Alice. The concept of variables was challenging for girls in Scratch and for both girls and boys in Alice; however, girls had difficulty in loops and abstraction in the Alice programming environment.

There are also studies focusing on the relationship between gender and attitudes toward programming and mostly conducted in higher education. Başer (2013) conducted a study aimed to examine the role of gender on attitudes toward programming and programming success. He studied with 179 university students enrolled in an introductory programming course and found that males feel more confident in programming and have positive attitude toward programming. Therefore, the studies examining the role of gender on programming and the development of CT is limited and the results are not consistent.

2.4 Approaches to assess computational thinking

Giving attention to assessment is important to understand how CT is developed (Grover & Pea, 2013). Various kinds of CT assessments are proposed varying from multiple choice tests to project-based assessments (e.g. Brennan & Resnick, 2012; Werner et al., 2012; Moreno-Leon & Robles, 2015; Roman-Gonzalez et al, 2016).

Existing CT assessments in the current literature mostly intended to analyze computational artifacts designed through programming experiences (Chen et al., 2017), rather than CT practices that students develop.

Werner, et al. (2012) have developed an assessment, called Fairy Assessment, to measure middle school students' understanding of three CT concepts that are algorithmic thinking, abstraction, and modelling. The Fairy Assessment consists of three independent game design tasks with different difficulty levels. The reason why there are three tasks is that if students fail in any task, their scores on other tasks are considered and so, this does not affect their overall evaluation. It was designed to evaluate students' game design projects created in the Alice programming environment. The Fairy assessment has some limitations in terms of construct validity and generalizability. Since it is context specific (Alice-based), it cannot be used in other learning environments.

Similar to the Fairy Assessment for Alice programming tool, Dr. Scratch developed by Moreno-Leon and Robles (2015) is a web-tool that evaluates Scratch projects in terms of seven CT dimensions of abstraction, logical thinking, synchronization, parallelism, flow control, user interactivity, and data representation (Moreno-Leon, Robles, & Roman-Gonzalez, 2016). A Scratch project is uploaded, or the URL of the project is provided to Dr. Scratch. It analyzes the project and gives a score (see Figure 2). If the score is low, the tool determines the learner as a novice, and thus, it gives a feedback on how to improve the code in a basic level. But, as the users become more advanced, the score increases, and the feedback contains all CT dimensions, including some CT dimensions mentioned above (parallelism, flow control, and data representation, abstraction, logical thinking, user interactivity, and synchronization). It gives an expression about how proficient students use CT

concepts in their Scratch projects. Although it gives an extensive analysis of the projects in terms of all the CT dimensions, it is not enough to measure students' knowledge of coding concepts and to reveal their experiences in the process of developing an artifact (Roman-Gonzalez et al., 2016). Also, it is Scratch based, like Fairy Assessment, thus, it could not be used in learning activities using other tools aimed to develop CT.

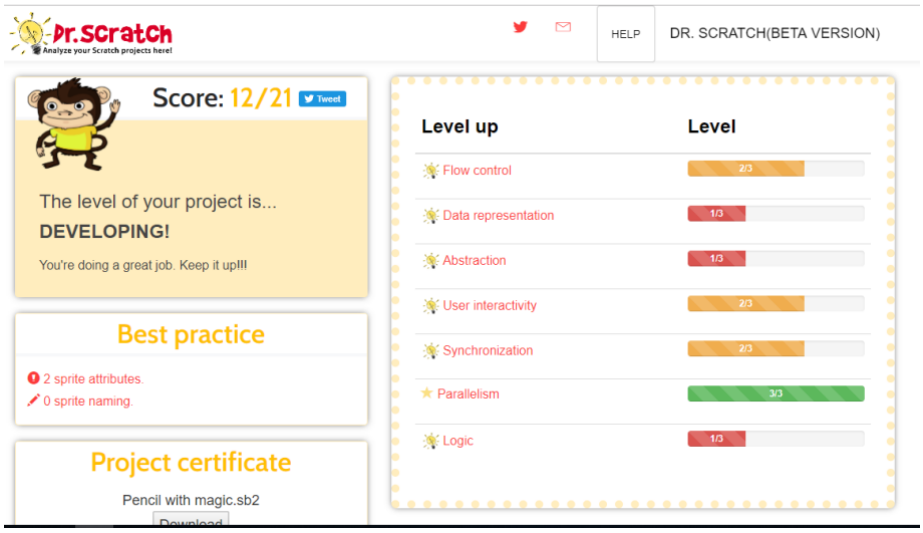


Figure 2. Dr. Scratch analysis

Assessing CT only through a project-based assessment approach is not objective, therefore, instruments need to be designed to provide a more objective assessment of CT which reflects students' conceptual knowledge of coding (Grover, 2014). In developing their framework, Brennan and Resnick (2012) proposed project portfolio analysis as the first approach. In this approach, the Scrape User Analysis tool (see in Figure 3) is used to evaluate the artifacts of students in terms of CT concepts used. This tool is similar to Dr. Scratch mentioned above, but Scrape User Analysis tool analyzes only how frequently a CT concept was used in the Scratch projects. This approach focuses on the complexity of the project and the blocks used in the projects, that is, the development of the use of CT concepts (e.g. loops,

variables). A project is uploaded in the tool, then it analyzes the projects in terms of blocks used in the project with colored representations.

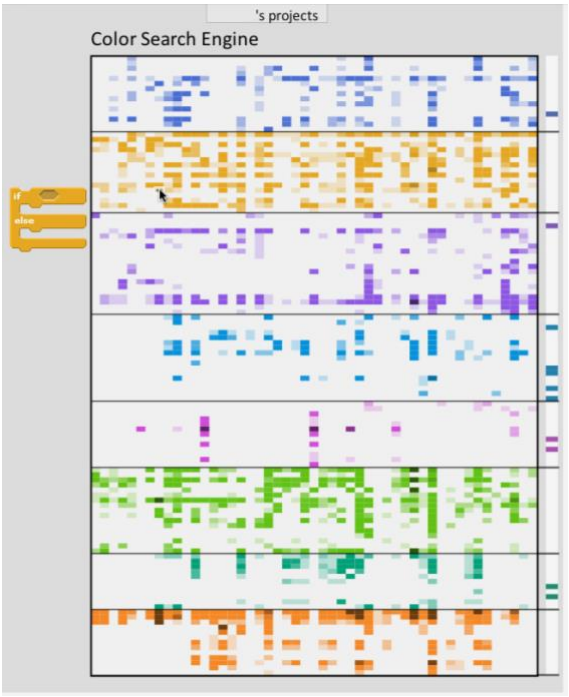


Figure 3. Scrape user analysis visualization of a student's artifact
Source: [Brennan & Resnick, 2012]

Roman-Gonzalez et al. (2016) develop a multiple-choice CT test based on the CT framework by Brennan and Resnick, which focuses on the dimensions of CT concepts (sequences, loops, conditionals, variables, and functions) and CT practice of debugging but does not consider CT perspectives. Grover (2014) also created a test- called the Computational Learning Test (CLT)- to assess knowledge of CT concepts of middle school students. The aim of this test is to measure students' achievement of computing concepts after a learning activity. It includes both multiple choice and open-ended questions about algorithms, loops, conditionals, sequencing, and variables. But Grover (2014) advocates that only using an assessment method remains limited to deeply understand the development of CT. Therefore, she suggested to use "System of Assessments" consisting of various

measurement instruments to assess students' development and learning in a deeper way. This assessment system includes both test and quizzes to measure knowledge of students and interviews with the students about their artifacts to understand their experiences while developing the artifact.

The second form of assessing CT development is artifact-based interview approach. Students are interviewed about their artifacts and these interviews lasted at least an hour. The aim of these interviews is to understand how students developed CT practices, such as testing and debugging strategies. The third one is to develop design scenarios approach, that is, design a set of projects, including two projects with different levels of difficulty. Students are asked to select a project from each set and each project has four tasks. They are expected to figure out what the project does, find the bug in the project and fix it, and extend it by adding new attributes.

According to Brennan and Resnick (2012), there is no single assessment method to measure CT, thus, using these three approaches together can be efficacious for understanding the process of CT development. In this study, CT concepts and practices were examined, excluding CT perspectives. The reason why CT perspectives were not considered is that Brennan and Resnick (2012) reported that three assessment methods (project analysis, artifact-based interviews, & design scenarios) they suggested are not comprehensive to understand CT perspectives of students. In addition, Lye and Koh (2014) conducted a literature review by examining 27 studies to understand which dimensions of Brennan and Resnick's CT framework were examined in the literature and found that while of six studies investigated CT practices, only two of them were at K-12 level. According to these findings, they suggested that it should be conducted more studies to focus on CT

practices in K-12 classes. Therefore, this current study contributes to the research area of CT practices when engaging in coding.

2.5 Summary of the literature review section

As discussed in the literature review section, it is accepted that CT is a way of thinking that needs to be developed by everyone. Many researchers have worked to define CT and classify the characteristics of it. However, researchers generally ended up identifying CT sub-dimensions such as algorithmic thinking, abstraction, and problem-solving instead of describing what CT is. The CT framework developed by Brennan and Resnick (2012) provides a wider perspective to define CT skills as it involves not only the knowledge of CT concepts, but also CT practices, and perspectives.

Since a common definition of CT has not yet been reached, how to develop and measure CT is also open to discussion. Several researchers have developed and examined various methods and approaches to develop CT. Coding has a considerable role in CT development. Many research has conducted to examine how students develop CT as they engage in coding activities. Researchers suggested that block-based programming environments and physical computing tools offer powerful learning environments to teach coding and develop their CT (Brennan & Resnick; 2012; Lye & Koh, 2014; Werner et al., 2012; Sullivan et al., 2013). In addition to these tools, it was found that using constructionist video games promote the development of CT, especially CT practices (Kazimoglu et al., 2012; Weintrop et al., 2016).

Minecraft is a highly preferred video game by many children aged 6-14 all over the world. Studies on Minecraft showed that it increases students' motivation

toward school, creativity, social interaction and self-efficacy (Brand et al., 2014; Lane & Yi, 2017; Slavin, 2015; Walker, 2012). Besides personal development, it also improves students in academic areas. Students develop better understanding of mathematics, science, history, and literacy, and also Minecraft improved students' knowledge of computer programming and develop their CT (e.g. Karsenti, Bugmann, & Gros, 2017).

In addition, diverse assessment methods have been developed to assess the development of CT. These assessment approaches generally focus on measuring students' knowledge through tests and only examining their computational artifacts in terms of coding concepts used without focusing on the practices that students experience.

Coding activities should be designed as to attract both girls and boys (Bruckman, Jensen, & DeBonte, 2002; Linn, 1985). Minecraft is an engaging game for both gender group so that Minecraft-based coding activities can increase both girls' and boys' interest in coding. However, the research on using Minecraft in coding education and CT development is rather sparse.

The purpose of this study was to examine how students develop CT concepts and practices through Minecraft-based coding activities. In accordance with this purpose, this study tried to answer the following research questions:

1. Is coding with Minecraft activities effective in developing CT concepts and practices?
    a) Is there a statistically significant difference between pre-test and post-test scores of students as measured by the CT concept knowledge test?
    b) What are the number and type of CT concepts displayed in the final Minecraft-based computational artifacts?

c) What are the number and type of CT practices displayed by students in the interviews?

2. Do girls and boys differ in the development of CT concepts?
    a) Is there a statistically significant difference between learning gains of female and male students as measured by the CT concept knowledge test?
    b) Is there any difference between boys and girls in terms of number and type of CT concepts displayed in the final Minecraft-based computational artifacts?

Depending on the presented literature, it is expected that CT concept knowledge of students will significantly increase after attending in Minecraft-based coding activities. There will be no statistically significant difference between girls and boys on CT concept knowledge as measured by CT concept knowledge test. Also, it is expected that students will use all CT concepts in their final artifacts. There will be no statistically significant difference in the final artifacts of students considering gender.

CHAPTER 3

METHOD

The purpose of this study is to examine how 5th-grade middle school students develop CT concepts and practices when they engaged in coding with Minecraft. In this chapter, the following sections were covered: (1) research design, (2) the context and participants, (3) data collection instruments, and (4) data collection procedures.

3.1 Research design

The study was employed one group pre-test post-test design (Creswell, 2012), supported with qualitative data (e.g. Chang, Jang, & Chen, 2015). According to the pre-experimental design, the dependent variable, knowledge of CT concepts, was firstly measured before treatment, and after the treatment, the dependent variable was measured again. In addition, qualitative data in terms of student computational artifacts and interviews were collected and analyzed to examine the development of CT concepts and practices.

The independent variable of the study Minecraft-based coding activities. The dependent variable is students' achievements in CT concept knowledge, using CT concepts in their final artifacts, and their CT practices displayed while creating an artifact through coding. The dependent and independent variables of the study are shown in Table 2.

Table 2. The Variables of the Study

Independent Variable	Dependent Variables
Minecraft-based coding activities	Achievement in CT concept knowledge CT concept usage in final computational artifact CT practices employed while creating a computational artifact

3.2 The context and participants

The target population of the study was 5th-grade middle school students in Turkey. The participants were selected using purposeful sampling method (Creswell, 2012) based on the following criteria: (a) being a middle school student, (b) no prior coding experience with Minecraft, or (very limited) prior school-based experience with coding.

The researcher received permission from the İstanbul Ministry of National Education to conduct this study at a public school (see Appendix A). After the permission, she contacted many schools in İstanbul, since the location of the researcher is Istanbul. That is, the accessible population for the researcher was middle schools that satisfy the above criteria in İstanbul. Therefore, after the necessary administrative approvals were obtained, a school close to the campus was selected. One class was chosen randomly among the 5th-grades in the selected school. Since the participating school does not have sufficient infrastructure in terms of the Internet and computer requirements, the study was carried out in a computer laboratory at the researcher's university (Bogazici University) by obtaining necessary permissions.

The participants were twenty 5th-grade students at a public school in Istanbul, Turkey. Of the 20 students, 11 were females and 9 were males. The age range of the

students were 10-11. They do not attend any programming course or after school activities to learn programming. Thus, they do not have any prior knowledge on programming. While the school is located in an affluent neighborhood, the students are children of low-income families.

3.2.1 Minecraft-based instruction

Within the scope of this study, Minecraft was used to teach students coding throughout six weeks and each week was one hour (60 minutes). Throughout the study, students learned coding concepts such as conditionals and loops and also engaged in coding practices such as testing and debugging. They created computational artifacts in Minecraft game environment based on coding concepts which they learned in that week. At the end of the courses, students developed a final project individually including a series of tasks. When they needed help, they got help from their friends or the instructor.

3.3 Data collection instruments

As mentioned before, CT cannot be measured merely by one measurement method, thus systems of assessments should be used to measure CT in depth (Brennan & Resnick, 2012; Grover, 2014). Table 3 provides the instruments used to assess students' knowledge of CT concepts and the development of CT practices. In this context, four sets of data collection methods were used in the present study.

Data collection instrument used to examine each research question is explained in detail below.

Table 3. Data Collection Instruments

Research Question	Instruments	Pre-Intervention	Post-Intervention	Resource
Is there a statistically significant difference between pre-test and post-test scores of students as measured by the CT concept knowledge test?	Computational learning test with Scratch	✔		Grover (2014)
Is there a statistically significant difference between pre-test and post-test scores of students as measured by the CT concept knowledge test?	Computational learning test with Minecraft		✔	Adapted from Grover (2014)
What are the number and nature of CT concepts displayed in the final artifacts? Is there any difference between boys and girls in terms of number and type of CT concepts displayed in the final Minecraft-based computational artifacts?	Final computational artifact analysis		✔	Brennan and Resnick (2012)
What are the number and nature of CT practices displayed by students in interviews?	Artifact-based interviews		✔	Brennan and Resnick (2012)

3.3.1 (Pre) Computational learning test with Scratch

Since one of the aims of the study is to observe the development of computational concepts of students, Computational Learning Test with Scratch (CLT with Scratch) was used to measure concept knowledge of computing. Since there is no Turkish instrument available to measure the knowledge of CT concepts, the test was adapted from Grover (2014). Grover (2014) developed this test by taking the rest items from Ericson and McKlin (2012) and Meerbaum-Salant et al. (2013). It is designed for middle school students (from 5th to 8th grade). The test was translated into Turkish by

consulting two content experts from Bogazici University foreign languages department. In addition, to ensure the content validity of the test, which is the extent to which a test measures an aimed content, three content experts reviewed the test items.

To assess the internal reliability of the translated test, it was administered to 99 5th grade middle school students with similar characteristics with students having the participants of the study. The reliability coefficient for the instrument was found .71, which is considered acceptable internal consistency (Field, 2009).

Before the intervention, students' prior CT concepts knowledge was assessed with the CT with Scratch. The test has 13 items, including 5 questions required textual responses and 8 multiple-choice questions which have 4 answer options. Multiple-choice questions have only one correct answer, and each correct answer for multiple-choice questions is scored 1. Also, some short answer questions have a specific answer so each correct answer for them was scored 1. But students' expressions for some short-answer questions change and these questions get a partial score based on a rubric (see Table 4). If participants answer all questions correctly, they have 36 points. The test can be found in Appendix B and C.

Table 4. Rubric for the Questions Having Partial Score

Question	1	2	3
Concepts- a Concepts- b Concepts- c Concepts- d	Gives an example of the concept.	Explains the concept.	-
Sequences-1b	Writes sequencing the numbers.	Writes sequencing the numbers from the smallest to biggest one.	-
Question 9	Writes drawing a shape.	Writes drawing a rectangle	Writes drawing a rectangle with long edge 15 and short edge 10
Question 10a	Writes drawing a rectangle.	Writes drawing a rectangle according to the short and long edge received from the user.	Writes drawing a rectangle according to the short and long edge received from the user and calculates its circumference and writes it on the screen.
Question 10c Question 10d	Writes only the name of the concept.	Writes only the explanation of the concept.	Writes both the name of the concept and its explanation.

3.3.2 (Post) Computational learning test with Minecraft

After the treatment, to assess students' learning gains of computational concepts, the Computational Learning Test with Minecraft (CLT with Minecraft) was used. The test is parallel to the pre-test described above, but the Scratch questions were adapted to Minecraft. In the adaptation process, Minecraft questions were constructed so that they are structurally parallel to the Scratch questions in the pre-test and met the same learning objective. In this process, three content experts evaluated the questions in terms of the objectives, as in the pre-test (CLT with Scratch).

As the same in pre-test, the post-test has 13 items: eight multiple-choice, and five short-answer (see Appendix D and E). Also, the answers to short answer questions were evaluated with the same rubric as used in pre-test, which means that the scoring is the same as the pre-test. The maximum score from the test is 36 points.

CLT with Minecraft was applied to 111 5th grade middle school students with similar characteristics with students having the participants of the study to assess the internal reliability of the test. The reliability coefficient for the instrument was found to be .75, which is considered an acceptable internal consistency (Field, 2009).

3.3.3 (Post) Final Minecraft-based computational artifact analysis

Students created a final Minecraft-based computational artifact at the end of the treatment. The artifacts were examined to further examine their knowledge of CT concepts development. Final artifact composed of six tasks which were similar to projects in the lessons for six weeks. Students were expected to use all the CT concepts (sequences, parallelism, events, loops, conditionals, operators, and variables), which were covered in the Minecraft-based coding activities. The tasks were briefly explained in Table 5.

Table 5. Final Artifact Tasks

Task	Task Explanation
Task 1	Write a code to control the agent to meet some conditions (e.g. if tp1 is written on chat command, agent teleports to player). The aim of this task is to use the CT concept of conditionals, operators, and variables together.
Task 2	Code the agent so that it can make a 25x25 square-shaped wall. With this activity, it is expected that students use loop which is one of CT concepts.
Task 3	Build a railway using loops.
Task 4	Given code scripts which consist of bugs, find the bugs and fix them. This part was designed to understand which CT practices students demonstrate when they encounter with an error.
Task 5	Use the CT concepts of loops, operators, and variables. Build a code which make any animal appear as much as the number written by the user.
Task 6	Given the codes needed to make a house (with some errors), find the bugs and fix them, also to add codes to form the roof of the house using loops.

In all tasks, students were expected to use events and sequences CT concepts. The tasks were designed by considering as to use all CT concepts which were

covered for 6 weeks and to display CT practices of students. That's why, the tasks included codes that are misplaced and some errors, as Brennan and Resnick (2012) suggested. Students were also asked to find and correct the errors. The details of final artifact tasks can be seen in Appendix F.

3.3.4 (Post) Artifact-based interviews

The aim of artifact-based interviews was to understand students' strategies and approaches while creating a computational artifact. In other words, the interviews were used to investigate the thinking process of students whilst working on an artifact using programming. These interviews were semi-structured and conducted one-on-one with 11 students (7 girls and 4 boys), who were selected randomly. The interview questions were prepared adapting the questions developed by Brennan and Resnick (2012). The interview protocol has six open-ended questions based on the CT practices of experimenting and iterating, testing and debugging, reusing and remixing, and abstracting and modularizing. The artifact-based interview protocol can be seen in Appendix G. The length of interview time varied from 5 to 18 minutes.

3.4 Data collection procedure

Before collecting the data, ethical approval was obtained from the Ethics Committee in Social Sciences and Humanities (SOBETIK) of Bogazici Universtiy (Appendix H). After the participating school was selected, a permission letter (see Appendix I) consisting of the purpose of the study, the importance of participation, and procedure of the study were sent to the parents of students. Students allowed to participate in

the study by their parents attended the study during six weeks in the second academic semester of 2018-2019.

Before the intervention, CLT with Scratch was applied as a pre-test to measure participants' prior knowledge of computational concepts. Because of two reasons, participants were pre-tested in a Scratch context. Students are generally familiar with Scratch and the code blocks in Scratch provides English-like language, as well, so it is understood easily and offers a more comprehensible language in terms of scripts. But the Minecraft coding environment is relatively new, and students need to have an experience to understand the meaning of code blocks, so it can lead to misunderstanding and be confusing for students having some experience with coding. Because of these reasons, the questions of pre-test were designed in the Scratch format.

One week after pre-testing, the Minecraft-based coding program began. The researcher was the instructor of the treatment. The intervention lasted six weeks, and each week contained an hour-long session.

Instructional objectives and Minecraft activities covering six weeks are shown in Table 6. A Minecraft-based coding curriculum for this study was designed by adapting the activities from the M: EE MakeCode curriculum designed by Microsoft. The order of topics was arranged based on the curriculum by Grover (2014). In addition, the learning objectives of the curriculum were prepared by taking into consideration those of the ICT curriculum prepared by the Republic of Turkey Ministry of National Education. The details of the curriculum can be found in Appendix J.

Table 6. Minecraft-Based Coding Curriculum

Time	CT Concepts	Instructional Objectives	Minecraft Activity
1st Week	Sequences, Events, & Parallelism	• Gives real-life algorithm examples • Develops algorithms to create a computer program • Uses events and event handlers	Yellow Brick Road
2nd Week	Loops	• Explains the loop structure and functions • Develops algorithms that include loop structure	Chicken Storm
3rd Week	Loops & Parallelism	• Explains the loop structure and functions • Develops algorithms that include loop structure • Uses variables where required	Building a house and surrounding an area with walls
4th week	Data, Loops & Parallelism	• Distinguishes the difference between constant and variable • Uses variables where required	Chicken Storm with a number determined by the user
5th week	Data, Conditionals, Operators & Parallelism	• Develops algorithms that include decision making	Controlling the agent by the user
6th week	Final Project		

The data collection process of the study is shown in Table 7. At the end of the lessons, students were asked to create a final Minecraft-based computational artifact which included six tasks. The tasks were similar to projects in the lessons. Students were expected to use all the CT concepts that they learned during the coding with Minecraft activities. According to Brennan and Resnick (2012), artifact-based interviews and design scenarios are effective ways to assess computational practices. Therefore, students created projects and were interviewed to observe the development of their computational practices. After the final project week, CLT with Minecraft as post-test was administered to assess students' development of CT concept knowledge.

One week later, interviews were conducted with eleven students who selected randomly. They were asked to discuss the process of developing the project including questions about what strategies they use, what problems they encountered, and how they solved these problems.

Table 7. The Study Procedures

Pre-test	Treatment	Post-test	Interview
CLT with Scratch	Minecraft-based coding activities	CLT with Minecraft	Artifact-based interviews
(Before intervention) (1st week)	(From 2st to 6th week)	(After intervention) (8th week)	(After intervention) (9th week)

3.4.1 The coding environment: Minecraft

During the intervention, Minecraft: Education Edition (M: EE) was used to teach coding. It is a sandbox game aiming at building something by placing blocks. It is also a popular video game played both children and adults, featuring exploration, creativity, entertainment, communication, cooperation, and engineering (Petrov, 2014).

In Minecraft, there is no defined goal for players. They are free to determine their own goals (Ekaputra, Lim, & Eng, 2013). They can create their own world by constructing structures or objects through blocks because of the representing the real world. Therefore, Minecraft can be adopted in education and fits in the game-based learning considering the aspects of motivation, fun, self-determination, and learning by experimenting (Petrov, 2014). Although it was not developed for educational purposes at first, then it was integrated into classroom settings and became a learning tool called Minecraft: Education Edition. It provides students to collaborate on

projects with their classmates, documents their projects and share, and communicate with their classmates and teachers in-game.

Figure 4. Minecraft environment

3.4.2 Code builder plug-in in Minecraft

Code builder plug-in was developed for coding in the Minecraft world and for developing CT of students as well. It can be connected to other block-based coding platforms such as Scratch or Tynker. It is a block-based environment but can be switched to JavaScript (see Figure 5). Only the block-based part was used in this study.

Code builder includes all the CT concepts such as loops, logic, and variables. So, it is appropriate for the CT framework used in this study. Students open the code builder by pressing C on the keyboard. They can demonstrate their actions by coding the Agent in the game.

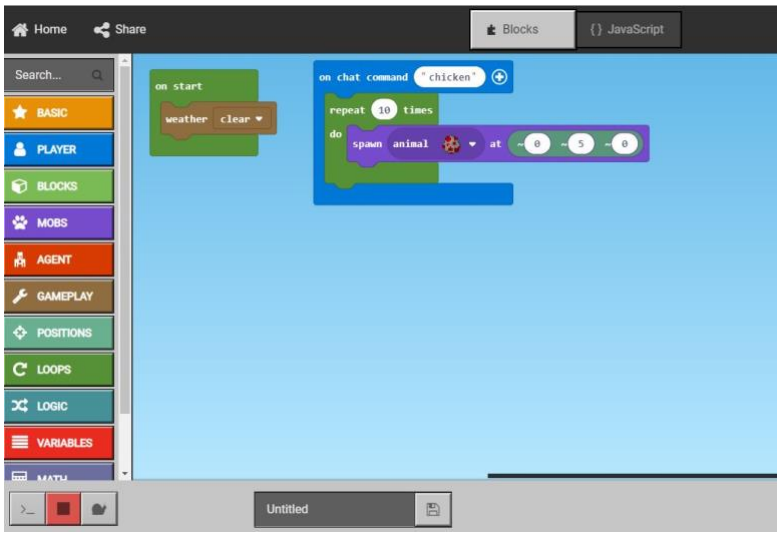


Figure 5. Minecraft code builder screen

3.5 Data analysis

First of all, pre-test and post-test scores of students were calculated, and the quantitative data were analyzed using the IBM Statistics software. Interviews were transcribed and coded based on the CT practices (testing and debugging, reusing and remixing, experimenting and iterating, abstracting and modularizing) by Brennan and Resnick (2012), and the final projects were examined based on the CT concept usage.

Before conducting the hypothesis testing, descriptive statistics and normal distribution of pre-test and post-test scores were examined. Based on this analysis, it was decided which statistical test, parametric or nonparametric, to be used for the research questions 1a and 2a. In the following part, data analysis process for each research question is explained in detail.

The research question 1a (Is there a statistically significant difference between pre-test and post-test scores of students as measured by the CLT?) was answered as follows. Since the pre-test scores were not normally distributed,

Wilcoxon signed-rank test which is a non-parametric test was conducted to analyze students' development of the knowledge of CT concepts.

In order to answer the research question 2a (Is there a statistically significant difference between learning gains of female and male students as measured by the CLT?), learning gains of Minecraft-based coding activities were calculated by subtracting pre-test scores of students from their post-test scores. Checking the parametric test assumptions, the Mann-Whitney U test was conducted to compare the learning gains of boys and girls.

For the research questions 1b (What are the number and nature of CT concepts displayed in the final projects?) and 2b (Is there any difference between boys and girls in terms of number and nature of CT concepts displayed in the final projects?), the final projects of students were examined regarding CT concepts (sequences, parallelism, loops, events, conditionals, operators, and data) used in the projects. The projects were evaluated based on a checklist examining whether a CT concept is existed or not, rather than analyzing how many times a CT concept was used. After that, to analyze which CT concept was used mostly, the frequency of each concept as percentage were calculated. Also, to understand the difference between male and female students in CT concept use, chi-square test was conducted.

The research question 1c (What are the number and nature of CT practices displayed by students in the interviews?) were answered by examining the interview data. The interview data was firstly coded by the researcher. Priori codes were used, and these codes were the CT practices (experimenting & iterating, testing & debugging, reusing & remixing, and abstracting & modularizing) suggested by Brennan and Resnick (2012).

CHAPTER 4

RESULTS

The results chapter includes the analysis carried out for answering the research questions. Data analysis of each research question was reported in a detailed way with both descriptive and inferential statistics.

4.1 Role of coding with Minecraft activities on CT concept knowledge development

4.1.1 Research Question 1a: Is there a statistically significant difference between pre-test and post-test scores of students as measured by the CLT?

The descriptive statistics results showed that the mean CLT scores increased from 3.55 to 12.9 by the end of the Minecraft-based coding activities (see Table 8).

In order to analyze CT concept knowledge development, Shapiro Wilk's test for normality was firstly conducted to check normal distribution of data. Based on the Shapiro Wilk's test' results, although the pre-test scores were not normally distributed ($p < .05$), post-test scores were normally distributed ($p > .05$), shown in Table 9. Pre-test scores of students with skewness of 1.967 (*SE* = .512) and kurtosis of 5.433 (*SE* = .992) and post-test scores were with skewness of .480 (*SE* = .512) and kurtosis of -.811 (*SE* = 992) were shown in Table 8.

Therefore, the non-parametric Wilcoxon signed-rank test was carried to examine the effects of Minecraft-based coding activities on students' CT concept knowledge.

Table 8. Descriptive Statistics of Pre-test and Post-test Scores

	N	Skewness	Kurtosis	Min	Max	Mean	Median	SD
Pre-test	20	1.967	5.433	0	13	3.55	3	2.87
Post-test	20	.480	-.811	5	24	12.9	13	5.94

Table 9. Shapiro-Wilk Result of Pre-test and Post-test Scores

	Shapiro Wilk		
	Statistic	df	Sig.
Pre-test	.825	20	.002
Post-test	.928	20	.142

The Wilcoxon signed-rank test result showed that the post-test scores of students were significantly increased after the Minecraft-based coding activities, z = -3.924, p= .000 (see Table 10 and 11), with a large effect size (r = .87) (Cohen, 1992).

Table 10. Wilcoxon Signed-Rank Test for Pre-test and Post-test

		N	Mean Rank	Sum of Ranks
Posttest Score -Pretest Score	Negative Ranks	0[a]	.00	.00
	Positive Ranks	20[b]	10.50	210
	Ties	0[c]		
	Total	20		

a.Posttest Score < Pretest Score
b.Posttest Score > Pretest Score
c.Posttest Score = Pretest Score

Table 11. Wilcoxon Signed Rank Test Result

	Posttest Score-Pretest Score
Z	-3.924[a]
Asymp. Sig. (2-tailed)	.000

a. Based on negative ranks

4.1.2 Research Question 2a: Is there a statistically significant difference between achievement scores of boys and girls as measured by the CLT?

Descriptive statistics results, showed in Table 12, indicate that boys' mean score (9.44) was slightly higher than girls' mean score (9.27). Scores of girls and boys were normally distributed ($p > .05$), as analyzed by a Shapiro Wilk's test, displayed in Table 13. But since the sample size for each group is small (n for girls = 11; n for boys = 9), Mann Whitney-U test as a non-parametric test was carried out to determine if there were any significant differences in scores between boys and girls.

Table 12. Descriptive Statistics of Boys and Girls

Gender	Mean	Std. Error	95% Confidence Interval	
			Lower Bound	Upper Bound
Girls	9.27	1.83	5.20	13.3
Boys	9.44	2.25	4.24	14.64

Table 13. Shapiro-Wilk Result of Achievement Scores of Girls and Boys

		Shapiro Wilk		
		Statistic	df	Sig.
Learning Gain	Girl	.948	11	.613
	Boy	.906	9	.289

According to the results of the Mann-Whitney U test, it was not found a statistically significant difference in the gain scores between girls and boys, $U = 48$, $p = .909$, shown in Table 15. This result shows that both girls and students benefitted similarly from the Minecraft-based coding activities in terms of learning CT concepts.

Table 14. Mann-Whitney U Ranks

	Gender	N	Mean Rank	Sum of Ranks
Learning Gain	Girl	11	10.64	117
	Boy	9	10.33	73
	Total	20		

Table 15. Mann-Whitney U Result for Comparing Boys and Girls Achievement Scores

	Learning Gain
Mann-Whitney U	48.000
Wilcoxon W	93.000
Z	-.115
Asymp. Sig. (2-tailed)	.909
Exact Sig. [2*(1-tailed Sig.)]	.941[b]
a. Grouping Variable: Gender	
b. Not corrected for ties.	

4.2 CT concepts used in final Minecraft-based computational artifacts

4.2.1 Research Question 1b: What are the number and nature of CT concepts displayed in final artifacts?

The final artifacts of all students (n = 20) were examined and CT concepts used in these projects were identified (see Table 16). Table 16 illustrates the CT concept usage in percentages (if the concept is used at least once).

Based on the results, all the artifacts included sequences and events. In almost all the projects, loops (15 out 20) and parallelism (14 out 20) were used. Variables were used by 35 % of the projects. The use of conditionals and operators was low, at only 30% and 25%.

Table 16. CT Concept Usage Percentage in Artifacts

CT Concepts	Number of Usage	Percentage %
Sequences	20	100
Events	20	100
Loops	15	75
Parallelism	14	70
Variables	7	35
Conditionals	6	30
Operators	5	25

For example, Student 1 used loops in the following manner in his project. He used loops to surround a garden with fences or to build a home in Minecraft (see the example code scripts in Figure 6).

Figure 6. Example code script of Student 1 using loops

As another example, conditionals, variables, and operators were used in Student 2's final artifact to control the robot according to commands written by the user. She used these three concepts together in her artifact, as seen in the Figure 7.

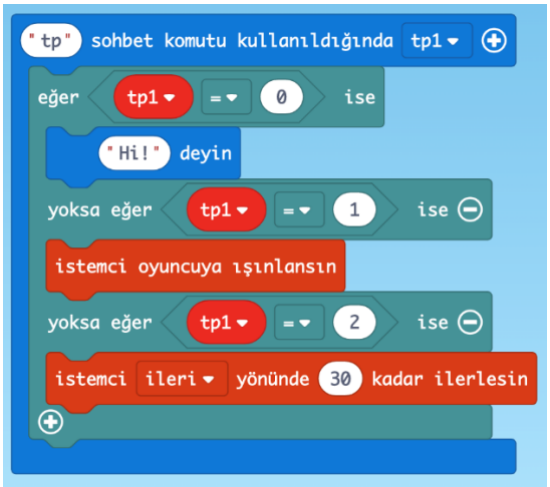


Figure 7. Example code script of Student 2 using conditionals, variables, and operators

4.2.2 Research Question 2b: Is there any difference between boys and girls in terms of number and nature of CT concepts displayed in final projects?

As displayed on Table 17, all boys and girls were able to use sequences and events in their projects. CT concepts including variables, conditionals, and operators were displayed in almost equal number of artifacts of girls and boys.

Table 17. Comparing Girls and Boys CT Concept Usage in Their Final Projects

CT Concept	Girls (N = 11)		Boys (N = 9)	
	Number of Usage	Percentage (%)	Number of Usage	Percentage (%)
Sequences	11	100	9	100
Events	11	100	9	100
Loops	10	83.3	5	62.5
Parallelism	8	72.7	5	55.6
Variables	4	33.3	3	37.5
Conditionals	3	25	3	37.5
Operators	2	16.7	3	37.5

To better understand whether gender is associated with CT concept usage, a Chi-Square test was applied for each CT concept separately. As shown in Table 18, the Chi-Square significance values are bigger than .05 for each CT concept usage;

thus, it was not detected any statistically significant association between for gender and CT concepts used in the final projects.

Table 18. Chi-Square Test Results for CT Concept Based on Gender

CT Concept	Value	Asymp. Sig. (2-sided)
Loops	3.300[a]	0.069
Parallelism	.642[b]	0.423
Variables	.020[c]	0.888
Conditionals	.087[d]	0.769
Operators	.606[e]	0.436

a. 2 cells (50.0%) have expected count less than 5. The minimum expected count is 2.25.
b. 2 cells (50.0%) have expected count less than 5. The minimum expected count is 3.15.
c. 2 cells (50.0%) have expected count less than 5. The minimum expected count is 2.70.
d. 2 cells (50.0%) have expected count less than 5. The minimum expected count is 2.25.
e. 2 cells (50.0%) have expected count less than 5. The minimum expected count is 2.25.

4.3 CT practices displayed by students while creating computational artifacts

4.3.1 Research Question 1c: What are the number and nature of CT practices displayed by participants in interviews?

The interview data were coded based on CT practices offered by Brennan and Resnick (2012), which are experimenting and iterating, testing and debugging, reusing and remixing, and abstracting and modularizing. Table 19 provides these codes, the frequency of each code, and the percentage for each code.

Table 19. Frequency of CT Practices used by Students

Code	Frequency	Percentage (%)
1 Testing & Debugging (TD)	30	51,7%
2 Experimenting & Iterating (EI)	14	24,1%
4 Abstracting & Modularizing (AM)	9	15,5%
3 Reusing & Remixing (RR)	5	8,6%

As shown in the Table 19, the most frequently mentioned CT practice was testing and debugging with 51,7%. Followed by, experimenting and iterating with 24,1%, abstracting and modularizing with 15,5%, and reusing and remixing with 8,6%.

Code: Testing and debugging

Almost all students showed testing and debugging strategies in different ways. Most students mentioned that when they encountered with an error in their scripts, they read the scripts to find out the cause of the error. Here are some representative quotes:

> Student 1: I look where the codes are. If there is something wrong, I try to fix it.
>
> Student 3: I look at the codes for what I did wrong. Then, I try to fix it.

They also tried different solutions to solve this error by making changes in their code scripts. One student explained it as:

> Student 4: I examine all of the scripts to control if there is anything that I misplaced. If there is a script which I did wrong, I extracted it or added another one, then tried it whether it is correct or not.

While these strategies were mentioned by Brennan & Resnick (2012), there were another strategy found in this study, that fell under this category. This strategy is seeking help from someone else (a colleague or teacher) when they encounter with

an error in their code. Most of the students mentioned that they first tried to solve the bug in their scripts by themselves but if they could not, they asked a friend, or a teacher. As one example of this strategy, one student said:

> Student 2: I tried a code myself first. If the code does not work the way I want it, I tried to find out why it did not run over and over again. If I still could not do it, I ask my teacher.
>
> Another student also expressed his debugging strategy as:
>
> Student 5: I try to solve the error myself first. Then I ask one of my friends. If I still could not do it, I finally ask the teacher.

Code: Experimenting and iterating

The second most used CT practice displayed in interviews was experimenting and iterating (24,1 %). Of eleven students, six of them mentioned about experimenting and iterating strategies while creating a computational artifact. The students expressed their EI strategies as an iterative process. They described this process as developing a part of the project and trying it out, and if it does not work, making revisions based on what happens. For instance, Student 6 said that "I did half of the scripts and controlled whether it was correct. Then I did the other half." Student 1 stated that "I did the scripts step by step and run it. For example, I first built the wall scripts and ran it to see if it works. Then I made the fence and tried it." Similarly, Student 7 explained her strategy as she developed her projects' sections one by one, instead of finishing all and then trying them.

Code: Abstracting and modularizing

When asked what they did before they started doing their projects, almost all of the students mentioned developing a plan and building an algorithm before starting their projects and said that they created the code scripts according to these algorithms.

Different from the strategies for this category by Brennan and Resnick (2012), building an algorithm was found another strategy displayed by students as abstracting and modularizing practice. Some examples of students' answers are as follows:

> Student 3: I first created an algorithm before coding.
>
> The interviewer: Based on what did you create your algorithm?
>
> Student 3: According to the program that I want to make. First, I wrote my mind what I would do, then I started to build my project.
>
> Student 7: Before I start coding, I think in my mind and prepare an algorithm like a plan. Then I organize the codes based on this plan and run them.

Code: Reusing and remixing

The least displayed CT practice in interviews was reusing and remixing. Some students preferred to use reusing and remixing strategies when faced with errors, while others used it at the planning stage before starting the project. For example, Student 4 stated that "I first though of the codes in the projects I used before. Then I researched how I can use them." The other student Student 8 mentioned that he looked at the codes he had used before when he had an error.

CHAPTER 5

DISCUSSION AND CONCLUSION

The current study examined the use of Minecraft-based coding activities on middle school students CT concept knowledge and CT practices strategies. The designed Minecraft-based activities were used with 5th-grade students from a public school for six weeks. The first main question of the study sought to answer was whether Minecraft-based activities was effective to develop CT concept knowledge and CT practices of fifth-grade students, who have insufficient opportunities to engage in such activities in their ICT classes. The results showed that students' CT concept knowledge significantly improved after the Minecraft-based coding activities. Also, there were not any statistically significant differences in scores between boys and girls. The analysis of students' final projects indicated that students struggled to use variables, operators, and conditionals. There was no statistically significant difference between girls and boys respecting CT concept use in projects, as similar to the achievement score finding in terms of gender. The interview findings supported to understand which CT practices students use while building a computational artifact. They showed that while students mostly displayed testing and debugging strategies, they used abstracting and modularizing practices the least.

In the following section, it was discussed the findings of each research questions, limitations of the study and further research suggestions.

5.1 Role of coding with Minecraft activities on CT concept knowledge and CT practices

The first research question of the study focused on the effects of Minecraft-based coding activities on the development of CT concept and practices of fifth-grade

students. It addressed assessing CT three approaches for in terms of CT concepts and practices- (1) CT concept knowledge as measured by the pre and post-tests, (2) CT concept usage in the computational artifacts, and (3) CT practices displayed by students while creating an artifact as explored through interviews. The following sections discuss the major findings for the three research questions that guided this main research question.

5.1.1 CT concepts knowledge of students

The analysis of the first sub-question (1a) of the main question showed that students significantly improved their CT concept knowledge. The result is consistent with the claims that teaching coding is effective to develop in fostering CT (e.g. Allsop, 2019; Grover & Pea, 2013; Lee et al., 2011; Lye & Koh, 2014; Weintrop et al., 2016). Many researchers studied students' CT concept knowledge after attending a coding course that teach coding through a block-based environment and they have agreed on the positive effects of these courses on students' CT concept knowledge (e.g. Grover, 2014; Meerbaum-Salant et al., 2010; Roman-Gonzalez et al., 2016).

The findings of the studies conducted with utilizing various programming tools and environments teach programming to children showed that when children engage in these tools and create computational artifacts within these environments, they learn coding concepts and also develop CT practices (e.g. Sullivan & Hefferman, 2016; Brennan & Resnick, 2012; Grover, 2012; Allsop, 2019). Scratch programming environment is seen as a powerful tool for students to facilitate CT development (Weese, 2017; Shute & Clarke, 2017). Not only it is easy to use and but also provides a meaningful learning environment. It was used in many studies which aim to teach programming and to develop CT (Grover, Pea, & Cooper, 2015; Oluk &

Korkmaz, 2016). Alice is another tool mostly used in studies to develop CT and teach programming (e.g. Werner et al., 2012). However, there is not much research aimed to teach coding and develop CT of students using Minecraft-based activities. Therefore, this study contributes to the literature in that way.

In this study, students tried to complete the tasks given to them in Minecraft through coding. So, this study is different from the previous ones in terms of the coding medium used and learning activities. Minecraft has been chosen as a coding environment for this study because it is a game that the age group chosen as participants enjoy playing. As it is known, it has not been used in any studies in the current literature to teach coding and develop CT of children.

This study also showed that the CT framework and measuring methods developed by Brennan and Resnick (2012) for Scratch can also be integrated into other coding environments such as Minecraft.

5.1.2 Using CT concepts in the computational artifacts

While almost all of the students used sequences, events, and loops in their final artifacts, a few of them were able to use the CT concepts of conditionals, operators, and variables. In other words, most of them struggled with the latter ones. The prior research supports these results. Denner et al. (2012) claimed that novice programmers had difficulty and need more guidance and support to use such complex concepts. They found that conditional statements appeared in only 1% of students' games and only 3% of the games included variables. Maloney et al. (2008) examined 425 Scratch projects designed by middle school students and found that certain CT concepts such as operators and variables were the least common concepts in the projects. Similarly, Allsop (2019) evaluated students' games in terms of the

presence of each coding concept in the projects in both Alice and Scratch programming environments. While the least common CT concept which was displayed in the projects of students in Scratch group were operators, students in Alice group had difficulty to use variables in their games. However, students in both groups were able to use other CT concepts such as events and loops in their projects.

All students in the study developed their projects in a correct sequence which is a key concept in programming. This finding is consistent with the finding in Grover (2014). She found that while students easily learned serial execution (sequences), they had difficulties mostly with loops. However, in this study, loops were the second most used concept. The possible reason for this is that students need to use loops in Minecraft most of the time. For example, when they build a house or make a field, they need to repeat a movement more than once. In Minecraft, students need such practices while creating their own worlds' through programming. Therefore, since repeating the same thing more than once is the logic of loop structures, it can be said that Minecraft offers a better learning environment to teach loops.

In addition, it was expected that each project includes the CT concept of events because students must use this concept to run the script prepared through coding with Minecraft. For this reason, it was also examined whether students use the concept of events in a correct way. And, all students used this concept appropriately in their artifacts. Variables, conditionals, and mathematical operators were the least used concepts in the final artifacts. Relevant with the literature, these CT concepts are complex ones for beginners (Grover, 2014). A possible explanation of for this finding might be that tasks concerning conditionals required the manipulation of variables and mathematical operators. That is, students would need

to use these three concepts together so that they could have had difficulty using these concepts in their artifacts. As these are complex concepts, more time can be spent on these concepts to provide students with more experience and practice.

Another mostly used concept in the final projects was parallelism, which refers to the idea that one can code things to happen at the same time. Although parallelism is not typically considered as a major computational concept in the literature, it is listed as one of the seven concepts in Brennan and Resnick's (2012) CT framework. When Minecraft is compared with Scratch which is the environment that Brennan and Resnick (2012) used to found the CT concepts, Scratch has several characters to pick and code to do things at the same time (which can be more intuitive to do), but in Minecraft there is only one main character. Even if the students were able to use this concept correctly in the present study, Minecraft may not be the most effective environment to teach it. In Minecraft, users can make use of the parallelism concept by coding the background and the character do things at the same time. Thus, students may view those programmable objects in Minecraft not ontologically equivalent, which can make the task of learning the parallelism concept more difficult - at least initially. Future research can further investigate the role of the Minecraft environments on the development of the parallelism concept.

5.1.3 CT practices displayed by students while creating computational artifacts

The third sub-question (1c) of the study focused on to understand the CT practices used by students when developing a computational artifact. The interview data were coded based on the CT practices included in CT framework developed by Brennan and Resnick (2012). These practices emerged from the interviews based on the artifacts developed in the Scratch environment, but in this study, it was found that students used the same CT practices when developing Minecraft-based coding

artifacts. In light of these findings, it may be stated that students display similar CT strategies and practices when they engage in coding, regardless of the coding environment.

Additionally, different indicators that students used in the process of creating artifacts were identified which are not explicitly given in this framework. One of the strategies found in this study is to seek help from their friends or the teacher when they encounter an error in their codes. In interviews, almost all students mentioned about this strategy (seeking help from someone) to fix a bug in their code scripts. Similarly, Allsop (2019) found that students worked with peers when they need help and claimed that this motivates students. She observed this strategy as part of the three separate learning behaviors: communication, collaboration, and debugging.

Another strategy that can be included in the CT framework is to develop an algorithm. Students mentioned that they planned what they would do by building an algorithm before starting to coding. The CT practice of abstraction and modularizing is about deciding the organization of scripts. Thus, developing an algorithm for planning may be a strategy for this practice. Reusing and remixing was shown to be the least common CT practice. Based on the CT framework used in the study, reusing and remixing means that students use a part of another project they developed, modify an existing project of others, or get inspired by reading the codes of other projects and trying them out. A few students stated that they used the scripts in their other artifacts, even though some of the tasks given to students for the final project required to use codes similar to those used in the products they created throughout the Minecraft-based coding activities. One of the possible explanations for this result might be that they are at a beginner level in coding. Another explanation for this finding can be that Minecraft does not provide a community

environment for accessing to other projects and for sharing the projects with others to reuse and remix.

5.2 Comparison of students based on gender in terms of CT concept

Another purpose of the study was to compare boys and girls in terms of CT concept and practices. This research question was examined under two sub-issues: (1) Gender-based comparison of CT concept knowledge and (2) comparing boys and girls in terms of using CT concepts in their final artifacts. The following sections discuss the major findings for each of the two research questions that guided the second research question.

5.2.1 Gender-based comparison of CT concept knowledge

Although studies examining the role of gender in programming and CT increased, the issue of gender differences in computer programming needs to be addressed more because the findings related to this topic are not consistent (Shute & Clarke, 2017). In this regard, we examined the gender differences in developing CT. The results showed that there were no statistically significant differences between students' scores regarding gender. These findings support the previous studies focused on gender differences in the development of CT (e.g. Atmatzidou and Demetriadis 2016; Zhong et al., 2015).

However, there are also studies with opposite results. Roman-Gonzalez and his colleges (2016) developed a computational thinking test to assess CT concept knowledge of middle school students, based on the CT framework of Brennan and Resnick (2012). Participants attended to an elective CS course twice a week and were applied the test developed by the researchers at the end of the course. When the

scores of male and female students were compared, the results showed that the males were more successful than the females.

However, this study found out that there is no difference between female and male students regarding the development of CT concept knowledge. The reason could be that Minecraft is fascinating for both girls and boys so that using Minecraft-based coding activities could have motivated both gender group. Bruckman et al. (2002) claimed that when coding activities are designed to encourage both girls and boys, it does not build a gender gap between them. Therefore, using Minecraft in CT development and to teach CT concept knowledge could be an effective learning environment for students by minimizing the gender gap.

5.2.2 Gender-based comparison of final Minecraft-based computational artifact analysis in terms of CT concepts

Although there are limited studies comparing boys and girls in terms of using programming concepts in their computational artifacts are scanty, these studies showed that girls and boys experience similar difficulties in learning programming concepts (Linn & Dalbey, 1985; Murphy et al., 2006; Oluk & Korkmaz, 2016). Similar to previous studies, evidence from this study indicated that there are no significant differences in CT concepts used in the projects of boys and girls.

5.3 Recommendations and implications for future research

The present study contributes to coding education research in the following ways. Firstly, it provides an easy to implement Minecraft-based curriculum to teach students programming. This allows to easily integrate Minecraft into the ICT classrooms in Turkey.

This study is parallel with the majority of the studies in the literature, covering the aims of developing and assessing students' CT. Nevertheless, it differs from the previous ones in terms of the coding environment and the methodological approach to assess CT. The studies focused on various environments or tools to develop CT such as Scratch, Alice, serious games, and robotic kits (Denner et al., 2014; Grover et al., 2015; Atmatzidou & Demetriadis, 2016; Kazimoglu et al., 2012). As far as is known, there is no study using Minecraft sandbox game to teach coding to students. This study providing data about Minecraft-based coding activities with middle school students contributes to the teaching coding and CT development literature. Based on the results, students were good at using events, sequences, and loops in their artifacts therefore, Minecraft can be effective to teach these CT concepts.

The participants of the study were students who did not have any prior knowledge on coding. These students usually need more support for engaging in more complex concepts in their artifacts (Denner et al., 2012). Therefore, as further research, the Minecraft-based activities may be replicated with students with some coding knowledge to examine how they used CT concepts in their artifacts. Consistent with the results of past studies, it clearly seems that students have difficulty grasping the coding concepts such as variable, mathematical operators. On account of this, future research should focus on how to support learners' understanding of such concepts. Also, the transfer of CT skills is another noteworthy issue (Bers et al., 2014; Shute & Clarke, 2017) since computational experiences support students for future computational work (Grover, 2014). Therefore, there should be more studies which examine how students transfer their CT knowledge and use CT practices into a new coding-based learning experience.

During the interviews, students were asked to solve an error in their scripts and think aloud while doing so. Such method provides to observe the cognitive processes of students (Lye & Koh, 2014) and gave information on the testing and debugging strategies. Nevertheless, this approach needs to be expanded. For future studies, the CT practices could be observed by making students to think aloud throughout the entire artifact creation process and also screen recording could be taken while students develop their artifacts so that development of CT could be better understood with the analysis of these data. Moreover, eye-tracking and screen record methods could be used so that students' behaviors can be to be tracked while they create a computational artifact through coding.

5.4 Limitations of the study

Minecraft Education Edition is not accessible to everyone because it is not offered free of charge and only students with an email address with a school extension can use it. At the same time, computers running the Minecraft plugin must have certain hardware features in order to upload the Minecraft on computers. For these reasons, low-income students will have less opportunities to have access to Minecraft-based coding activities.

Another limitation might be the research design of the study. There was only one group and the participants were not randomly assigned to the treatment. Therefore, some internal validity threats may affect the study. In addition, the small sample size can cause statistical power issues. To generalize the study findings to a larger population, the study should be replicated using a true experimental design with more participants.

APPENDIX A

SURVEY AND RESEARCH PERMISSION FROM ISTANBUL MINISTRY OF EDUCATION

T.C.
İSTANBUL VALİLİĞİ
İl Millî Eğitim Müdürlüğü

Sayı : 59090411-20-E.2764873 08/02/2019
Konu : Anket ve Araştırma İzin Talebi

VALİLİK MAKAMINA

İlgi: a) 31.01.2019 tarihli ve 2142532 Gelen Evrak No'lu dilekçe.
b) MEB. Yen. ve Eğ. Tk. Gn. Md. 22.08.2017 tarih ve 12607291/ 2017/25 No'lu Gen.
c) Milli Eğitim Müdürlüğü Araştırma ve Anket Komisyonunun 07.02.2019 tarihli tutanağı.

Boğaziçi Üniversitesi Sosyal Bilimler Enstitüsü yüksek lisans öğrencisi Emine KUTAY'ın **"Minecraft ile Kodlama: Ortaokul Öğrencilerinin Bilgisayarca Düşünme Becerilerinin Gelişimi"** konulu tezi kapsamında, ilimiz genelinde bulunan özel/resmi ortaokullarda öğrenim gören öğrencilere; bilgi-işlemsel düşünme kavramları ön test bilgi-işlemsel düşünme kavramları son test ve programlama ön deneyim anketini uygulama istemi hakkındaki ilgi (a) dilekçe ve ekleri Müdürlüğümüzce incelenmiştir.

Araştırmacının söz konusu talebi; bilimsel amaç dışında kullanılmaması, **uygulama sırasında bir örneği müdürlüğümüzde muhafaza edilen mühürlü ve imzalı veri toplama araçlarının kurumlarımıza araştırmacı tarafından ulaştırılarak uygulanılması, katılımcıların gönüllülük esasına göre seçilmesi, araştırma sonuç raporunun müdürlüğümüzden izin alınmadan kamuoyuyla paylaşılmaması koşuluyla, okul idarelerinin denetim, gözetim ve sorumluluğunda, eğitim-öğretimi aksatmayacak şekilde ilgi (b) Bakanlık emri esasları dâhilinde uygulanması, sonuçtan Müdürlüğümüze rapor halinde (CD formatında) bilgi verilmesi kaydıyla** Müdürlüğümüzce uygun görülmektedir.

Makamlarınızca da uygun görülmesi halinde olurlarınıza arz ederim.

Levent YAZICI
İl Milli Eğitim Müdürü

Ek:
1- Genelge.
2- Komisyon Tutanağı.

OLUR
08/02/2019

Ahmet Hamdi USTA
Vali a.
Vali Yardımcısı

İl Milli Eğitim Müdürlüğü Binbirdirek M. İmran Öktem Cad.
No:1 Eski Adliye Binası Sultanahmet Fatih/İstanbul
E-Posta: sgb34@meb.gov.tr

A. BALTA VHKİ
Tel: (0 212) 455 04 00-239

APPENDIX B

CLT WITH SCRATCH

Student Number:

CT VOCABULARY

Here is a list of concepts. Write a short explanation of each one. If the concept is not familiar, write X.

Algorithm: _____

Variable: _____

Loop: _____

Conditional: _____

Sequence of instructions

There are three playing cards laid out in a row on a table (as shown below); each card is labelled with a number. You are given the following sequence of instructions:

1. compare the number on the left-hand card with the number on the center card
2. if the number on the left-hand card is greater than the number on the center card
   2.1. exchange the two cards
3. compare the number on the center card with the number on the right-hand card
4. if the number on the center card is greater than the number on the right-hand card
   4.1. exchange the two cards

_____

On the table are the following cards:

24	2	15

(1) Write the order of the cards after you carry out the instructions above (in red):

(2) What is the goal of this procedure i.e. what do these instructions achieve?

Sequence of Instructions-2

Here is a sequence of instructions:

1. Turn left
2. Carry out 10 times:
   2.1 Move 5 steps
3. Turn right
4. Carry out 10 times:
   4.1 Move 5 steps
5. Turn right
6. Carry out 10 times:
   6.1 Move 5 steps

(a) If you carry out these instructions, you will follow a path that is the form of some letter in the English alphabet. What is it?

(b) Add more instructions at the end of the list so that the path obtained will be a regular polygon.

(c) What is the length (in steps) of each side?

Scratch-1

What is the above an example of?

a. Conditional execution
b. Handling an event
c. Loop
d. Variable

Scratch-2

How many times will be 

executed above?

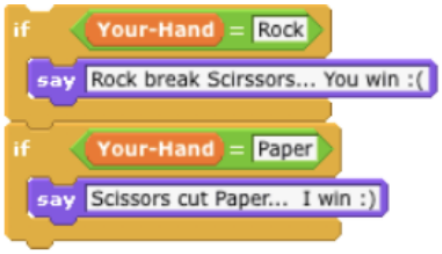
a. 25
b. 20
c. 100
d. 10

Scratch-3

What is the above an example of?

a. Conditional execution
b. Handling an event
c. Loop
d. Variable assignment

Scratch-4



What is the above an example of?

a. Conditional execution
b. Handling an event
c. Loop
d. Variable assignment

Scratch-5

What is this an example of?

a. Conditional execution
b. Handling an event
c. Loop
d. Variable assignment

Scratch-6

What does the following code do?

a. Repeat a simple animation
b. Draw a square using a pen
c. Increment the score
d. Stamp the current costume at the current mouse location

Scratch-7

What does the following code do?

a. Repeat a simple animation
b. Draw a square
c. Increment the score
d. Stamp the current costume at the current mouse location

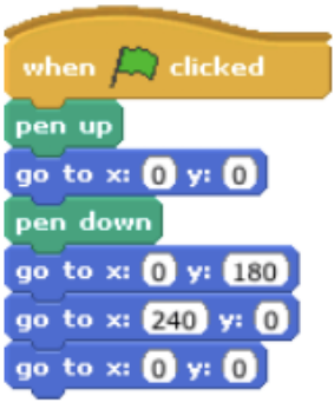
Scratch-8

What will be said when the following executes and the user answers with No?

a. Great!
b. I had better get out of here
c. It won't say anything
d. You will get an error message

Scratch-9

What will be the result of executing the following script? The coordinates of the center of the stage are x=0; y=0.



Scratch-10

(1) Describe in plain English what does the program above does. What is the goal of the program?

(2) What is the term for what the 3 orange blocks in the beginning are doing?

(3) There are 3 yellow blocks in this image above. The FIRST is the Run block (with the green flag). What is the SECOND yellow block doing? What is the term used to describe such a block?

(4) What is the THIRD (last) yellow block in the code doing? What is the term used to describe such a block?

APPENDIX C

CLT WITH SCRATCH (TURKISH)

Öğrenci Numarası:

Kavramlar

Aşağıda verilen dört kavramla ilgili kısa bir açıklamasını yazınız. Eğer kavram hakkında bilginiz yoksa bunun yerine X yazın.

(a) Algoritma: _____
(b) Değişken: _____
(c) Döngü: _____
(d) Koşullar: _____

Yönergeler-1

Aşağıda gösterildiği gibi yan yana üç oyun kartı vardır; her kart bir numara ile etiketlenmiştir. Aynı zamanda şu yönergeler verilmiştir:

1. Sol taraftaki kartın numarasını ortadaki kartın numarasıyla karşılaştır
2. Eğer sol taraftaki kartın numarası ortadaki kartın numarasından daha büyükse
    2.1. iki kartın yerini değiştir
3. Ortadaki kartın numarasını sağ taraftaki kartın numarasıyla karşılaştır
4. Eğer ortadaki kartın numarası, sağdaki kartın numarasından daha büyükse
    4.1. iki kartın yerini değiştir

Kartlar şu şekildedir:

24	2	15

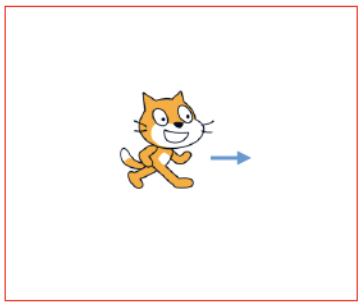
(a) Yukarıda kırmızı ile yazılan yönergeleri sırasıyla uyguladıktan sonra kartların sırasını yazınız.

(b) Uyguladığın bu yönergeler ile neyi gerçekleştirmiş oluyorsun?

Yönergeler-2

Kedi ok yönünde durmaktadır. Kediye aşağıdaki yönergeler verilmiştir:

1) Sola dön
2) 10 defa:
  2.1) 5 adım ilerle
3) Sağa dön
4) 10 defa:
  4.1) 5 adım ilerle
5) Sağa dön
6) 10 defa:
  6.1) 5 adım ilerle



(a) Bu yönergeler yerine getirildiği zaman alfabemizde olan bir harf ortaya çıkar. Bu harf hangisidir?

(b) Çizilen bu harfin düzgün çokgen (kare, dikdörtgen gibi) olması için gereken yönerge veya yönergeleri ekleyiniz.

(c) Ortaya çıkan şeklin bir kenarının uzunluğu kaçtır (adım olarak)?

Scratch

Aşağıda Scratch'ten alınmış ekran görüntüleri ile ilgili sorular yer almaktadır. Soruları ilgili görüntülere göre cevaplayınız.

1. Yanda verilen kod aşağıdakilerden hangisinin bir örneğidir?

A. Döngü
B. Değişken
C. Koşullu ifade
D. Bir olayı yerine getirme

3

2. Yandaki kod çalıştığında kaç defa 

komutu çalışır?

A. 20
B. 10
C. 100
D. 5

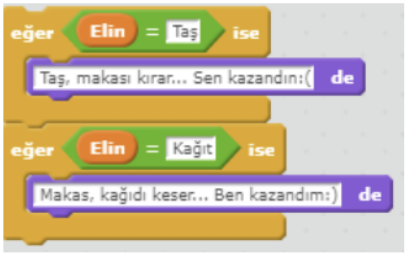
---

3. Yandaki kod aşağıdakilerden hangisinin bir örneğidir?

A. Döngü
B. Değişken
C. Koşullu ifade
D. Bir olayı yerine getirme

---

4. Yandaki kod aşağıdakilerden hangisinin bir örneğidir?



A. Döngü
B. Değişken
C. Koşullu ifade
D. Bir olayı yerine getirme

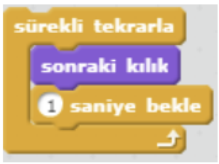
---

5. Yandaki kod aşağıdakilerden hangisinin bir örneğidir?

A. Döngü
B. Değişken
C. Koşullu ifade
D. Bir olayı yerine getirme

---

6. Yandaki kod ne yapar?



A. Basit bir animasyonu tekrarlar
B. Bir kare çizer
C. Skoru arttırır
D. Mevcut kostümün farenin olduğu yerde kopyasının çıkması

4

7. Yandaki kod ne yapar?

A. Basit bir animasyonu tekrarlar
B. Bir kare çizer
C. Skoru arttırır
D. Mevcut kostümün farenin olduğu yerde kopyasının çıkması

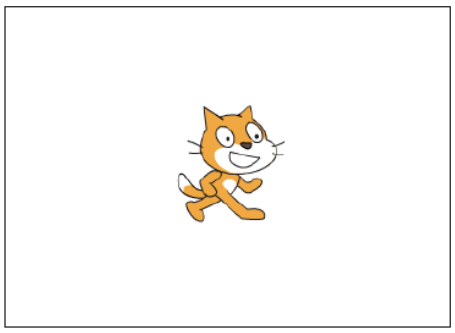
---

8. Yandaki kod çalıştığında ve kullanıcı **H** yanıtını verdiğinde ne söylenir?

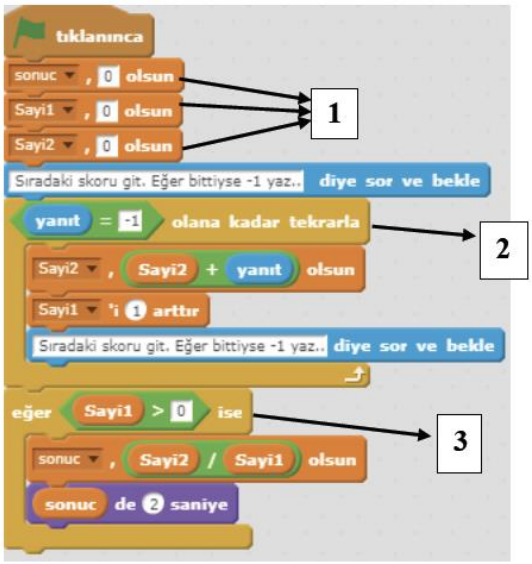
A. Harika!
B. Buradan gitsem iyi olacak
C. Hiçbir şey söylemez
D. Bir hata mesajı alırsın

---

9. Aşağıdaki kod çalıştığında sonucu ne olacaktır? Sahnenin merkezinin koordinatları x = 0; y=0.



10. Soruları aşağıdaki koda göre yanıtlayın.



(a) Bu programın amacını kısaca açıklayınız.

(b) 1 numara ile gösterilen, 3 tane turuncu kod bloğu için kullanılan kavram nedir?

(c) 2 numara ile gösterilen kod blokları için kullanılan kavram nedir? Bu kod bloklarının ne işe yaradığını kısaca açıklayınız.

(d) 3 numara ile gösterilen kod ne işe yarar? Bu tür kod blokları için kullanılan kavram nedir?

6

APPENDIX D

CLT WITH MINECRAFT

Concepts

Here is a list of concepts. Write a short explanation of each one. If the concept is not familiar, write X.

(a) Algorithm: _____
(b) Variable: _____
(c) Loop: _____
(d) Conditional : _____

Sequences-1

There are three playing cards laid out in a row on a table (as shown below); each card is labelled with a number. You are given the following sequence of instructions:

1. compare the number on the left-hand card with the number on the center card
2. if the number on the left-hand card is greater than the number on the center card
   2.1. exchange the two cards
3. compare the number on the center card with the number on the right-hand card
4. if the number on the center card is greater than the number on the right-hand card
   4.1. exchange the two cards

_____

On the table are the following cards:

24	2	15

(1) Write the order of the cards after you carry out the instructions above (in red):

(2) What is the goal of this procedure i.e. what do these instructions achieve?

Sequences-2

The robot stands as in the picture. Some seqeunce of instructions is given to the robot. Here is a sequence of instructions:

1) Turn left
2) Carry out 10 times:
2.1. Move 5 steps
3) Turn right
4) Carry out 10 times:
4.1. Move 5 steps
5) Turn right
6) Carry out 10 times:
6.1. Move 5 steps

(a) If you carry out these instructions, you will follow a path that is the form of some letter in the English alphabet. What is it?

(b) Add more instructions at the end of the list so that the path obtained will be a regular polygon.

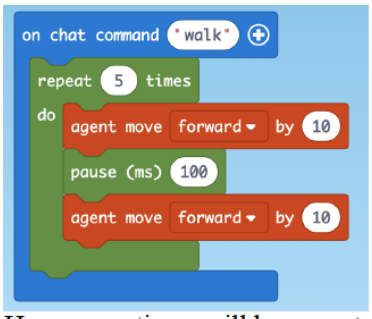
(c) What is the length (in steps) of each side?

Minecraft-1

What is the above an example of?

A. Loop
B. Variable
C. Conditional execution
D. Handling an event

Minecraft-2



How many times will be executed above?

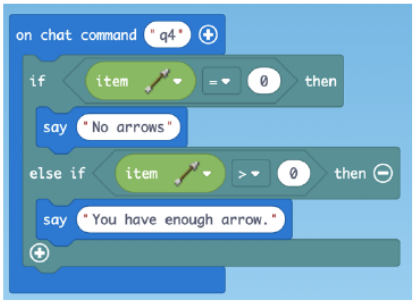
A. 20
B. 10
C. 100
D. 5

Minecraft-3

What is the above an example of?

A. Loop
B. Variable
C. Conditional execution
D. Handling an event

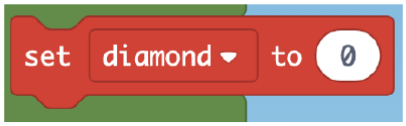
Minecraft-4



What is the above an example of?

A. Loop
B. Variable
C. Conditional Execution
D. Handling an Event

Minecraft-5



What is this example of?

A. Loop
B. Variable
C. Conditional Execution
D. Handling an Event

Minecraft-6

What does the following code do?

A. Repeat a simple animation
B. Chane location of the agent
C. Increment the score
D. Draw a square

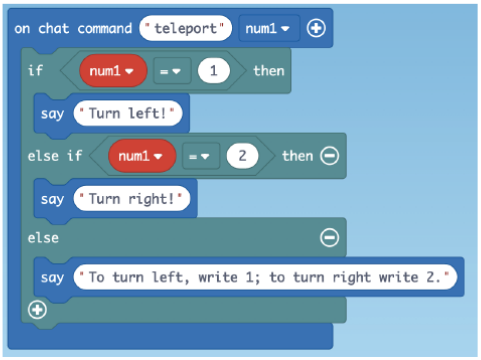
Minecraft-7

What does the following code do?

A. Repeat a simple animation
B. Draw a square shape
C. Increment the score
D. Create a variable

Minecraft-8

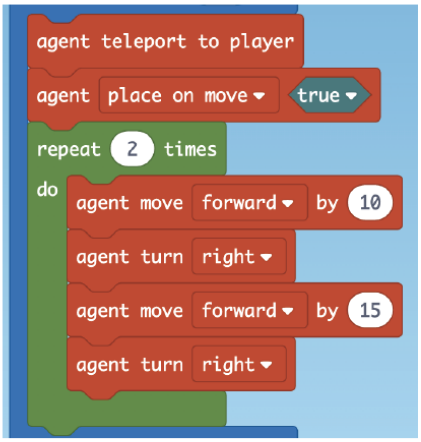
What will be said when the following executes, and the user writes the chat command "teleport 2"?



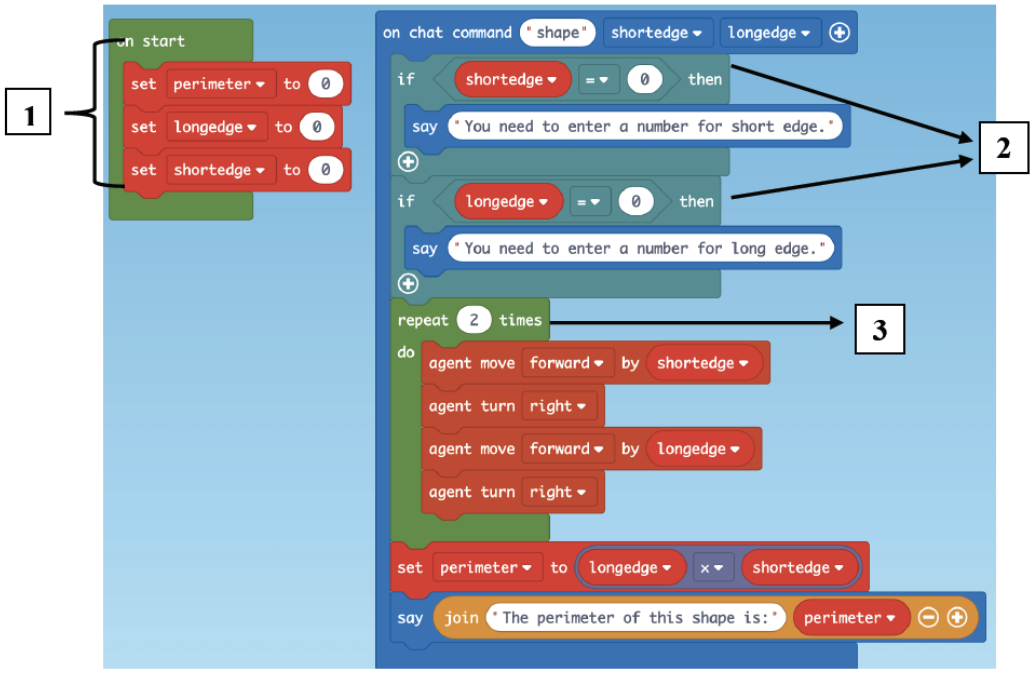
A. Turn left
B. Turn right
C. You should type 1 to turn left, 2 to turn right
D. It won't say anything

Minecraft-9

What will be the result of executing the following script?



Minecraft-10



(a) Describe in plain English what does the program above does. What is the goal of the program?

(b) What is the term for what the 3 orange blocks in the beginning are doing?

(c) What is the name of the coding concept used for blue if-then code blocks denoted as number 2? Briefly explain what these code blocks do.

(d) What does the green code block shown by number 3 do? What is the name of the coding concept used for such code blocks?

APPENDIX E

CLT WITH MINECRAFT (TURKISH)

Öğrenci Numarası:

Kavramlar

Aşağıda verilen dört kavramın kısa bir açıklamasını yazınızı. Eğer kavram hakkında bilginiz yoksa bunun yerine X yazınız.

(a) Algoritma: _____
(b) Değişken: _____
(c) Döngü: _____
(d) Koşullar : _____

Yönergeler-1

Aşağıda gösterildiği gibi yan yana üç oyun kartı vardır; her kart bir numara ile etiketlenmiştir. Sana aşağıdaki şu yönergeler verilmiştir:

1. Sol taraftaki kartın numarasını ortadaki kartın numarasıyla karşılaştır
2. Eğer sol taraftaki kartın numarası, ortadaki kartın numarasından daha büyükse
    2.1. iki kartın yerini değiştir
3. Ortadaki kartın numarasını sağ taraftaki kartın numarasıyla karşılaştır (Yeni oluşan sıraya göre)
4. Eğer ortadaki kartın numarası, sağdaki kartın numarasından daha büyükse
    4.1. iki kartın yerini değiştir

Kartlar şu şekildedir:

24	2	15

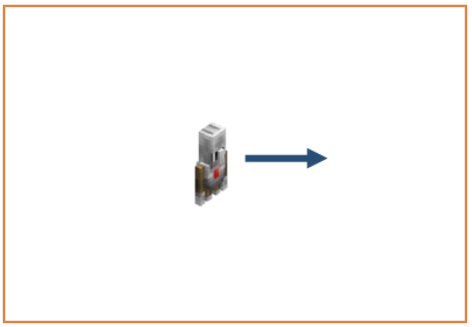
(a) Yukarıda kırmızı ile yazılan yönergeleri sırasıyla uyguladıktan sonra kartların sırasını yazınız.

(b) Bu işlemin amacı nedir? Yani uyguladığın bu yönergeler ile neyi gerçekleştirmiş oluyorsun?

2

Yönergeler-2

Robot ok yönünde durmaktadır. Robota, aşağıdaki yönergeler verilmiştir:



1) Sola dön
2) 10 defa yap:
   2.1) 5 adım ilerle
3) Sağa dön
4) 10 defa yap:
   4.1) 5 adım ilerle
5) Sağa dön
6) 10 defa yap:
   6.1) 5 adım ilerle

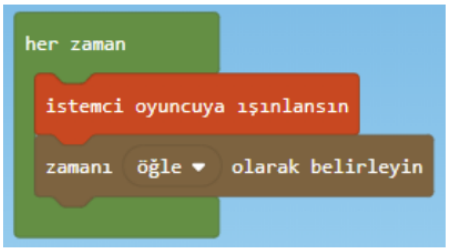
(a) Robota bu yönergeleri yaptırdığında, robot alfabemizdeki bir harf şeklinde bir yol çizer. Bu harf hangisidir?

(b) Çizilen bu harfin düzgün çokgen olması için yukarıdaki yönergelerin sonuna birkaç yönerge daha ekleyiniz.

(c) Ortaya çıkan şeklin bir kenarının uzunluğu kaç kaçtır (adım olarak)?

Minecraft

Aşağıda Minecraft'tan alınmış ekran görüntüleri ile ilgili sorular yer almaktadır. Soruları ilgili görüntülere göre cevaplayınız.



1. Yukarıdaki kod aşağıdakilerden hangisinin bir örneğidir?

   A. Döngü

   B. Değişken

   C. Koşullu ifade

   D. Bir olayı yerine getirme

2. Yukarıdaki koda göre "Yürü" komutu kullanıldığıda, kaç kez 

komutu çalışır?

   A. 20

   B. 10

   C. 100

   D. 5

3. Yandaki kod aşağıdakilerden hangisinin bir örneğidir?

A. Döngü
B. Değişken
C. Koşullu ifade
D. Bir olayı yerine getirme

---

4. Yandaki "if- then" kod bloğu aşağıdakilerden hangisinin bir örneğidir?

A. Döngü
B. Değişken
C. Koşullu ifade
D. Bir olayı yerine getirme

---

5. Yandaki "0 için küme elmas" aşağıdakilerden hangisinin bir örneğidir?

A. Döngü
B. Değişken
C. Koşullu ifade
D. Bir olayı yerine getirme

---

6. Yandaki kod ne yapar?

A. Basit bir olayı tekrarlar
B. İstemcinin konumunu değiştirir
C. Skoru arttırır
D. Kare şeklinde bir şekil oluşturur

7. Yandaki kod ne yapar?

   A. Basit bir animasyonu tekrarlar

   B. Kare şeklinde bir şekil oluşturur

   C. Skoru arttırır

   D. Değişken oluşturur

---

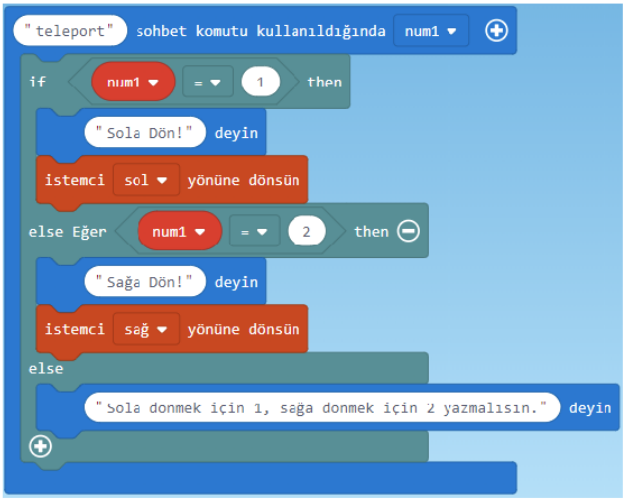
8. Yandaki kod çalıştığında ve kullanıcı kodu çalıştırmak için **"teleport 2"** yazdığında, ekranda aşağıdakilerden hangisi yazar?

   A. Sola Dön

   B. Sağa Dön

   C. Sola dönmek için 1, sağa dönmek için 2 yazmalısın

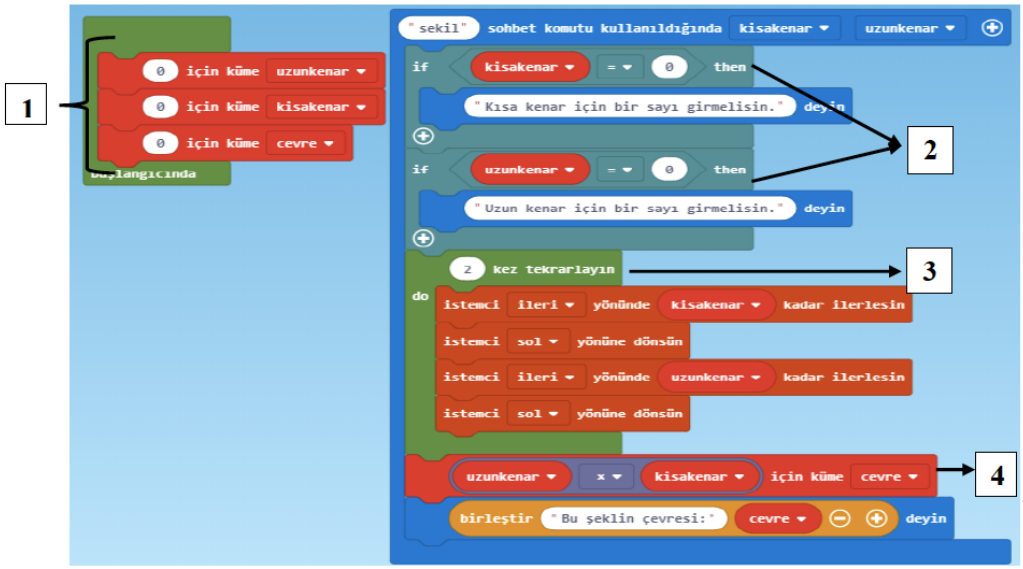
   D. Hiçbir şey yazmaz



---

9. Aşağıdaki kod çalıştırıldığı zaman sonuç ne olacaktır?

10. Aşağıda a, b, c ve d şıklarında verilen soruları, aşağıdaki koda göre yanıtlayınız.



(a) Bu programın amacını kısaca açıklayınız.

(b) 1 numara ile gösterilen, 3 tane turuncu kod bloğu için kullanılan kodlama kavramının adı nedir?

(c) 2 numara ile gösterilen mavi renkli **"if-then"** kod blokları için kullanılan kodlama kavramının adı nedir? Bu kod bloklarının ne işe yaradığını kısaca açıkla.

(d) 3 numara ile gösterilen yeşil renkli kod bloğu ne yapıyor? Bu tür kod blokları için kullanılan kodlama kavramının adı nedir?

APPENDIX F

FINAL MINECRAFT-BASED COMPUTATIONAL ARTIFACT TASKS

Task #	Task Explanation
1	Commands will be given to the agent to go forward, backward, turn right and left, and teleport to the player. Code the agent to meet the given conditions: (a) If "tp 1" is written on the chat command, the agent teleports to the player. (b) If "tp 2" is written on the chat command, the agent moves forward by 1. (c) If "tp 3" is written on the chat command, the agent moves backward by 1. (d) If "tp 4" is written on the chat command, the agent turns left. (e) If "tp 5" is written on the chat command, the agent turns right. (f) If "tp 0" is written on the chat command, a warning message will appear on the screen as follows: 1) Write "tp 1" to teleport the agent to the player; 2) write "tp 2" to move the agent forward by 1 and "tp 3" to move the agent backward by 1; 3) Write "tp 4" for turning the agent right and "tp 5" for turning the agent left. PS: The numbers 1,2,3,4, and 5 will be written by the user, not there will be in the codes.
2	The agent will make a 25x25 square-shaped wall. When "wall" is written on the chat command, the agent will be placed materials from its inventory as it goes forward.
3	Code your agent to build a railway so that you can go from one point to another within the wall you created. When "railway" is written into the chat command, the agent will place material from its inventory as it goes forward.
4	Some codes were given to you for the agent to make an area surrounded with fences. However, there are some errors in the codes. For this reason, the robot cannot create a fenced area. By finding the errors, you will fix the code to make the agent create a square area being 10 blocks of one side surrounding with fences.
5	You created an area that is surrounded by fences in your Minecraft world. Now, you will create a code that allows the number of animals determined by the user to be spawn in this fenced area. For example, when "animal 10" is written in the chat command, 10 animals will spawn. PS: You can choose the type of animal you want.
6	Some codes were given to you for the agent to make a house. There are some errors in the codes needed to make the roof of the house. By finding the errors, you will fix the code to make the agent build the roof.

APPENDIX G

ARTIFACT-BASED INTERVIEW PROTOCOL

Interview Protocol Project: Coding with Minecraft: The development of middle school students' computational thinking

Time of interview:

Date:

Place:

Interviewer:

Interviewee: Emine Kutay

Position of interviewee: One-on-one

1. Bana projenden bahseder misin? (Projenin işlevi, kullanılan kod blokları).
2. Projeni nasıl tasarladın?
    a. Projeni yapmaya başlamadan önce nasıl planlama yaptın? (Düşünme sürecinden, kod blokların organize edilmesi sürecine kadar)
3. Projende kullandığın kod bloklarını ve nesneleri neye göre seçtin?
    a. Kod örnekleri üzerinden anlatabilir misin?
4. Projeni yaparken nasıl problem veya hatalarla karşılaştın? (Denedikleri şeyler, çalışıp çalışmadıkları, bunları nasıl çözdü (problemi araştırması, yardım için başvurdukları))
    a. İstediğin şekilde çalışmayan şeyler nelerdi?
5. Bu problemleri çözmek için neler yaptın?
6. Projene yeni neler eklemek isterdin?
7. Eklemek istediğin bu özelliği nasıl yapabilirsin? (Öğrenciden kodu denemesi istenir.)

APPENDIX H

ETHICS COMMITTEE APPROVAL

T.C.
BOĞAZİÇİ ÜNİVERSİTESİ
İnsan Araştırmaları Kurumsal Değerlendirme Alt Kurulu

Sayı: 48-2018

01 Kasım 2018

Emine Kutay
Bilgisayar Öğretim ve Teknolojileri Eğitimi

Sayın Araştırmacı,

"Coding with Minecraft: The development of computational thinking" başlıklı projeniz ile ilgili olarak yaptığınız SBB-EAK 2018/49 sayılı başvuru İNAREK/SBB Etik Alt Kurulu tarafından 01 Kasım 2018 tarihli toplantıda incelenmiş ve uygun bulunmuştur.

Doç. Dr. Mehmet Yiğit Gürdal

Doç. Dr. Gül Sosay

Doç. Dr. Ebru Kaya

Dr. Öğr. Üyesi İnci Ayhan

Dr. Öğr. Üyesi Nur Yeniçeri

APPENDIX I

PERMISSION LETTER

KATILIMCI BİLGİ ve ONAM FORMU

Araştırmayı destekleyen kurum: Boğaziçi Üniversitesi

Araştırmanın adı: Minecraft ile kodlama: Ortaokul öğrencilerinin bilgisayarca düşünme becerilerinin gelişimi

Proje Yürütücüsü: Doç. Dr. Diler ÖNER (Tez Danışmanı)

E-mail adresi: diler.oner@boun.edu.tr

Telefonu: 0 212 359 73 11

Araştırmacının adı: Emine KUTAY

E-mail adresi: emine.kutay@boun.edu.tr

Telefonu: 0 212 359 67 89

Sayın Veli,

Boğaziçi Üniversitesi Eğitim Teknolojisi Bölümü yüksek lisans öğrencisi Emine Kutay "Minecraft ile kodlama: Ortaokul öğrencilerinin bilgisayarca düşünme becerilerinin gelişimi" adı altında bilimsel bir araştırma projesi yürütmektedir.

Bu araştırmaya yardımcı olmanız için siz velilerin, öğrencinizin araştırmaya katılmasını için onayınızı bekliyorum. Kararınızdan önce araştırma hakkında sizi bilgilendirmek istiyorum. Bu bilgileri okuduktan sonra araştırmaya katılmak isterseniz lütfen bu formu imzalayıp kapalı bir zarf içinde bize ulaştırınız.

Bu çalışmanın amacı Minecraft ile kodlama öğrenmenin öğrencilerin bilgisayarca düşünme becerisi üzerindeki etkisini incelemektir. Çalışma sonunda öğrencilere kodlamanın temel kavramlarının öğretilmesi ve öğrencilerin bilgisayarca düşünme becerilerinin geliştirilmesi hedeflenmektedir. Bilgisayarca düşünme, bilgisayar bilimi ve programlamanın kavramları ve uygulamalarından yararlanarak bir problemi çözme, ürün geliştirme veya sistem tasarımı geliştirmek için gerekli olan bilgi ve becerilere sahip olmak şeklinde tanımlanmaktadır. Teknolojinin gelişimiyle birlikte matematik, biyoloji, tarih ve sanat gibi birçok disiplinde ve iş alanında bu düşünme şekline sahip bireylere ihtiyaç duyulmaktadır. Bu nedenle bilgisayarca düşünme günümüzde erken yaşlardan başlayarak herkes tarafından edinilmesi gereken bir düşünme formu haline gelmiştir. Bilgisayarca düşünme becerisini geliştirmenin en etkili yollarından biri kodlama ya da diğer adı ile programlamadır. Ancak kodlama öğrenciler tarafından öğrenilmesi zor bir alan olarak görülmektedir ve bu nedenle çoğu zaman kodlamaya karşı olumlu bir tutum sergilememektedirler. Birçok araştırmacı video oyunlarının öğrenci motivasyonu ve derse katılım üzerinde olumlu etkilerini vurgulamaktadır. Bu çalışma ile öğrenciler tarafından sevilen Minecraft video oyunu ile öğrencilerin kodlamaya karşı motivasyonlarının artırılması hedeflenmektedir.

Bu araştırmaya katılmayı kabul ettiğiniz takdirde öncelikle öğrencilerin var olan bilgilerini ölçmek amacıyla öğrencilere 13 soruluk bir ön test verilecektir. Bu testin sonuçları hiçbir şekilde öğrencilerin ders veya karne notlarına etki etmeyecektir. İkinci olarak, Bilişim Teknolojileri ve Yazılım dersi müfredatında programlama ünitesi yer almaktadır. Bu bağlamda, öğrenciniz 6 hafta boyunca Bilişim Teknolojileri ve Yazılım dersinde programlamayı öğrenecektir. Bu dersler Türk Milli Eğitim Bakanlığı tarafından hazırlanan müfredatta yer alan kazanımlar doğrultusunda yapılacaktır. Programlamayı öğretmek amacıyla derslerde Minecraft eğitim sürümü kullanılacaktır. Derslere

araştırmacıyla birlikte dersin öğretmeni de girecektir. Üçüncü olarak, araştırmaya katılan öğrencilerden, derslerde araştırmacı tarafından öğretilen konular doğrultusunda projeler yapmaları istenecektir. Araştırmacı, gerektiği takdirde öğrencilerle projeleri ile ilgili olarak yaklaşık 15 dakikalık görüşmeler yapacaktır. Yapılan bu görüşmeler sadece araştırmacı tarafından dinlenmek üzere kayıt alınacaktır. Araştırma sonunda öğrencilerin bu dersler sonunda öğrendiklerini ölçmek için ön teste paralel 13 soruluk bir son test verilecektir. Bu testin amacı öğrencilerin gelişimlerini değerlendirmektir. Bu testlerden aldığı puanlar, öğrencinin herhangi bir notuna etki etmeyecektir. Öğrencilerden toplanan ses kayıtları, testler ve öğrencilerin ürünleri hiçbir şekilde amacının dışında kullanılmayacaktır.

Bu araştırma bilimsel bir amaçla yapılmaktadır ve katılımcı bilgilerinin gizliliği esas tutulmaktadır. Test ve anket sonuçları değerlendirilirken öğrencilerin ismi yerine bir numara kullanılacaktır. Öğrencilerin doldurmuş oldukları test ve anketler araştırma projemiz süresince güvenli bir şekilde muhafaza edilecektir.

Bu araştırmaya katılmak tamamen isteğe bağlıdır. Katıldığınız takdirde çalışmanın herhangi bir aşamasında herhangi bir sebep göstermeden onayınızı çekmek hakkına da sahipsiniz. Öğrenciniz, araştırmadan çekildiği takdirde ders notu vb. gibi herhangi bir nedenden dolayı olumsuz etkilenmeyecektir ve öğrencinizden toplanan veri yok edilecektir. Bu araştırmada farklı okulları veya farklı sınıfları karşılaştırmadığımızı vurgulamak istiyorum. Araştırma projesi hakkında ek bilgi almak istediğiniz takdirde proje yürütücüsü (Ofis Telefonu: 0 212 359 73 11) ile irtibata geçebilirsiniz. Ayrıca araştırma projesi ile ilgili haklarınız konusunda Boğaziçi Üniversitesi Sosyal ve Beşeri Bilimler İnsan Araştırmaları Etik Kurulu'na danışabilirsiniz (Telefon: 0212-359 6810, Adres: Boğaziçi Üniversitesi, Güney Kampüs, 34342 Bebek, İstanbul).

Eğer bu araştırma projesine katılmayı kabul ediyorsanız, lütfen bu formu imzalayıp kapalı bir zarf içerisinde bize geri yollayın.

Yukarıdaki metni okudum ve istenen çalışmanın kapsamını ve amacını tamamen anladım. Velisi bulunduğum, ... adlı öğrencimin çalışmaya katılmasını onaylıyorum. Çalışma hakkında soru sorma imkânı buldum. Bu çalışmayı istediğim zaman ve herhangi bir neden belirtmek zorunda kalmadan bırakabileceğimizi ve bıraktığımız takdirde herhangi bir olumsuzluk ile karşılaşmayacağımızı anladım.

Bu koşullarda söz konusu araştırmaya kendi isteğimle, hiçbir baskı ve zorlama olmaksızın öğrencimin katılmasını kabul ediyorum.

Formun bir örneğini aldım / almak istemiyorum (bu durumda araştırmacı bu kopyayı saklar).

Katılımcının VELİSİNİN Adı-Soyadı: .....

İmzası: .....

Tarih (gün/ay/yıl): ...../...../.....

APPENDIX J

MINECRAFT-BASED CODING CURRICULUM

CT Concepts	**Section #**	**Learning Objectives**	**Student Activity**	**Teacher Activity**
Sequences (Algorithms) Events	S1	To develop algorithms To use events and event handlers	**Unplugged Activity (My Agent Teacher)** Students will try to reach the teacher at the end of the maze by developing an algorithm. **Minecraft Activity (Yellow Brick Road)** Students will create a code that leaves flowers everywhere the player walks by writing its algorithm.	1) Explaining the term algorithm and giving algorithm examples from daily life. 2) Shows the events in MakeCode. 3) Explains the coordinate system in Minecraft
Loops	S2	To explain the loop structure and functions To develop algorithms that include loop structure	**Unplugged Activity** Move around the house. **Minecraft Activity (Chicken Storm)** Students will create a code that makes chickens fall from the sky like a storm of chickens.	1) Explaining the term of iteration and giving examples of actions that are done repeatedly in daily life. 2) Explaining code blocks of loops
Loops	S3	To explain the loop structure and functions To develop algorithms that include loop structure	**Minecraft Activity (Code a Park Fence)** Students will make a code that surrounds an area with fences using loops. **Minecraft Activity (Code a Building)** Students will make a code that builds their own buildings using loops.	Guiding students in developing algorithm process for Minecraft activities.
Data Loops	S4	To define what variable is To use variables where required To develop algorithms that	**Unplugged Activity (The Rhythm Robot)** Creating 3 number variables for 3 movements to tell the robot how many	1) Introducing the concept of a variable and constant and giving some examples from daily life. 2) Showing how to create variables and

CT Concepts	Section #	Learning Objectives	Student Activity	Teacher Activity
		include loop structure	times to do each of these movements. **Minecraft Activity (Chicken Storm with the number determined by the user)** Using a variable to determine the number of chickens to spawn in Minecraft.	use them in MakeCode.
Conditionals Operators Data	S5	To use conditional statements To develop algorithms that include decision making	**Minecraft Activity (How old are you?)** Students will code a program that uses a condition to compare their friends' age with their own, and print different messages if they are younger, older or the same age.	1)Explaining conditionals 2) Drawing conditionals diagram of an example on the board. 3) Giving a series of conditional statements as example. 4) Introducing students with conditional and operators blocks in MakeCode.

REFERENCES

AAUW. (2010). Women and girls in science, technology, engineering, and mathematics. In C. Hill, C. Corbett, & A. St. Rose (Eds.), *Why so few? Women in science, technology, engineering, and mathematics*. Washington, DC: Author.

Allsop, Y. (2019). Assessing computational thinking process using a multiple evaluation approach. *International Journal of Child-Computer Interaction, 19*, 30-55.

Annetta, L. A. (2008). Video games in education: Why they should be used and how they are being used. *Theory into Practice*, *47*(3), 229-239.

Ater-Kranov, A., Bryant, R., Orr, G., Wallace, S., & Zhang, M. (2010, October). Developing a community definition and teaching modules for computational thinking: accomplishments and challenges. In *Proceedings of the 2010 ACM Conference on Information Technology Education (SIGITE' 10)* (pp. 143-148). New York, NY: ACM.

Atmatzidou, S., & Demetriadis, S. (2016). Advancing students' computational thinking skills through educational robotics: A study on age and gender relevant differences. *Robotics and Autonomous Systems*, *75*, 661-670.

Basu, S., Mustafaraj, E., & Rich, K. (2016). CIRCL primer: Computational thinking. *CIRCL Primer Series*. Retrieved from http://circlcenter.org/computational-thinking

Başer, M. (2013). Developing attitude scale toward computer programming. *International Journal of Social Science*, *6*(6), 199-215.

Bers, M. U., Flannery, L., Kazakoff, E. R., & Sullivan, A. (2014). Computational thinking and tinkering: Exploration of an early childhood robotics curriculum. *Computers & Education*, *72*, 145-157.

Bocconi, S., Chioccariello, A., Dettori, G., Ferrari, A., Engelhardt, K., Kampylis, P., & Punie, Y. (2016, June). Developing computational thinking: Approaches and orientations in K-12 education. In *EdMedia: World Conference on Educational Media and Technology* (pp. 13-18). Association for the Advancement of Computing in Education (AACE).

Bourgonjon, J., Valcke, M., Soetaert, R., & Schellens, T. (2010). Students' perceptions about the use of video games in the classroom. *Computer & Education*, *54*(4), 1145–1156.

Brackmann, C. P., Román-González, M., Robles, G., Moreno-León, J., Casali, A., & Barone, D. (2017, November). Development of computational thinking skills through unplugged activities in primary school. In *Proceedings of the 12th Workshop on Primary and Secondary Computing Education (WiPSCE'17*) (pp. 65-72). Nijmegen, Netherlands.

Brand, J. E., de Byl, P., Knight, S. J., & Hooper, J. (2014). Mining Constructivism in the University. In N. Garrelts (Ed.), *Understanding Minecraft: Essays on Play, Community and Possibilities* (pp. 57-75). Jefferson, NC: McFarland & Company, Inc.

Brennan, K., & Resnick, M. (2012). *New frameworks for studying and assessing the development of computational thinking*. Paper presented at the Annual American Educational Research Association meeting, Vancouver, BC, Canada.

Bruckman, A., Jensen, C., & DeBonte, A. (2002, January). Gender and programming achievement in a CSCL environment. In *Proceedings of the Conference on Computer Support for Collaborative Learning: Foundations for a CSCL Community* (pp. 119-127). Boulder, Colorado.

Chang, Y., Jang, S. J., & Chen, Y. H. (2015). Assessing university students' perceptions of their physics instructors' TPACK development in two contexts. *British Journal of Educational Technology*, *46*(6), 1236-1249.

Chen, G., Shen, J., Barth-Cohen, L., Jiang, S., Huang, X., & Eltoukhy, M. (2017). Assessing elementary students' computational thinking in everyday reasoning and robotics programming. *Computers & Education*, *109*, 162-175.

Cheng, G. (2019). Exploring factors influencing the acceptance of visual programming environment among boys and girls in primary schools. *Computers in Human Behavior*, *92*, 361-372.

Ciampa, K. (2014). Learning in a mobile age: An investigation of student motivation. *Journal of Computer Assisted Learning*, *30*(1), 82-96.

Cipollone, M., Schifter, C. C., & Moffat, R. A. (2014). Minecraft as a creative tool: A case study. *International Journal of Game-Based Learning (IJGBL)*, *4*(2), 1-14.

Cohen, J. (1992). A power primer. *Psychological Bulletin, 112*(1), 155.

Conway, M., Audia, S., Burnette, T., Cosgrove, D., & Christiansen, K. (2000, April). Alice: lessons learned from building a 3D system for novices. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems Conference* (pp. 486-493). The Hague, Netherlands.

Cooper, S., Dann, W., & Pausch, R. (2000). Alice: a 3-D tool for introductory programming concepts. *Journal of Computing Sciences in Colleges*, *15*(5), 107-116.

Cordova, D. I., & Lepper, M. R. (1996). Intrinsic motivation and the process of learning: Beneficial effects of contextualization, personalization, and choice. *Journal of Educational Psychology*, *88*(4), 715-730.

Creswell, J. W. (2012). *Educational Research: Planning, Conducting, and Evaluating Quantitative and Qualitative Research* (4th ed.). Boston: Pearson.

Curzon, P., McOwan, P. W., Plant, N., & Meagher, L. R. (2014, November). Introducing teachers to computational thinking using unplugged storytelling. In *Proceedings of the 9th Workshop in Primary and Secondary Computing Education* (pp. 89-92). New York: ACM.

Çakır, N. A., Gass, A., Foster, A., & Lee, F. J. (2017). Development of a game-design workshop to promote young girls' interest towards computing through identity exploration. *Computers & Education, 108,* 115-130.

Demir, Ö., & Seferoğlu, S. S. (2017). Yeni kavramlar, farklı kullanımlar: Bilgi-işlemsel düşünmeyle ilgili bir değerlendirme. *Eğitim Teknolojileri Okumaları, 41*, 468-483.

Denner, J., Werner, L., & Ortiz, E. (2012). Computer games created by middle school girls: Can they be used to measure understanding of computer science concepts? *Computers & Education, 58,* 240-249.

Denning, P. J. (2009). The profession of IT: Beyond computational thinking. *Communications of the ACM, 52*(8), 28–30.

diSessa, A. A. (2000). Changing minds: Computers, learning, and literacy. Cambridge, MA: The MIT Press.

Egenfeldt-Nielsen, S. (2006). Overview of research on the educational use of video games. *Nordic Journal of Digital Literacy, 1*(3), 184-214.

Ekaputra, G., Lim, C., & Eng, K. I. (2013, December). Minecraft: A game as an education and scientific learning tool. In *Proceedings of the Information Systems International Conference, ISICO* (pp. 237-242). Bali, Indonisia.

Ericson, B., & McKlin, T. (2012, February). Effective and sustainable computing summer camps. *In Proceedings of the 43rd ACM Technical Symposium on Computer Science Education* (pp. 289-294). Raleigh, NC.

Fessakis, G., Gouli, E., & Mavroudi, E. (2013). Problem-solving by 5–6 years old kindergarten children in a computer programming environment: A case study. *Computers & Education, 63,* 87–97.

Field, A. (2009). *Discovering statistics using SPSS* (3rd ed.). New York, NY: Sage Publications.

Frand, J. (2000). The information age mindset: Changes in students and implications for higher education. *Educause Review, 35*(5), 15–24.

Garris, R., Ahlers, R., & Driskell, J. E. (2002). Games, motivation, and learning: A research and practice model. *Simulation & Gaming, 33*(4), 441-467.

Gee, J. P. (2003). *What video games have to teach us about learning and literacy*. New York: Palgrave Macmillan.

Gee, J. P. (2005). Good video games and good learning. *Phi Kappa Phi Forum, 85*(2), 33-37.

Grover, S. (2014). *Foundations for advancing computational thinking: Balanced designs for deeper learning in an online computer science course for middle school students.* (Doctoral dissertation, Stanford University, Stanford, CA, United States). Retrieved from Stanford Digital Repository.

Grover, S., Basu, S., Bienkowski, M., Eagle, M., Diana, N., & Stamper, J. (2017). A framework for using hypothesis-driven approaches to support data-driven learning analytics in measuring computational thinking in block-based programming environments. *ACM Transactions on Computing Education (TOCE)*, *17*(3), 1-25.

Grover, S., & Pea, R. (2013). Computational thinking in K–12: A review of the state of the field. *Educational Researcher*, *42*(1), 38–43

Grover, S., Pea, R., & Cooper, S. (2015). Designing for deeper learning in a blended computer science course for middle school students. *Computer Science Education*, *25*(2), 199-237.

Guzdial, M. (2008). Paving the way for computational thinking. *Communications of the ACM*, *51*(8), 25–27.

Gülbahar, Y., Kert, S. B., & Kalelioğlu, F. (2018). Bilgi işlemsel düşünme becerisine yönelik öz yeterlik algısı ölçeği: Geçerlik ve güvenirlik çalışması. *Turkish Journal of Computer and Mathematics Education (TURCOMAT)*, *10*(1), 1-29.

Heintz, F., Mannila, L., & Färnqvist, T. (2016). A review of models for introducing computational thinking, computer science and computing in K-12 education. In *2016 IEEE Frontiers in Education Conference (FIE)* (pp. 1-9). IEEE.

Henriksen, P., & Kölling, M. (2004, October). Greenfoot: Combining object visualization with interaction. In *Companion to the 19th Annual ACM SIGPLAN Conference on Object-Oriented Programming Systems, Languages, and Applications* (pp.73-82). ACM.

Holbert, N., Penney, L., & Wilensky, U. (2010, August). Bringing constructionism to action game-play. In *Proceedings of Constructionism* (pp.1-7). Paris, France.

Hoover, A. K., Barnes, J., Fatehi, B., Moreno-León, J., Puttick, G., Tucker-Raymond, E., & Harteveld, C. (2016, October). Assessing computational thinking in students' game designs. In *Proceedings of the 2016 Annual Symposium on Computer-Human Interaction in Play Companion Extended Abstracts* (pp. 173-179). Austin, TX.

Hu, C. (2011, June). Computational thinking- What it might mean and what we might do about it. In *Proceedings of the 16th Annual Joint Conference on Innovation and Technology in Computer Science Education* (pp. 223-227). Darmstadt, Germany.

ISTE & CSTA (2011). *Computational thinking: Teacher resources* (2nd ed.) [PDF file]. Retrieved from https://id.iste.org/docs/ct-documents/ct-teacher-resources_2ed-pdf.pdf

Kafai, Y. (2005). Constructionism. In R. Sawyer (Ed.), *The Cambridge Handbook of the Learning Sciences* (Cambridge Handbooks in Psychology, pp. 35-46). Cambridge: Cambridge University Press.

Kafai, Y. & Burke, Q. (2013). Computer programming goes back to school. *Phi Delta Kappan, 95*(1), 61-65.

Kafai, Y. B., & Burke, Q. (2015). Constructionist gaming: Understanding the benefits of making games for learning. *Educational Psychologist, 50*(4), 313-334.

Kafai, Y. B., & Burke, Q. (2016). *Connected gaming: What making video games can teach us about learning and literacy*. Cambridge (MA): MIT Press.

Karsenti, T., Bugmann, J, and Gros, P. P. (2017). *Transforming education with Minecraft? Results of an exploratory study conducted with 118 elementary-school students*. Montréal, Canada: CRIFPE.

Kazakoff, E. R., Sullivan, A., & Bers, M. U. (2012). The effect of a classroom-based intensive robotics and programming workshop on sequencing ability in early childhood. *Early Childhood Education Journal, 41*(4), 245-255.

Kazimoglu, C., Kiernan, M., Bacon, L., & MacKinnon, L. (2012). Learning programming at the computational thinking level via digital game-play. *Procedia Computer Science, 9*, 522-531.

Kelleher, C., Pausch, R., & Kiesler, S. (2007, April). Storytelling Alice motivates middle school girls to learn computer programming. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (pp. 1455-1464). San Jose, California, USA.

Kuhn, J. (2018). Minecraft: Education edition. *Computer Assisted Language Instruction Consortium Journal, 35*(2), 214-223.

Kuittinen, M., and Sajaniemi, J. (2004, September). Teaching roles of variables in elementary programming courses. In *Proceedings of the 9th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education* (pp. 57-61). New York, NY, USA.

Kwon, D. Y., Kim, H.S., Shim, J.K., & Lee, W.G. (2012). Algorithmic bricks: A tangible robot programming tool for elementary school students. *IEEE Transactions on Education, 55*, 474–479.

Lane, H. C., & Yi, S. (2017). Playing with virtual blocks: Minecraft as a learning environment for practice and research. In *Cognitive development in digital context*s (pp. 145-166). Elsevier Inc.

Lee, I., Martin, F., Denner, J., Coulter, B., Allan, W., Erickson, J., . . .Werner, L. (2011). Computational thinking for youth in practice. *ACM Inroads, 2*, 32–37.

Linn, M. C., & Dalbey, J. (1985). Cognitive consequences of programming instruction: Instruction, access, and ability. *Educational Psychologist*, *20*(4), 191-206.

Liu, J., Lin, C. H., Wilson, J., Hemmenway, D., Hasson, E., Barnett, Z., & Xu, Y. (2014, March). Making games a snap with Stencyl: A summer computing workshop for K-12 teachers. In *Proceedings of the 45th ACM Technical Symposium on Computer Science Education* (pp. 169-174). Atlanta, Georgia, USA.

Luo, F., Antonenko, P. D., & Davis, E. C. (2020). Exploring the evolution of two girls' conceptions and practices in computational thinking in science. *Computers & Education*, *146*, 103759.

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

Maloney, J., Peppler, K., Kafai, Y., Resnick, M., & Rusk, N. (2008). Programming by choice: Urban youth learning programming with Scratch. *SIGCSE Bulletin*, *40*, 367–371.

Meerbaum-Salant, O., Armoni, M., & Ben-Ari, M.M. (2013). Learning computer science concept with Scratch. *Computers Science Education, 23*(3), 239-264.

Moreno-Leon, J., & Robles, G. (2015). Dr. Scratch: Automatic analysis of scratch projects to assess and foster computational thinking. *Revista de Educación a Distancia, 46*, 1-23.

Moreno-Leon, J., Robles, G., & Roman-Gonzalez, M. (2016, April). Comparing computational thinking development assessment scores with software complexity metrics. In *Proceedings of 2016 IEEE Global Engineering Education Conference (EDUCON)* (pp. 1040-1045). Abu Dahbi.

Murphy, L., Richards, B., McCauley, R., Morrison, B. B., Westbrook, S., & Fossum, T. (2006). Women catch up: gender differences in learning programming concepts. *ACM SIGCSE Bulletin*, *38*(1), 17-21.

Nebel, S., Schneider, S., & Rey, G. D. (2016). Mining learning and crafting scientific experiments: A literature review on the use of Minecraft in education and research. *Journal of Educational Technology & Society, 19*(2), 355-366.

Oluk, A., & Korkmaz, Ö. (2016). Comparing students' scratch skills with their computational thinking skills in terms of different variables. *International Journal of Modern Education & Computer Science*, *8*(11), 1-7.

Overby, A., & Jones, B. L. (2015). Virtual LEGOs: Incorporating Minecraft into the art education curriculum. *Art Education*, *68*(1), 21-27.

Papavlasopoulou, S., Sharma, K., Giannakos, M., & Jaccheri, L. (2017, June). Using eye-tracking to unveil differences between kids and teens in coding activities. In *Proceedings of the 2017 Conference on Interaction Design and Children* (pp. 171-181). Stanford, California, USA.

Papavlasopoulou, S., Sharma, K., & Giannakos, M. N. (2020). Coding activities for children: Coupling eye-tracking with qualitative data to investigate gender differences. *Computers in Human Behavior*, *105*, 105939.

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

Papert, S., & Harel, I. (1991). Situating constructionism. *Constructionism*, *36*(2), 1-11.

Peckham, M. (2016). Minecraft is now the second best-selling game of all time [Web log post]. Retrieved from http://time. com/4354135/minecraft-bestelling

Petrov, A. (2014). *Using Minecraft in education: A qualitative study on benefits and challenges of game-based education* (Unpublished master's thesis, University of Toronto, Ontario, Canada). Retrieved from TSpace Repository.

Piaget, J. (1952). *The origins of intelligence in children*. NY: International Universities Press.

Pusey, M., & Pusey, G. (2016). Using Minecraft in the science classroom. *International Journal of Innovation in Science and Mathematics Education*, *23*(3), 22-34.

Prensky, M. (2000). *Digital game-based learning*. New York: McGraw-Hill.

Relkin, E. (2018). *Assessing young children's computational thinking abilities* (Unpublished master's thesis, Tufts University, Medford, United States). Retrieved from Tufts Digital Library. (b2774686p).

Repenning, A., Webb, D., & Ioannidou, A. (2010, March). Scalable game design and the development of a checklist for getting computational thinking into public schools. 272 In *Proceedings of the 41st ACM Technical Symposium on Computer Science Education (SIGCSE '10)* (pp. 265–269). Wisconsin, USA.

Resnick, M., & Rosenbaum, E. (2013). Designing for tinkerability. In M. Honey & D.E. Hunter (Eds.), *Design, make, play* (pp. 163-181). New York, NY: Routledge.

Resnick, M., Maloney, J., Monroy-Hernandez, A., Rusk, N., Eastmond, E., Brennan, K., … & Kafai, Y. (2009). Scratch: Programming for all. *Communications of the ACM*, *52*(11), 60–67.

Román-González, M., Pérez-González, J. C., & Jiménez-Fernández, C. (2016). Which cognitive abilities underlie computational thinking? Criterion validity of the Computational Thinking Test. *Computers in Human Behavior*, *72*, 678-691.

Sáez-López, J. M., Miller, J., Vázquez-Cano, E., & Domínguez-Garrido, M. C. (2015). Exploring application, attitudes and integration of video games: MinecraftEdu in middle school. *Educational Technology & Society, 18* (3), 114–128.

Sarkar, S. (2017, February 27). Minecraft sales hit 122M copies [Web log post]. Retrieved from https://www.polygon.com/2017/2/27/14755644/minecraft-sales-122m-copies

Shaffer, D. W., Squire, K. R., Halverson, R., & Gee, J. P. (2005). Video games and the future of learning. *Phi Delta Kappan, 87*(2), 105-111.

Shute, V. J., Sun, C., & Asbell-Clarke, J. (2017). Demystifying computational thinking. *Educational Research Review*, *22*, 142-158.

Steinkuehler, C., Squire, K., & Barab, S. (2012). *Games, learning, and society: Learning and meaning in the digital age.* New York: Cambridge University Press.

Sullivan, A., Kazakoff, E. R., & Bers, M. U. (2013). The wheels on the bot go round and round: Robotics curriculum in pre-kindergarten. *Journal of Information Technology Education: Innovations in Practice*, *12*, 203–219.

Sullivan, F. R., & Heffernan, J. (2016). Robotic construction kits as computational manipulatives for learning in the STEM disciplines. *Journal of Research on Technology in Education, 48*(2), 105-128.

Squire, K. D. (2013). Video game-based learning: An emerging paradigm for instruction. *Performance Improvement Quarterly, 21*(2), 7-36.

Tan, J., Guo, X., Zheg, W., & Zhong, M. (2014). Case-based teaching using the laboratory animal system for learning C/C++ programming. *Computer Education*, *77*, 39–49.

Tüzün, H. (2007). Blending video games with learning: Issues and challenges with classroom implementations in the Turkish context. *British Journal of Educational Technology*, *38*(3), 465-477.

Weese, J. L. (2017*). Bringing computational thinking to K-12 and higher education* (Doctoral dissertation, Kansas State University, Manhattan, Kansas).

Weintrop, D., Holbert, N., Horn, M. S., & Wilensky, U. (2016). Computational thinking in constructionist video games. *International Journal of Game-Based Learning (IJGBL), 6*(1), 1-17.

Weintrop, D., & Wilensky, U. (2014). Program-to-play video games: Developing computational literacy through gameplay. In *Proceedings of the 10th Games, Learning & Society Conference* (pp. 264-271). Madison, WI.

Werner, L., Denner, J., Campe, S., & Kawamoto, D. C. (2012, February). The Fairy performance assessment: Measuring computational thinking in middle school. In *Proceedings of the 43rd ACM Technical Symposium on Computer Science Education (SIGCSE '12)* (pp.215-220). New York, NY: ACM.

Wing, J. M. (2006). Computational thinking. *Communications of the ACM, 49*(3), 33-35.

Wing, J. M. (2008). Computational thinking and thinking about computing. *Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences, 366*(1881), 3717-3725.

Wing, J. M. (2011, March). Research notebook: Computational thinking-- What and why? *The Link Magazine, 6*, 20-23. Retrieved from https://www.cs.cmu.edu/link/research-notebook-computational-thinking-what-and-why

Witherspoon, E. B., Higashi, R. M., Schunn, C. D., Baehr, E. C., & Shoop, R. (2017). Developing computational thinking through a virtual robotics programming curriculum. *ACM Transactions on Computing Education (TOCE), 18*(1), 1-20.

Voogt, J., Fisser, P., Good, J., Mishra, P., & Yadav, A. (2015). Computational thinking in compulsory education: Towards an agenda for research and practice. *Education and Information Technologies, 20*(4), 715-728.

Zhong, B., Wang, Q., Chen, J., & Li, Y. (2016). An exploration of three-dimensional integrated assessment for computational thinking. *Journal of Educational Computing Research, 53*(4), 562-590.