Universidad de Deusto

An analysis of computational thinking development through personalized learning paths of programming challenges in a block-based maze game by measuring learners' performance and challenge difficulty.

by

Ioanna Kanellopoulou

A dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy within the PhD Program in Engineering for the Information Society and Sustainable Development.

Supervised by Dr. Pablo Garaizar Sagarmínaga and Dr. Maria Luz Guenaga Gómez

Universidad de Deusto

An analysis of computational thinking development through personalized learning paths of programming challenges in a block-based maze game by measuring learners' performance and challenge difficulty.

by

Ioanna Kanellopoulou

A dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy within the PhD Program in Engineering for the Information Society and Sustainable Development.

Supervised by Dr. Pablo Garaizar Sagarmínaga and Dr. Maria Luz Guenaga Gómez

The candidate

The supervisors

Bilbao, July 2022

An analysis of computational thinking development through personalized learning paths of programming challenges in a block-based maze game by measuring learners' performance and challenge difficulty.

Author: Ioanna Kanellopoulou
Supervisor: Pablo Garaizar Sagarmínaga
Supervisor: Maria Luz Guenaga Gómez

This work has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Sklodowska-Curie grant agreement Nº 665959. In addition, a number of institutions back and co-finance this project.

Text printed in Bilbao
Second edition, July 2022

To my father.

Abstract

In a world where algorithms are ubiquitous, the development of computational thinking is becoming progressively important among students, technology professionals, and 21st-century citizens in general. Computational thinking has gained importance in the scientific and educational communities over the last decade, and it has been advocated as a fundamental competence that should be included in the compulsory educational curriculum. In addition, several reports have shown that the supply of computer science professionals is not meeting demand in the technological sector. Research has indicated that this problem could be overcome by promoting, from an early age, computational thinking skills that are closely related to computer science and programming. Thus, there is a growing trend to include computational thinking in primary education worldwide due to its many benefits. Educational games as a means of promoting computational thinking have been widely used in recent years. Game-based learning is a type of gameplay with defined learning outcomes that has the potential to provide effective learning experiences for players by including strategies for learning and engagement.. According to research, visual programming, and in particular block-based programming environments, play an important role in introducing K-12 students to the fundamental principles of programming and the computer science world,. For this reason, many block-based games have been developed, and important initiatives to promote computational thinking at a local and international level are based on them. Their broad use increases the need to offer efficient block-based programming environments. This can be achieved by providing block-based programming environments with personalized learning paths through adaptive difficulty.

The investigation presented herein is focused on offering an adaptive game that offers programming challenges of adapted difficulty based on the learners' performance. In the course of our research, we proposed an innovative way to define the difficulty of maze-based programming challenges using log data obtained from Kodetu, a block-based maze game. Specifically, we conducted three studies with 9- to 16-year-old learners who were asked to solve sequences of maze-based programming challenges. Using log data from these studies, we investigated the maze characteristics and the coding limitations that affected performance in the challenges and calculated the performance obtained by the participants using a fuzzy rule-based

system. The results showed that the turns in a maze, the total number of steps of a maze, and the blocks provided affect student performance. Using regression analysis, we defined a difficulty function for maze-based programming challenges that considers the weights of these factors and provides a first step towards the design of adaptive learning paths for computational thinking-related educational games.

Having defined the difficulty, we were able to develop the adaptive version of Kodetu, following the approach of the computerized adaptive testing systems. A set of 110 programming challenges was created, comprising the challenge bank of the adaptive Kodetu. These challenges belong to three main categories according to the necessary blocks needed to solve them: sequential, loop, and conditional. The learning path on the adaptive Kodetu is personalized; i.e., depending on how well the learners performed on the previous challenge, the next challenge is provided based on that performance, with the difficulty level dropping or growing. The data obtained from the experiment performed with 9- to 11-year-old learners showed that the adaptive Kodetu is more effective than the non-adaptive version. The comparisons made between the data from the experiments with the adaptive and the non-adaptive Kodetu show that the learners perform better when using the adaptive Kodetu even when the learners playing with the non-adaptive are older. In the sequential and conditional challenge categories, the learners are able to solve difficult programming challenges in less time and with less effort, allowing them to continue playing and developing computational thinking.

In summary, this investigation presents a novel way to measure the difficulty of block-based maze games and demonstrates the efficiency of learning paths consisting of programming maze-based challenges that adapt the difficulty to the learners' performance. The proposed difficulty function helps teachers and educational stakeholders to personalize learning paths based on their students' needs. The use of the difficulty function to develop the adaptive Kodetu and the promising results obtained lead the way to the development and implementation of efficient automated adaptive tools that could be integrated into the curriculum of K-12 education to effectively promote computational thinking.

Acknoledgements

Finally, the most pleasant moment of the PhD has arrived. This work would have never been completed if I hadn't had the help of a lot people, and it's time to thank them. First of all, I would like to thank my supervisors, Pablo Garaizar Sagarmínaga and Maria Luz Guenaga Gómez who have been guiding, helping, and challenging me throughout the whole PhD. We have walked a difficult path together over the years, and it is time to enjoy it. Thank you for all that I have learned from you.

Next, I would like to thank the former and current members of the LearningLab who from the very first moment welcomed me with a smile and were always ready to answer every question I had and help me with eagerness. Thank you Andoni, thank you Olga, thank you Pablo Espeso!

I would like to say a huge thank you to my Basque family. I am one of the few extremely lucky people who have not only one, but two families. Thank you Zarate Gonzalez family, Iñaki, Sylvia, Oihane, Iñakitxu for being by my side during one of the most difficult periods of my life, and with your tenderness you helped me continue the work to complete this PhD.

It is now time to officially thank three people that were instrumental in achieving this PhD. Ekaitz, Iratxe, Oihane (again), I don't think there are words to thank you and show my gratitude. From the first day I met you, until today you have been by my side through the easy and the hard, the pleasant and the unpleasant. I hope to be able to show you one day how grateful I am to you. Thank you...

Many thanks to my friends in all corners of the world, Lia, Villy, Ioanna, Maria, Michalis, Ane, Aiur, Paloma, Niamh, Leire, Edurne for sharing this experience for so long either online or in person.

Of course, this PhD would never have happened without my family. Thank you so much, Yianni, Alexandre, Theoni, Nefeli, Marianna, Eleni, Yiorgo, Pano, Harry, and Oleia. You have always given me courage, a smile, and support to keep going. Mostly, I want to thank my parents, Yiorgo and Eleni who have been by my side with unlimited

love and support. You have been with me from the first day I left to start this journey until the end. You are the strongest people I have ever known. Thank you for everything.

Finally, I want to thank Antreas, without his infinite and unconditional support this PhD would not exist. Thank you from the bottom of my heart for everything. Thank you for making me feel that together we can achieve anything. This thesis is yours too.

Ithaka
BY C. P. CAVAFY
TRANSLATED BY EDMUND KEELEY

As you set out for Ithaka hope your road is a long one,
full of adventure, full of discovery.
Laistrygonians, Cyclops,
angry Poseidon—don't be afraid of them:
you'll never find things like that on your way
as long as you keep your thoughts raised high,
as long as a rare excitement stirs your spirit and your body.
Laistrygonians, Cyclops,
wild Poseidon—you won't encounter them
unless you bring them along inside your soul,
unless your soul sets them up in front of you.

Hope your road is a long one.
May there be many summer mornings when,
with what pleasure, what joy,
you enter harbors you're seeing for the first time;
may you stop at Phoenician trading stations to buy fine things,
mother of pearl and coral, amber and ebony,
sensual perfume of every kind—
as many sensual perfumes as you can;
and may you visit many Egyptian cities
to learn and go on learning from their scholars.

Keep Ithaka always in your mind.
Arriving there is what you're destined for.
But don't hurry the journey at all.
Better if it lasts for years, so you're old by the time you reach the island,
wealthy with all you've gained on the way,
not expecting Ithaka to make you rich.

Ithaka gave you the marvelous journey.
Without her you wouldn't have set out.
She has nothing left to give you now.

And if you find her poor, Ithaka won't have fooled you.
Wise as you will have become, so full of experience,
you'll have understood by then what these Ithakas mean.

Table of Contents

List of Figures

List of Tables

Acronyms

CT	Computational thinking
STEM	Science, technology, engineering, & mathematics
ISTE	International society for technology in education
CSTA	Computer science teachers association
CAS	Computing at school
T&L	Teaching and learning
GBL	Game-based learning
DGBL	Digital game-based learning
AHS	Adaptive hypermedia environments or system
AH	Adaptive hypermedia
AEHS	Adaptive educational hypermedia system
ICT	Information and communication technologies
CAT	Computerized adaptive testing
P&P	Paper-and-pencil
CTQ	Computational thinking quest
VCAA	Victorian curriculum and assessment authority
AI	Artificial intelligence
IQ	Intelligence quotient
FRBS	Fuzzy rule-based system
FL	Fuzzy logic
MF	Membership function
COG	Center of gravity
MAE	Mean absolute error
AST	Average success time
ASt	Estimated average success time
ANA	Average number of attempts
ANI	Average number of interactions

Chapter

1

Introduction

1.1. Motivation & Scope

The great increase in the use of technology in everyday life during the last years could not leave the field of education unaffected. The COVID-19 pandemic has highlighted, more than ever, the relevance of technological literacy for stakeholders in education. Homeschooling, online tutoring, and the use of digital tools are some of the challenges that students, teachers, and families have had to overcome. In many sectors, the growing need for technology has increased the demand for computer science professionals (Computer and Information Technology Occupations, 2020; ICT Specialists in Employment - Statistics Explained, 2020.). In particular, computational thinking (CT) is one of the key skills of computer scientists; it is also valuable for professionals in other fields. When Wing first introduced the concept to the scientific community, she referred to CT as a fundamental skill that every person should develop in order to perform in modern society. Therefore, it is necessary to promote and support CT through appropriate educational tools, methodologies, and strategies (Wing, 2006).

Many initiatives (Scratch Day, Hour of Code, AI Leagues) and tools (Scratch, Alice, Blockly, App Inventor) have been created to develop CT. The vast majority of

these tools use block-based programming features that make programming more approachable for novice learners (Jeon & Song, 2019). Block-based programming tools typically feature a large number of commands to pick from with descriptive names, which eliminates the need to memorize commands and makes it simple for novice learners to begin coding. Regarding the type of activities, many of them present maze-based programming challenges (Bontchev & Panayotova, 2018; Koupritzioti & Xinogalos, 2020; Ternik et al., 2017), where the learners' goal is to guide a character towards the exit of a maze by using the blocks provided. Most of them include fixed challenges that are independent of the user's performance during the activity. A focus on adapting challenges aids motivation and facilitates reaching the games' full educational potential, a significant factor in positive learning outcomes (Hooshyar et al., 2021). Therefore, a critical aspect being considered is the possibility of adapting the learning process to personalize the learners' needs and adapt to their progress so that they accomplish the most effective learning outcomes (Tlili et al., 2019). According to the flow theory, which has been the basis of contemporary adaptive game design, providing adequate difficulty scaffolding related to the learners' abilities is key to offering challenges that are neither too difficult nor too easy for their competencies, consequently avoiding anxiety or boredom, respectively (Gallego-Durán et al., 2018). To this end, we must be able to calculate the difficulty of these activities so that teachers—or, ideally, the proper tool—can generate increasingly challenging learning activities with adaptive difficulty based on the learner's performance.

Considering the importance and relevance of CT in society, and the need for adaptive CT games, our interest was to provide a maze-based online system to develop CT using block-based programming that provides learners with adaptive learning paths (Angeli, 2022; Hooshyar, Pedaste, et al., 2021). However, the lack of a clear definition of the difficulty in this type of educational maze-based game led us to first define how we can measure the difficulty of such activities. Once we had a way to automatically define the difficulty of maze-based programming challenges, we addressed the design and development of the adaptive block-based maze game. Throughout this thesis, we will explain the challenges we have encountered along the way and how we have solved them to achieve this goal.

1.2. Hypothesis and research questions

The primary aim of the thesis was to develop an adaptive CT learning tool addressed to novice programmers between 8 and 16 years old that contributes to the development of CT. The final result was an adaptive block-based maze game that provided programming challenges of difficulty adapted to the learners' performance. Based on the above, the particular objectives we set are as follows:

1. Determine the variables of a block-based maze game that affects performance and, therefore, the difficulty of the game. This type of programming environment consists of the maze, blocks, and workspace. Any block can be selected and dropped into the workspace in order to guide the main character through the maze from the beginning to the final destination. For the purposes of this thesis, we used the block-based maze game Kodetu, developed by the group LearningLab of the Faculty of Engineering of Deusto University. In line with this objective, we set the following research questions:

   Research question 1. How do maze characteristics (width, height, total number of steps in the maze, optimal path, maze loops, turns, number of x-crosses and t-crosses) affect the performance in a maze-based programming challenge?

   Research question 2. How do coding limitations (blocks provided, block limit) affect the performance in a maze-based programming challenge?

2. Define an accurate measurement of difficulty in block-based maze games in order to provide efficient learning paths with difficulty that will be adapted to the learners' performance. We analyzed how the maze and coding characteristics characteristics affect the difficulty of the maze, and finally, we calculated the function that measures the difficulty of block-based maze programming challenges.

   Research question 3: How can the difficulty be measured in block-based maze games?

3. Design and develop the adaptive block-based maze game. Having calculated the difficulty function we were able to define the specifications of the adaptive system, design and develop it in order to achieve the creation of an efficient

adaptive Kodetu that will provide challenges of difficulty adjusted to learner's performance. To evaluate the effectiveness of the difficulty function and the adaptive Kodetu, we compared it to the corresponding non-adaptive Kodetu version. To achieve this, we set the following research questions:

Research question 4. How does the version (adaptive/non-adaptive) of the block-based maze game affect learners' achievements?

Research question 5. How does the version (adaptive/non-adaptive) of the block-based maze game affect the learners' performance in a maze-based programming challenge?

All the above determined the research hypothesis that we attempt to validate:

Learners can improve their performance in a set of block-based maze challenges for the development of computational thinking through a Computerized Adaptive Testing system that estimates the difficulty of each challenge according to maze and code characteristics.

These objectives, research questions, and hypotheses established the activities we have carried out in this research in order to validate them.

1.3. Research Methodology

To conduct the research presented in this thesis we followed the spiral process depicted in Figure 1.1 and briefly detailed below. The key idea behind this process is that the knowledge gained in the early stages allowed us to advance the investigation progressively and advance the comprehension and knowledge in the domains covered.

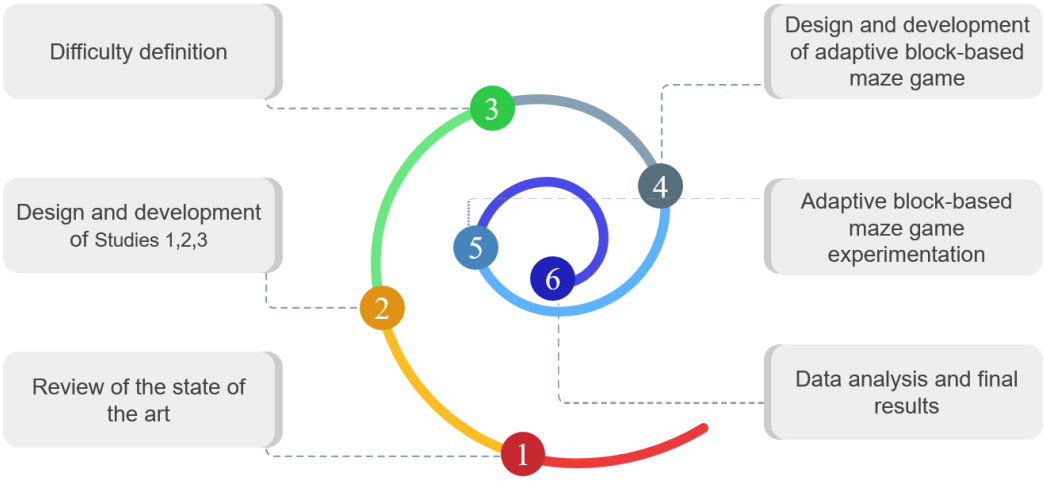

Figure 1.1 Research methodology followed to achieve the goals set during this thesis.

1. **Review of the state of the art**: During this step, we focused on examining the state-of-the-art in the main fields under consideration, CT, programming through educational games, adaptive educational games, and difficulty in educational games, in order to identify the current situation, tools, and technologies as well as the gaps in current literature. To accomplish this goal, we reviewed publications from the scientific community published in journals and conference proceedings.
2. **Design and development of Studies 1, 2, and 3**: In this phase, previously acquired or updated knowledge (new literature review) was used to design and develop the three studies. We defined the variables to investigate their influence based on our research questions, the design of the studies, as well as the target audience of our investigation. After each study the data gathered were analyzed from the different dimensions of interest, with the use of statistical tools. After the statistical analysis of data from Study 1, and based on the results and the limitations identified, we designed Study 2, investigating the influence of new variables, and improving the process of the study. The low performance identified in Study 2 led to performing Study 3, identical with Study 2 but with a different target audience (older participants).
3. **Difficulty definition**: The results from Studies 1, 2 and 3 constituted the basis for the difficulty definition of block-based maze programming challenges. During this phase we used the dataset obtained to test various algorithms and

find the right one to measure the performance of the learners and, therefore, the difficulty of the challenges.

4. **Design and development of adaptive block-based maze game**: The difficulty definition followed the design and development of the adaptive block-based maze. We defined the specifications of the adaptive game and developed it based on the needs of the investigation.
5. **Adaptive block-based maze game experimentation**: The goal of this phase was to acquire the necessary data to test the effectiveness of the difficulty function and the adaptive block-based maze game.
6. **Data analysis and final results**: The dataset obtained from the previous stage was used to perform the necessary statistical analysis, tests, and comparisons to obtain the final results that allowed us to validate the research hypothesis.

1.4. Structure of the Thesis

This thesis is structured in five chapters:

In the current chapter, we describe the motivation and context that gave rise to this research; we summarize the research questions and hypothesis of the thesis and its research methodology.

Chapter 2 presents the main studies analyzed during the literature review in relation to CT definition and the tools and methodological approaches to promote it. The influence of game-based learning, educational games, difficulty measurement, and adaptivity in the engagement and motivation of the learners, and the current approaches in the literature and the existing gaps are described.

Chapter 3 describes Kodetu, the maze-based game used in the research. The methodology followed to estimate the difficulty of a programming challenge in Kodetu is explained in addition to the results obtained in the three studies conducted. The methodology to define the difficulty function is described in detail, and the function is presented.

Chapter 4 comprises the methodology followed to design and develop the adaptive block-based maze game. The experimental procedures, the evaluation procedures, and the results achieved are included in the chapter.

Chapter 5 summarizes the main findings and conclusions of the research work, exposes the objective validation, and proposes future lines of work and challenges.

Chapter

2

Background and Related Work

This thesis analyses the development of CT through an educational game of adaptive difficulty based on the learners' performance. Therefore, we studied the wide range of CT definitions and theoretical backgrounds in the literature, the tools that enable effective acquisition of CT, and various methodological approaches to CT that have led to a better understanding of the CT development process.

The literature review has focused on the following areas: a) CT definition and theoretical backgrounds; b) learning to program through games; c) adaptive programming-related educational games; and d) the definition of difficulty in educational games, paying particular attention to block-based maze programming environments.

2.1. Computational Thinking

According to Wing (2006), Computational Thinking (CT) allows us to solve problems in a way that a computer can execute. . In just over a decade, since the author coined the concept (previously introduced by Papert, 1980), the movement started growing rapidly, affecting not only the scientific community and the educational world but also the tools and content available (Buitrago Flórez et al., 2017). Thanks to all this effort, significant changes are happening in the field of education, although many challenges

remain to be solved (Wing & Stanzione, 2016). The definition of the CT concept is still insufficient (Çoban & Korkmaz, 2021), and there is little consensus on its structuring in the educational curriculum (McCormick & Hall, 2021), where multiple initiatives from national and international organizations are facing difficulties due to the already complex structure of the educational policy, sometimes different depending on the region (Wilson et al., 2010).

The term CT is confusing in two ways. In the first place, a computer is not always necessary to develop CT. Many activities allow its development with letters, cards, puzzles, or simply pencil and paper activities, in what has been called "unplugged computing" (Computer Science-CS unplugged) (Duncan & Bell, 2015). Computational thinking involves ways of thinking and structuring problems or processes that allow proposing solutions stated in a systematic way, with a structured language. A computer is especially useful for later implementing that solution.

In the second place, considering the ubiquity of computers in our digital society, CT is an essential skill for many professional sectors, and is useful for any professional (Wing, 2006). Rushkoff (2010) anticipates that in the digital era, people must learn not only to use programs but also how to create them (program or be programmed). Gardner & Davis (2013) comment that, thanks to our apps, pursuing new possibilities make us active while only using them makes us dependent (app-enabling vs. app-dependent). In this sense, the English term “computer literacy” was introduced to refer to the incorporation of proactive ways of building technology beyond its simple use in the educational process (Rushkoff, 2010).

During the last decade, CT has been gaining importance within the scientific and educational communities, being proposed as a fundamental skill that must be included in the compulsory educational curriculum. Internationally, there is a growing tendency to incorporate it in primary education (Fagerlund et al., 2021; Geldreich et al., 2019; Sun et al., 2021; Tengler et al., 2021) and in secondary education (Djambong et al., 2018; M. Lee & Lee, 2021; Relkin et al., 2021; Torres-Torres et al., 2021). This movement occurs in parallel to the impetus of the STEM areas (Science, Technology, Engineering, & Mathematics) in education, and scientists give CT a relevant, if not central, role among the STEM disciplines (Henderson, 2009). Literature relates CT to educational areas already present in the curriculum, such as mathematics and science (Barbosa & Maltempi, 2019; Weintrop et al., 2016), social

studies and artistic language (V. Barr & Stephenson, 2011), chemistry and science (Gautam et al., 2020), geometry (Echeverria et al., 2019; Hutchins, 2018), music (Chong, 2019), or even ethics (Seoane Pardo, 2018).

Furthermore, in higher education, multiple studies relate CT to better learning and understanding of areas not directly related to computing, particularly within STEM sciences, such as probability and statistics (Abrahamson, 2006), biology (Wilensky & Reisman, 2006), physics (Perkins et al., 2006) or particle physics (Tinker & Xie, 2008), and in areas such as medicine (Saqr & Tedre, 2019) or telecommunications (Lui et al., 2020).

It is currently a topic of discussion how important CT is, how it should be taught in all levels of compulsory education (Denning, 2017; Tikva & Tambouris, 2021) and integrated into the curriculum (Lye & Koh, 2014). There is also an open discussion about the benefits of CT. On the one hand, it is claimed that CT improves general cognitive skills that can be transferred to other domains where the learners can improve their ability to solve more complicated problems (Csizmadia et al., 2015; Wing, 2006). On the other hand, Guzdial (2015) indicates that there is no significant evidence to support this statement; he does believe that many advantages of CT have been validated, with the main one being providing basic support to more effective learning of multidisciplinary areas.

Given that the lack of computer professionals in the present will continue to grow in the future, and due to the many benefits of CT, especially in regard to programming and computer science, many initiatives have been developed in the last decade at the local and international level to promote CT, such as the Hour of Code[1] , the Code Week[2] , or the Scratch Day[3] . These global events support changes in the educational world and provide easily accessible digital tools that allow learners to be introduced to basic CT skills. In this way, CT learners can use a wide variety of tools to develop these skills, often autonomously, but also supervised by educators. Most of these initiatives are based on digital platforms (mostly the web) where apprentice programmers can develop and improve their CT skills through games.

It is important to precisely define CT to study it and incorporate it into learning processes. To do this, we first differentiate it from two other similar concepts:

[1] https://hourofcode.com/
[2] https://codeweek.eu/
[3] https://day.scratch.mit.edu/

computer programming and algorithmic thinking. Computer programming is the process that professionals use to write code that instructs how a computer, application, or software program, performs. In general, computer programs are a set of instructions to facilitate specific actions. These actions should be encoded in a specific programming language (Robins et al., 2003). The traditional way of teaching programming has been reviewed in many studies. It has a recognized difficulty due to its cognitive complexity, the abstract nature of many of the concepts, and the educational methodologies used in the teaching process (Bashir & Hoque, 2016; Kuittinen & Sajaniemi, 2004; Robins et al., 2003; Rudder et al., 2007).

Algorithmic thinking is a concept that is quite close to CT, and not all the literature differentiates them. According to Futschek (2006), algorithmic thinking is a cognitive competence that allows a solution to be obtained through a series of steps, requiring a programmer to analyze given problems, specifying these problems precisely, dividing them into parts, finding the appropriate basic actions, and building a correct algorithm with those actions.

Although there is not a single agreed definition of CT, following the guidelines of the main introducer of the term Jeannette M. Wing (Wing 2006; 2011), CT was generally defined as "*the thought processes involved in formulating problems and their solutions so that the solutions are represented in a form that can be effectively carried out by an information-processing agent*." Aho (2012) simplified it by defining it as the set of mental processes related to the formulation of problems whose "*solutions can be represented with steps and computational algorithms.*"

Based on Wing's (2006) definition of CT, Reppening et al. (2016) differentiated three basic stages that describe CT as the Computational Thinking Process (the "Three A's"):

1. Problem Formulation (**Abstraction**): Problem formulation entails attempting to vocally comprehend a problem, such as by attempting to ask a question like "How does magma form?" or by using visual thinking (Arnheim, 1997), such as by creating a diagram describing items and relationships.
2. Solution Expression (**Automation**): In order for the computer to carry out the solution, it must be expressed in a non-ambiguous manner. This expression is made possible through computer programming.
3. Execution and Evaluation (**Analysis**). The computer will carry out the solution in ways that demonstrate the direct repercussions of one's own thinking. The

evaluation of solutions is aided by visualizations, such as the display of pressure values in the magma as colors.

CT includes a set of high-level skills and practices that are at the foundation of computing but are also applicable to many areas beyond computing. These skills include some recognized key concepts as proposed by Selby & Woollard (2013):

- Abstraction: Simplify a system by removing complexity and unnecessary details. It is a way of unraveling complex systems by picking out the relevant details , in order to hide the ones that are not. This can be done in a multitude of ways, for example, by changing the representation of the system. Many authors and curriculum definitions identify abstraction as the essential mental capacity of CT.
- Generalization: Solve new problems based on previously solved problems. Related to abstraction, it refers specifically to how the same algorithm can be generalized to solve a set of problems that include the original and other similar ones. It also includes the idea of recognizing and reusing common parts of one solution in another, and what is commonly recognized as patterns.
- Decomposition: Dividing a task (process, problem, system) into smaller parts; or being able to think of the whole in terms of its parts. The separate parts can be understood, solved, and evaluated separately, making large or complex problems easier to tackle and solve.
- Algorithmic thinking: As we indicated before, algorithmic thinking is a cognitive skill that allows a solution to be obtained through a series of steps.
- Evaluation: Ensuring that a solution is correct and that it fulfills its purpose (which often requires trade-offs between speed, resource use, or usability).

Of these five skills, algorithmic thinking embedded in programming is not only one of the fundamental activities that support CT skills but also serves as an objective demonstration of CT proficiency (Grover & Pea, 2013).

There are other skills and concepts that are usually close to CT (Council et al., 2010; Grover & Pea, 2013). In general, they are related in some way to the five mentioned above. These are the most significant ones: 1) data/information management

(collection, analysis, representation); 2) systematization; 3) logical thinking; 4) mathematical thinking; 5) procedural thinking; 6) system design; 7) problem-solving; 8) algorithms and procedures; 9) debugging; 10) parallelization; 11) automation; 12) modeling; 13) simulation; 14) recursion; and 15) efficiency and performance limitations (V. Barr & Stephenson, 2011). Fessakis et al. (2013) reviewed a series of significant school materials and identified the concepts according to their presence in the activities. In the order of appearance, the most common are: 1) algorithmic thinking (66%); 2) problem decomposition (50%); 3) abstraction (47%); 4) data representation (45%); 5) logical reasoning (43%); and 6) generalization (41%).

Kalelioglu et al. (2016) carried out a meta-analysis of 125 studies on CT up to 2014, where they identified the main concepts related to the definition of CT and ordered them by appearance. They are, in descending order: problem-solving, abstraction, computation, process, science, data, effectiveness, algorithm, concepts, capabilities, tools, and analytics. They also collected CT-related skills and ordered them as follows: abstraction, algorithmic thinking, problem-solving, pattern recognition, design-based thinking, conceptualization, decomposition, automation, analysis, testing and debugging, generalization, mathematical reasoning, implementation of solutions, and modeling. Other more recent meta-analyses fundamentally confirm this list of competencies (Palts, 2020) and reduce it to eight skill categories: problem formulation, abstraction, problem reformulation, decomposition, data collection and analysis, algorithmic thinking (including parallelization, iteration, and automation), generalization, and testing and evaluation.

There are other ways to analyze CT without using a list of competencies. Next, we comment on some of the most significant in the literature:

1. Relating it to science and mathematics in a taxonomy of four CT application practices: data, modeling and simulation, computational problem solving, and systems thinking (Weintrop et al., 2016).
2. The problem-solving skills that are learned with programming: processes and transformations, models and abstractions, patterns and algorithms, tools and resources, inference and logic, evaluation, and improvements (Gouws et al., 2013).
3. Differentiating three complementary dimensions, depending on how the learners' behavior is observed: concepts, practices, and perspectives (Brennan &

Resnick, 2012; Lye & Koh, 2014). Concepts, in this case, are those directly used by programmers, such as the variables and the statements. Practices refer to problem-solving practices that occur while designing the process, such as abstraction, debugging, or remixing. The perspectives have to do with the personal aspects of the learners, such as expression, connection, and reflection.
4. In a practical approach, Y.-J. Lee (2011) proposes three useful dimensions from which to face concrete CT experiences in young people: abstraction, automation, and analysis.

There are also operational definitions aimed at the educational community. One of the most cited references is found in the resource guide created by the American organizations ISTE (International Society for Technology in Education) and CSTA (Computer Science Teachers Association) (CSTA and ISTE, 2011). In this guide, CT is defined as a problem-solving process that includes (at least) the following features: formulating problems in a way that allows computers and other electronic tools to be used to help solve them; logical organization and data analysis; data representation with abstraction such as models and simulations; automation of solutions with algorithmic thinking (as an ordered series of steps); identification, analysis, and implementation of possible solutions in order to obtain the most efficient and effective combination of steps and resources; generalization and transfer of this process to a wide variety of problems. The British organization CAS (Computing at School) (Csizmadia et al., 2015) relates CT to logical reasoning and includes the skills of algorithmic thinking, decomposition, generalization, patterns, abstraction, representation, and evaluation. The ISTE (2016) updates the recognizable skills in students regarding CT: formulate problem definitions prepared for technological methods and algorithmic thinking; collect data or identify relevant datasets with digital tools to analyze them; break problems down into parts to extract key information and develop descriptive methods of understanding complex systems, and understand how automation works by using algorithmic thinking to craft a sequence of solution steps. In the same vein, Mannila et al. (2014) carried out a study of CT perception among secondary school teachers, identifying three sets of associated concepts: 1) data (collection, analysis, representation); 2) abstraction (including problem decomposition and algorithms), and 3) simulation, automation, and parallelism.

As we have seen in this section, there is a wide variety of approaches to defining CT and its use. In Table 2.1 we have summarized all the CT concepts and approaches presented in this section. For our research, regarding the fundamental skills (abstraction, generalization, decomposition, algorithmic thinking, and evaluation), we will consider that programming fosters the development of these skills, especially algorithmic thinking.

Computational Thinking Concepts and Approaches		
CT general definition	The thought processes involved in formulating problems and their solutions so that the solutions are represented in a form that can be effectively carried out by an information-processing agent.	Wing (2006;2011)
	Simplified to: the set of mental processes related to the formulation of problems whose "*solutions can be represented with steps and computational algorithms*".	Aho,2012
CT Process (the *"Three A's"*)	• Problem Formulation (Abstraction) • Solution Expression (Automation) • Execution and Evaluation (Analysis)	Reppening et al. (2016)
CT high-level skills Foundation of computing	• Abstraction • Generalization • Decomposition • Algorithmic thinking • Evaluation	Selby & Woollard (2013)
	Eight skills categories: • Problem formulation • Abstraction • Problem reformulation • Decomposition • Data collection and analysis • Algorithimic thinking • Generalisation • Testing and evaluation	Palts (2020)

Computational Thinking Concepts and Approaches		
Problem-solving skills learned with programmin	• Processes and transformations • Models and abstractions • Patterns and algorithms • Tools and resources • Inference and logic • Evaluation and improvements	Gouws et al. (2013)
Other CT skills	• Data/information management (collection, analysis, representation) • Systematization • Logical thinking • Mathematical thinking • Procedural thinking • System design • Problem-solving • Algorithms and procedures • Debugging • Parallelization • Automation • Modelling • Simulation • Recursion • Efficiency and performance limitations	Council et al. (2010) Grover & Pea (2013) V. Barr & Stephenson (2011) Kalelioglu et al. (2016)
Taxonomy of CT application processes related to science and mathematics	• Data • Modelling and simulation • Computational problem solving • Systems thinking	Weintrop et al. (2016)
CT complementary dimensions depending learners' behaviour	• Concepts directly used by programmers i.e. variables and statements. • Problem-solving practices that occur while designing the process such as abstraction, debugging, or remixing. • Personal perspectives of the learners i.e. expression, connection, and reflection.	Brennan & Resnick (2012) Lye & Koh (2014)

<table>
<tr><td colspan="3">Computational Thinking Concepts and Approaches</td></tr>
<tr><td rowspan="4">Operational definitions aimed at the educational community</td><td>Problem-solving process that includes (at least) the following:
Formulating problems in a way that allows computers and other electronic tools to be used to help solve them.
Logical organisation and data analysis.
Data representation with abstraction.
Automation of solutions with algorithmic.
Identification, analysis, and implementation of possible solutions.
Generalisation.</td><td>ISTE, CSTA (2011)</td></tr>
<tr><td>CT is related to logical reasoning and includes the skills ofalgorithmic thinking, decomposition, generalization, patterns, abstraction, representation, and evaluation.</td><td>CAS (2015)</td></tr>
<tr><td>Recognizable skills in students regarding CT:
Formulate problem definitions prepared for technological methods and algorithmic thinking.
Collect data or identify relevant datasets with digital tools to analyse them.
Break problems down into parts to extract key information and develop descriptive methods of understanding complex systems, and understand how automation works by using algorithmic thinking to craft a sequence of solution steps.</td><td>ISTE (2016)</td></tr>
<tr><td>Three sets of CT-associated concepts:
Data (collection, analysis, representation).
Abstraction (including problem decomposition and algorithms).
Simulation, automation and parallelism.</td><td>Mannila et al. (2014)</td></tr>
</table>

Table 2.1: Summary of the different CT concepts and approaches presented in this section.

2.2. Learning programming through educational games

Learning to program is well acknowledged to be a difficult task (Bashir & Hoque, 2016; Demirkiran & Tansu Hocanin, 2021). Novice programmers must learn concepts and abilities that are often unrelated to their previous experiences (Smith & Webb, 1999). As a result, the knowledge structures they must integrate with the information they are learning are frequently ineffective. Lectures and laboratory sessions are two ways of teaching programming, in which the learners reinforce what they have learned in lectures by constructing little programs of their own (Garner, 2003). However, although the lecture-based approach is widely used (Bennedsen & Caspersen, 2008; Mulholland, 1998), there is a scarcity of empirical data demonstrating its benefits (Price et al., 1993; Thomsen, 2008). Learners may find these classes monotonous, reducing their desire and enthusiasm for learning (Prensky, 2003b; Sarkar, 2006). It is difficult to keep learners involved in class if they are not interested. Innovative pedagogical methods to teaching and learning (T&L) techniques are used to improve student learning in order to achieve required IT-based skill sets such as CT.

2.2.1. Game-based Learning

As a successful educational strategy, behavioral scientists recommend using fun-based interventions to stimulate the learners (Dicheva et al., 2015; Oblinger, 2006). The use of game-based learning (GBL), in which people of all ages and genders can play games for several hours without understanding they are possibly in a T&L environment, is one such strategy for bringing interesting components to classrooms (Soflano, 2011). GBL is a significant area of learning that has become more relevant in recent decades. Wu et al. (2012) conducted a meta-analysis in which they observed that GBL is related to numerous learning foundations.

Garris et al. (2002) proposed a GBL model (Figure 2.1), in which the main feature of educational games is that instructional information is masked by the game characteristics. Students are absorbed in the game and forget about the "learning" aspect of the experience, even though they are frequently presented with new concepts to which they must adapt in order to succeed in the game. Based on

emotional and cognitive reactions elicited by contact with and feedback from gameplay, the player is supposed to induce desirable behaviors, and players should be encouraged to repeat the cycles within the game (Zapušek & Rugelj, 2013).

Figure 2.1 Garris et al. – Game Cycle

Many confuse the term gamification with GBL, and although they are closely related, it is important to differentiate the two terms. Although GBL involves designing learning activities so that game characteristics and game principles are inherent within the learning activities, gamification is the integration of game elements (such as point systems, leaderboards, and badges, among others) into conventional learning activities (Landers & Landers, 2014). According to Mohamad et al. (2017), GBL attempts to introduce information and skills through a game, whereas gamification uses game aspects to teach. Nevertheless, integrating GBL and gamification learning approaches in the curriculum results in increased learning engagement and creates an enjoyable learning experience for the learners (Hsieh et al., 2015; Hung et al., 2014; Khan et al., 2017).

Gaming activities are a good source of engagement and pleasure in learning because they provide players with immediate gratification when tasks are accomplished successfully, allowing them to progress to higher levels in the game (Mathrani et al., 2016). In particular, there are several important arguments for why games are effective learning environments that are deeply supported by current theory and research (Plass et al., 2015). The most popular and cited argument is that games increase motivation. Through a range of game elements that are common, gamification elements have been found to drive learners to stay engaged for longer periods of time (Khaleel et al., 2016). Incentives like stars, points, leaderboards, badges, trophies, and game dynamics and activities that learners enjoy or find fascinating, are among these elements (Alsawaier, 2018; Hidi & Renninger, 2006; Mekler et al., 2017; Rotgans & Schmidt, 2011). According to Przybylski et al. (2013), they serve as positive, informative performance feedback, and hence are a key aspect

of the motivational appeal of digital games because they provide players with the opportunity to satisfy their desire for competence. Furthermore, Francisco-Aparicio et al. (2013) mention points, levels, and leaderboards as ways to enhance the satisfaction resulting from the competence needed in games if offered in a non-controlling setting. In a similar study, Jung et al. (2010) discovered that in an idea generation task, providing feedback (in the form of points) and clear goals (in the form of levels and leaderboards) resulted in significant performance gains when compared to the control group, and theorized that these results were due to the game elements satisfying people's need for competence. Last but not least, Wang et al. (2015) discovered that users performed better when given demanding but reachable performance targets (i.e., levels) rather than moderate ones.

Related to motivation, one of the most frequently claimed reasons to use digital games for learning is that they provide a wide range of ways to engage learners. There are different types of engagements that affect the design of the game and reflect the individual learning aim, learner's characteristics, and setting. Domagk et al. (2010) presented the INTERACT model of learner activity, which differentiates between cognitive engagement (mental processing and metacognition), affective engagement (emotion processing and regulation), and behavioral engagement (gestures, embodied actions, and movement). The purpose of all of these sorts of engagement is to increase the learner's cognitive engagement with the learning mechanism. Games that do not create cognitive engagement are unlikely to aid the learner in achieving their learning objective. All types of play can lead to all three types of engagement (affective, cognitive, behavioral). Different game elements bring out different types of engagement in different contexts, and for different learners; hence the actual type of engagement will vary by game and within a game.

The learner's engagement is aided in part by the numerous options to make a game adaptive, adjustable, or personalized for the player (Andersen, 2012; Leutner, 1993; Plass et al., 1998; Turkay et al., 2013). Adaptivity refers to the game's ability to engage each student in a way that is tailored to their individual scenario. This could be related to the student's present level of knowledge, cognitive abilities, emotions, or a variety of other factors. The initial step in adaptive design is to identify the variable for which the game is designed to adjust, such as prior knowledge or self-control abilities. The following stage is to provide the learner with an appropriate response. This may entail altering the type and complexity of the tasks and advice

provided to the student (Azevedo et al., 2011; Koedinger, 2001), or it may entail a combination of those (Steinkuehler & Duncan, 2008).

Another benefit of game-based learning is that it allows for graceful failure: rather than being considered a negative conclusion, failure is viewed as a natural and often even required part of the learning process (Kapur, 2008; Kapur & Bielaczyc, 2012; Kapur & Kinzer, 2009; Plass et al., 2010). Failure has fewer consequences in games, encouraging players to take risks, try new things, and explore (Hoffman & Nadelson, 2010). They also allow for self-regulated learning while playing, with the player executing goal-setting, goal-monitoring, and goal-assessment procedures, as well as evaluating the effectiveness of the tactics employed to attain the intended goal (Barab et al., 2009; B. Kim et al., 2009). Many of the previously described concerns, such as motivation, engagement, and adaptivity, are linked to the ability to fail gracefully.

One significant variant of GBL is digital game-based learning (DGBL) (Prensky, 2003a), which is the one most directly related to most of the tools with which we are dealing. DGBL is an evolution from GBL, which promoted learning principles into digital game environments (Van Eck, 2006, 2015). Extensive research has been done on digital educational games showing that they help the learner to understand the concept of programming and develop a positive disposition toward programming through game-based activities (Demirkiran & Tansu Hocanin, 2021; H. Liu et al., 2022; Mathrani et al., 2016; Panskyi & ROWIŃSKA, 2021; Zhao et al., 2022). Consequently, based on the benefits of GBL and the difficulties entailed in programming, learning to program using digital educational games has become widely popular in the last years (Hou et al., 2022; Malik et al., 2020; Papadakis & Kalogiannakis, 2019).

Serious games can be defined as games whose primary goal is to teach or train and not to simply entertain (Abdellatif et al., 2018). For Zyda (2005), a serious digital game can be "*a mental contest, played with a computer in accordance with specific rules, that uses entertainment to further government or corporate training, education, health, public policy, and strategic communication objectives.*" Consequently, any video game designed to be more than pure amusement might be classified as a serious digital game. As a result, serious digital games encompass a wide range of digital games. (Muratet et al., 2009). Serious digital games may help computer programming students become more involved in the learning process by allowing them to "learn while having fun" (Coelho et al., 2011). In particular, the game

design affects the learning process. Merging and balancing game aspects (game characteristics, game mechanics, and gaming) with learning programming (knowledge, skills, and learning processes) while maintaining the serious game's potential promises is the major issue (Bergeron, 2005; Kirkley et al., 2007; Michael & Chen, 2005). Designing and delivering a serious digital game that engages and affects learners necessitates a thorough knowledge of the field and ideas involved: game design, learning theories, and domain content (De Freitas & Jarvis, 2006; Mestadi et al., 2018).

In general, numerous studies used serious digital games as learning environments to support the introductory programming education. Specifically, Malliarakis et al. (2017) developed CMX, a massively multiplayer online role-playing game to teach and enhance the learning of computer programming. The design framework of CMX includes important concepts in the development of educational games, such as the distinct characteristics of the users, the educational material organization and presentation, and the scenarios and activities supported by the game. Their research conducted with first-year undergraduate students showed that the game increased the students' motivation and enhanced their knowledge of computer programming (Kanellopoulou et al., 2021). Robocode (O'Kelly & Gibson, 2006), one of the first environments developed as an open-source educational game to support Java programming, Catacombs, Saving Serra (Barnes et al., 2007), and Elemental (Chaffin et al., 2009) are other examples of games that are specifically developed to teach programming (Kazimoglu et al., 2012).

However, learning to program differs crucially between university students and K-12 education students (Mitamura et al., 2012). At universities, programming teaching is aimed at students with varying academic backgrounds; thus, the flexibility entailed in the serious games makes them ideal for learning to program. As for primary and secondary school students, research has shown that visual programming and specifically block-based programming environments, due to their playful characteristics, among other aspects, play a significant role in introducing them to the basic concepts of programming and the computer science world and constitute them the main tool to achieve it (Min et al., 2020; Pelánek & Effenberger, 2020; Weintrop, 2019; Xinogalos et al., 2017).

2.2.2. Block-based programming games

Many novice-programming environments have been built using block-based graphical languages. These educational environments overcome the problem of the complex syntax of text-based programming languages based on their interaction with drag-and-drop and the natural language descriptions of the blocks (Weintrop & Wilensky, 2015).

There are several important block-based visual programming languages that have been developed. Among the first significant attempts were BridgeTalk (Bonar & Liffick, 1987) and Alice (M. J. Conway, 1998), followed by LogoBlocks (Begel & Klopfer, 2007). These types of tools base the user interface on visual blocks that are moved and placed like an assembly game, usually using the visual metaphor of puzzles, with their fitting pieces and shapes. These blocks function as an abstraction of the programming components: statements, data, control structures, and procedures, among others. Consequently, they considerably limit the prior knowledge that one must have to program and reinforce the structure of the program, eliminating the distraction of syntax and focusing on the logic that exists in the activity that one wants to propose. We see examples of these blocks from some of the more popular tools in Figure 2.2. Learners can compose functional programs using only the mouse to drag and combine pieces, receiving immediate visual feedback when constructs are valid or invalid (Bau et al., 2017).

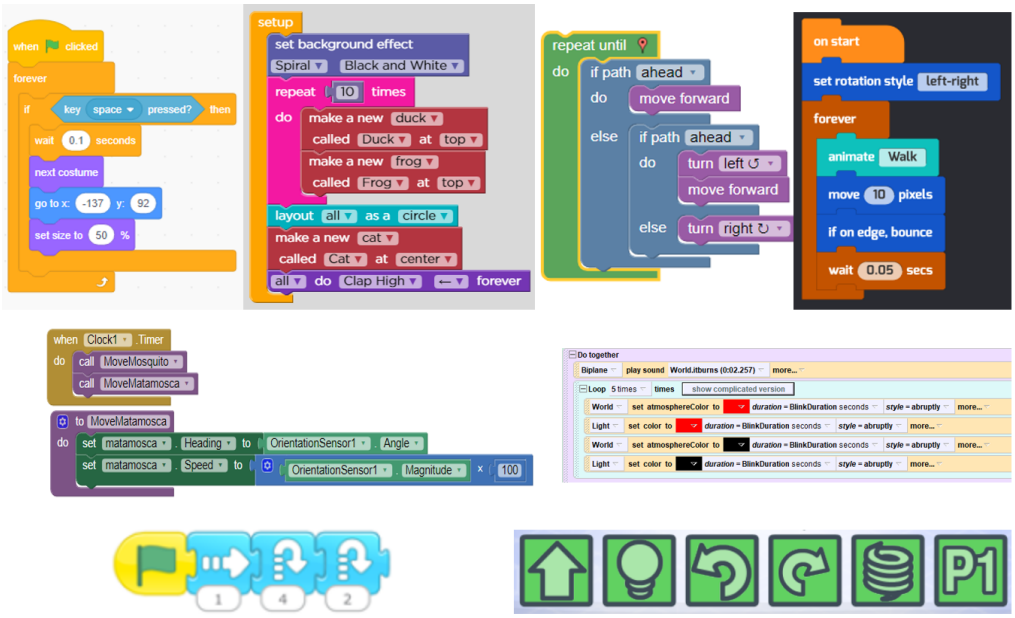

Figure 2.2 Examples of blocks from some tools. Top row: Scratch, code.org, Blockly, Tynker. Middle row: App Inventor, Alice. Bottom: ScratchJr, Lightbot.

Here are a few examples of popular block-based environments:

- Scratch[4] (Maloney et al., 2004) is a web-based programming language that makes it simple to create interactive tales, animations, games, music, and art. It is intended to assist young people (ages 8 and up) in the development of learning abilities in a variety of areas. Young students learn fundamental mathematical and computational concepts while also deepening their understanding of the creative process as they build Scratch projects. Scratch is perhaps the most well-known example of block-based programming (Resnick et al., 2009). Its concept has been taken by several block-based learning environments (e.g., Code.org[5], Tynker[6], Kodu[7], MakeCode[8], and HopScotch[9]), and has been popularized even further by Google's Blockly framework for creating visual programming environments.
- Blockly[10] is an open-source library for building Web applications using visual programming blocks. There are also offered the Blockly Games, a set of instructional games that teach how to program in a directed and progressive manner through several stages divided into seven categories: Puzzle, Maze (movement to exit a maze with repetitive and alternate structures), bird (continuous 2d motion with twists and x-y displacement based on conditions and Boolean expressions), turtle (Logo style drawing with repetition over angles and distances), movie (movement of geometric shapes based on a time graphic that ranges from 0 to 100), pond tutor (a shooting game with angles and forces that introduces text programming corresponding to the visual), and pond (a shooting game with players to beat controlled by the computer) (Eguíluz et al., 2018).
- Alice2 (Kelleher et al., 2002) is a programming environment for teaching programming while creating virtual environments in three dimensions. With this drag and drop programming system, users can play with the logic and programming structures taught in introductory programming classes without

[4] https://scratch.mit.edu/
[5] https://code.org/
[6] https://www.tynker.com/
[7] https://www.kodugamelab.com/
[8] https://makecode.microbit.org/
[9] https://www.gethopscotch.com/
[10] https://developers.google.com/blockly

making syntax errors. Users can experiment around conditionals, loops, variables, parameters, and methods (Muratet et al., 2009).

Usually, each block provides visual clues (color, shape, or size) that represent the combinatorial logic between pieces, and uses natural language tags (e.g., "while," "if," "forward") to represent its functionality. In addition, other visual clues are often used, such as color to differentiate conceptual use (e.g., differentiating by color the types of blocks as in Scratch) or encompassing visual shape as a metaphor for nesting/scoping.

Figure 2.3 shows an example of the Scratch editing screen, showing the most common spaces in this type of tool (labeled A, B, C, D). The tool provides a source space (A) in which all the available blocks appear, grouped in different ways depending on the tool. From that space, the learner can locate and drag each block to a different code (or work or construction) space (B) where the blocks can be grouped together to compose programs that can then be executed. In many tools, this execution produces visual feedback in the code space, in what would be the metaphorical equivalent to debugging in conventional programming. To complete this approach, there is a scenario (or execution) space (C) where the result of the execution is displayed, replacing the conventional programming console. In Scratch, as in many of the more elaborate tools, there are additional areas (D), such as the sprite space or the configuration space.

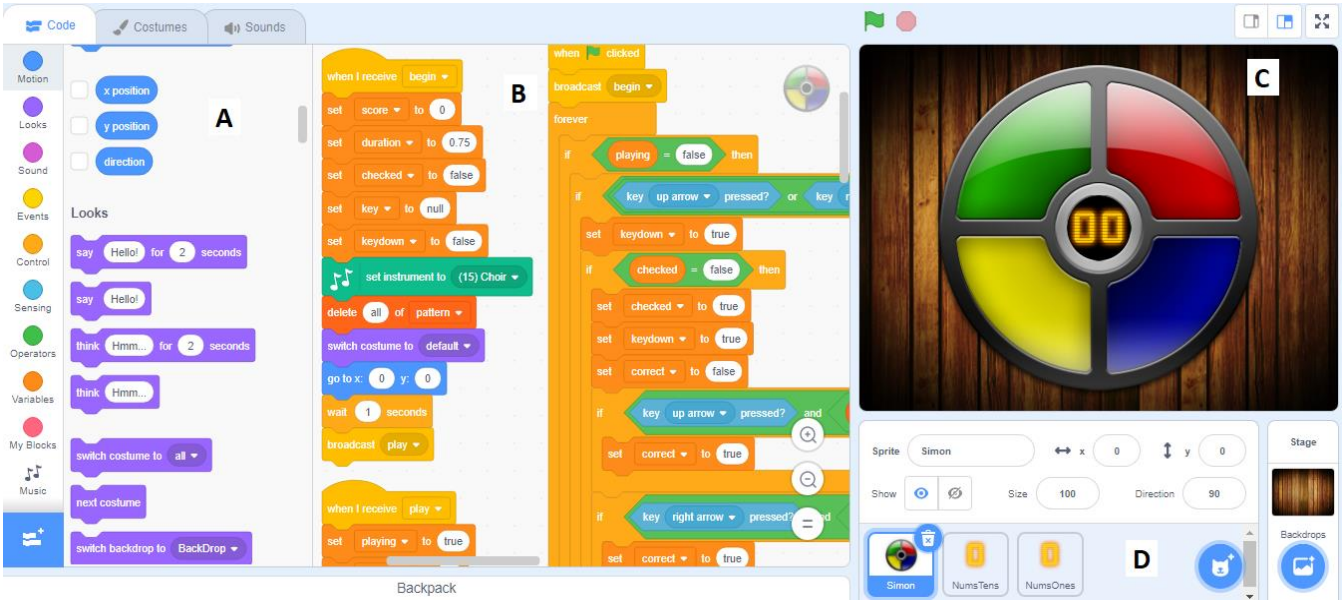

Figure 2.3 Scratch example screen. A: Source space. B: Code space. C: Scenario. D: Sprite and configuration space.

Although the distribution changes in each tool, the first three named spaces are very common. In a simpler application such as Kodable[11] (Figure 2.4) they can also be identified.

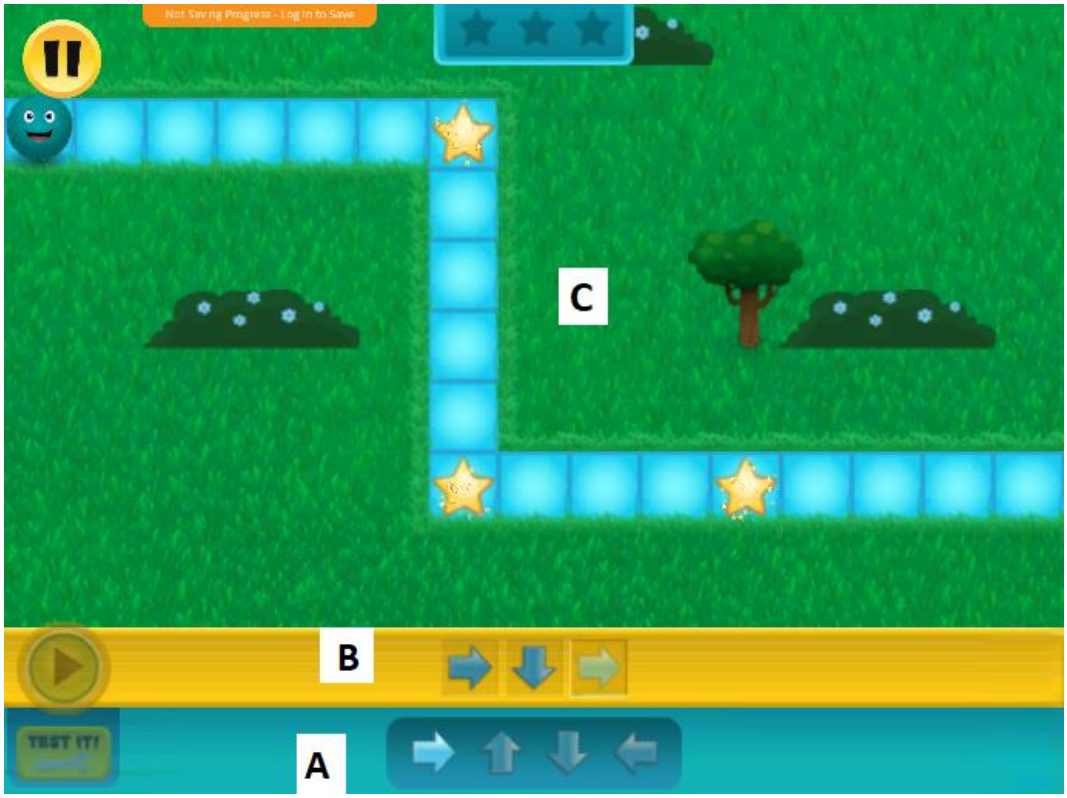

Figure 2.4 Kodable interface appearance. A: Source space. B: Code space. C: Scenario.

Several of the visual metaphors used are very relevant. The blocks (pieces) may have connecting gaps with geometric shapes that aid placement: not every block fits every other block, as is the case of Scratch (Figure 2.5). This is one of the fundamental elements of the conversion from a lexical syntax, very error-prone and distant from the human one, to a visual one that is easier to understand and practically avoids syntax errors (Bau et al., 2017).

[11] https://www.kodable.com/

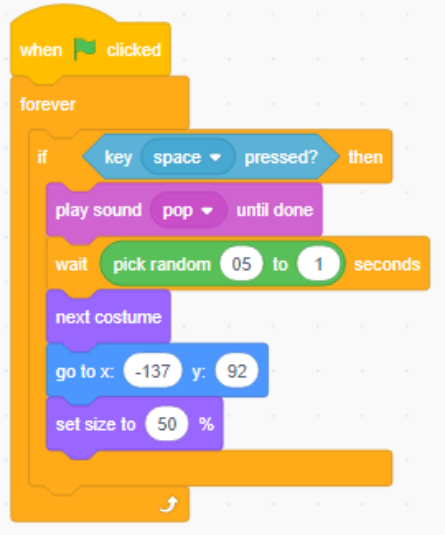

Figure 2.5 Code blocks in Scratch.

Another significant aspect is that sequentiality is represented by visual contiguity, either vertically (as in Scratch) or horizontally (as in Kodable 2.3. or Scratch Jr.). In this way, in code execution, blocks are executed in a natural way for learners: from top to bottom or from left to right. The blocks in the source area are in view so that they can be dragged with the mouse into the code space. In complex environments such as Scratch or Alice, the number of available blocks makes this composition impossible, and they are usually grouped in tabs, hierarchies, or palettes.

The importance of the nomenclature used in these tools is also noteworthy. Those aimed at younger ages try to avoid text as much as possible and use visual metaphors. Those that incorporate text reduce jargon as much as possible and use common and well-known concepts.

The abstractions used in these tools are also crucial to enable users to quickly understand how a program will behave when one of the constructs is used. For example, Alice's designers faced the challenge of developing intuitive abstractions to control 3D animations. To do so, they worked with users in the design process and replaced complex aspects such as transformation matrices with more intuitive concepts such as relative object motions (M. Conway et al., 2000).

In Figure 2.6 we summarized the basic concepts and approaches regarding programming through educational games. The investigation conducted in this thesis aimed at covering the gap regarding the development of adaptive block-based programming games that promote CT.

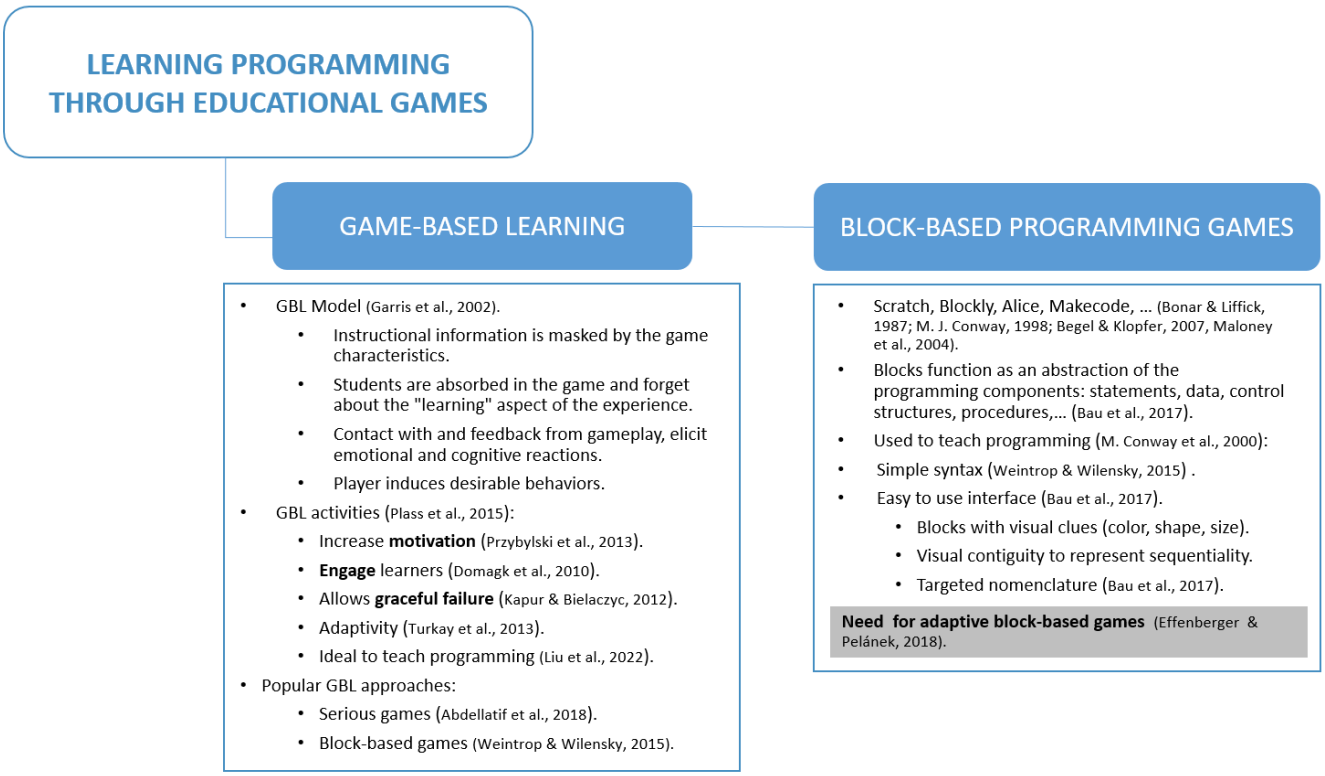

Figure 2.6 Basic concepts of section Learning programming through educational games.

For our research, regarding the fundamental skills (abstraction, generalization, decomposition, algorithmic thinking, and evaluation), we will consider that programming fosters the development of these skills, especially algorithmic thinking. It would be nice to have the ability to personalize the learning process and adjust to the learner's progress in these environments in order to achieve the most effective learning outcomes (Tlili et al., 2017). To accomplish this, we need to be able to assess the difficulty of block-based games so that teachers—or, ideally, a tool—can create increasingly harder learning activities with adaptive difficulty based on the learner's performance.

2.3. Adaptive educational games

As a result of the rising interest in and viability of GBL development, critical design concerns are surfacing. A primary concern is the creation of a game that is not only entertaining but also uses appropriate educational methods to achieve learning objectives. GBL settings are typically digital, self-paced games with control measures built through programming and then packaged, adding another layer of complexity to

the problem (A. Adcock & Van Eck, 2012). This modality necessitates that designers pay attention to automated pedagogical affordances that facilitate the growth of knowledge while accounting for as much player heterogeneity as possible. The creation of adaptive GBL settings is one technique to accomplish this (A. B. Adcock et al., 2011; Van Eck, 2007).

According to Shute and Zapata-Rivera (2012), the adaptive environments are defined as soft and hard technologies combined into a platform with the objective of enhancing student learning via adaptation. Adaptive GBL environments change the goal structure, the challenge difficulty, and the game narratives to accommodate a player's growing knowledge base. These environments work by modeling the learners' knowledge and following confirmed pedagogical standards consisting of growing set complexity and eliminating the help or scaffolding to facilitate schema production based on every learner's degree of understanding (Anderson et al., 1985; Graesser et al., 1999; Van Eck, 2007).

There are several well-known types of adaptive environments based on Shute & Zapata-Rivera (2012):

Adaptive Hypermedia Environments

Adaptive hypermedia environments or systems (AHSs) build on the foundation of an intelligent tutoring system by combining adaptive educational systems and hypermedia-based systems (Shute & Zapata-Rivera, 2012). Adaptive hypermedia (AH) is a development technique for hypermedia systems that offers an alternative to the usual "one-size-fits-all" approach. AHSs create a model of each unique user's goals, preferences, and expertise and utilize that model during the interaction with the user to adjust to the user's demands (Brusilovsky, 1996). In an adaptive educational hypermedia system, for example, a student will be provided a presentation tailored to their knowledge of the subject and a proposed set of most relevant links to progress further (Bra & Calvi, 1998; Brusilovsky et al., 1998). Brusilovsky (2001) distinguished between two different types of AHS: (1) adapting the presentation of content (i.e., different media formats) and (2) adapting the navigation or learning path (Kinshuk & Lin, 2004).

Adaptive Educational Hypermedia Environment

An adaptive educational hypermedia system (AEHS) is one sort of AHS. Because of its focus on a specific topic, the hyperspace of an AEHS is kept very small; as a result, the focus of the learner's model is solely on the learner's domain knowledge (Brusilovsky, 1996). According to Henze and Nejdl (2003), an AEHS comprises a document space, a learner's model, observations, and an adaptation component that proposes information and alters the look of links and icons. The document space is part of the hypermedia system and is filled with related information such as annotations and knowledge graphs. The learner's information, knowledge, and preferences are stored, described, and inferred by the learner's model. Observations are used to record information about the learner's interactions with the AEHS.

Collaborative Learning Environment

The idea of collaborative learning assumes that knowledge can be acquired within a population when people actively interact by exchanging experiences and taking on asymmetric roles (Mitnik et al., 2009). To put it another way, collaborative learning environments are those where learners collaborate on a common task in which each individual is dependent on and accountable to the others. These involve both face-to-face (Chiu, 2008) and digital interactions (such as online forums and chat rooms) (Chen & Chiu, 2008).

Simulation and Immersive Environment

Although simulations and immersive environments vary in response to specific user activities, the change is often caused by a specified set of rules rather than an underlying learner model (Rickel & Johnson, 1997). Immersive and interactive technologies, such as virtual reality, have changed the way we engage with our surroundings and even how we think about new ways to our relationship with reality. Virtual reality and other immersive information and communication technologies (ICT) have the potential to change the way we interact with the actual world (Rubio-Tamayo et al., 2017).

Computerized Adaptive Testing Environment

Computerized adaptive testing (CAT) is a test that adapts to the examinee's ability in real-time by selecting questions from a bank to provide a more accurate evaluation of their ability level on a common scale (Chang & Ying, 1996). CAT is a key advancement

in measuring applications that have benefited from advances in item response theory. Many researchers have explored the advantages of CAT over standard paper-and-pencil (P&P) examinations (Kingsbury & Weiss, 1983; McBride & Martin, 1983; Wainer et al., 2000). The main advantage of CAT systems derives from processes designed to deliver items that are difficulty-matched to the examinee's estimated trait level. CAT requires significantly fewer items to be administered and offers equivalent or higher measurement precision than full-length P&P versions of the same tests (Magis & Barrada, 2017; McKinley & Reckase, 1980; van der Linden & Glas, 2010; Weiss, 1982).

Flow theory, introduced by Csikszentmihalyi and Csikszentmihalyi (1992), has been the basis of contemporary adaptive game design principles. It explains that the goal of a game is to provide learners with challenges that balance the difficulty with their skills to prevent boredom (easy challenge-high skills) and anxiety (difficult challenge-low skills) (Kiili et al., 2012).
Specifically, the Flow Channel (Csikszentmihalyi, 2000) (Figure 2.7) depicts how the learner's difficulty and skills relate to one another, as follows:

1. Anxiety arises when the challenge level exceeds the abilities of the learners. This can be described psychologically as learners considering their abilities as insufficient and hence becoming demotivated. They usually believe the activity necessitates too much effort in comparison to their apparent ability. This frequently leads to early abandonment.
2. In contrast, boredom emerges if the learners' skills already contain the activity's learning outcome. Lost time is defined as having to commit time and money to obtain an output that already exists.
3. When abilities and difficulty are balanced, the learners enter a state of Flow, where interest fades, motivation fades, and boredom sets in.

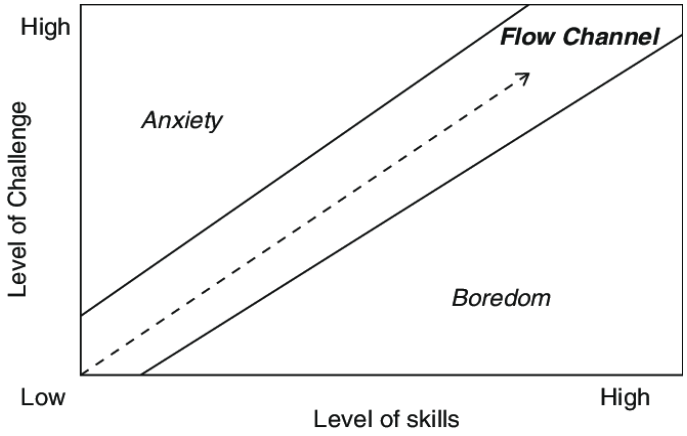

Figure 2.7 The Flow Channel (Csıkszentmihalyi, 2000)

Boredom and anxiety are undesirable emotions that drive players to seek out the flow state. If the player becomes bored, the game's difficulty must be increased. The player can choose a more demanding goal that is appropriate for their abilities. To get back into the flow state, they could, for example, play against an appropriate opponent whom they can barely beat. If the player is anxious, on the other hand, they must improve their skills in order to return to the flow state (Mayes & De Freitas, 2007).

Adaptive games are considered to be superior to non-adaptive games because they constantly assess the learners' performance and adjust the difficulty of activities to match the learners' particular level. Teachers in primary school face significant disparities in the learners' prior knowledge (Tomlinson et al., 2003). These variances can be resolved through personalized learning support. Tailored learning, on the other hand, is difficult to implement in a typical classroom due to the time and knowledge required by teachers to construct individualized programs (Dowker, 2017). In this context, adaptive digital educational games, which enable adaptive learning, respond to this difficulty. 'The purpose of adaptive educational systems is to make learning more effective and entertaining by adapting the system's behavior to a specific learner' wrote Papoušek and Pelánek (2015). Adaptive behavior is based on learner models, which measure the pupils' understanding (Vanbecelaere et al., 2020).

Adaptive educational games can help follow the learners' different learning paces. As a result, personalized learning can be achieved, with some learners demonstrating more learning efficiency (All et al., 2015). Van Oostendorp et al. (2014) randomly assigned students to either an adaptive or non-adaptive version of

an educational game. The adaptive version was adapted during playtime for either the challenges or the complexity of presentation. The players' proficiency was determined based on whether: a) they forgot to take procedure steps, b) they took steps in the wrong order, c) they took unnecessary steps, or d) actions were done within the allotted time. Although the results showed no difference in the engagement rating between the two versions, there was a big difference in the learning efficiency of the adaptive version. Students in the adaptive condition completed more tasks in the same amount of time as students in the control condition, making learning the educational material more efficient.

Similar results were observed in the study of Ali & Sah (2017), where students were randomly assigned to either an adaptive quiz-based e-learning game or a non-adaptive but similar e-learning environment to practice geometry, physics, and chemistry. The researchers conducted user assessments and compared their system to a baseline system to test the effectiveness of the proposed adaptive quiz-based e-learning system (non-adaptive Semantic Web-based version). The results showed that students in the adaptive system achieved greater scores in less time than students in the non-adaptive system.

Although the benefits of adaptive game-based learning are recognized and well-reported, and their learning potential and efficiency are known, more research is needed into developing adaptive educational computer games to promote students' CT. In particular, Hooshyar et al. (2019, 2021) introduced the game AutoThinking which promotes CT skills and conceptual knowledge by providing adaptive gameplay and learning processes. In AutoThinking, players must devise various tactics and solutions in order to finish three different levels of the game. The player has the role of the mouse, and their aim is to collect all cheese pieces and gain as many points as possible while avoiding two cats in the maze. Players can devise up to 20 strategies to clear the maze's 76 cheese pieces. Throughout the game, the player earns extra points for solutions that involve multiple CT ideas or skills. The game provides players with many alternatives for developing alternate solutions for the current state of the maze. During gameplay, the player's developed answer is graded based on its quality, and the game adapts accordingly. Furthermore, based on the current condition of the maze and the player's skill level, the game provides players with textual, graphical, or video feedback regarding CT concepts and skills that are incorporated in the game. The game also shows some of the game elements or buttons as pointers to help

players improve their solutions. The researchers separated 79 elementary students into two groups, the experimental group that AutoThinking was for teaching CT and the control group that used a conventional technology-enhanced learning approach. The findings show that AutoThinking increased students' CT skills and conceptual knowledge more than the traditional method. Regarding their prior knowledge, students improved their knowledge gain more when using the adaptive game than when using the traditional strategy. Finally, the results of the research indicated that the adaptive game improved students' learning attitudes toward CT more effectively.

Significant steps have been taken toward making block-based programming environments adaptive (Ludi 2015). Effenberger and Pelánek (2018) created a game presented as a spaceship navigating in space. The goal is to bring the spaceship to its destination while avoiding obstacles and fulfilling additional requirements, such as gathering gems, using a block-based environment. Left and right actions are immediately associated with a forward movement (flying). This enables the design of even complex spaceship trajectories in a compact manner. The game is not adaptive yet; nevertheless, they propose a method for adaptive task selection in order to construct a student model that estimates a student's knowledge and then uses the estimated knowledge to select an appropriate task. A student model of this type might assess knowledge in separate knowledge components relating to various programming ideas, with a Q-matrix mapping between tasks and knowledge components.

Computational Thinking Quest (CTQ), an innovative platform for learning computational thinking through online adaptive gamification, was introduced by Ng et al. (2021). CTQ contains a full interactive storyline that includes avatars, mini-games, and questions. It is developed using the Unity game engine and the Blockly block-based visual programming language. The questions have three levels of difficulty in order to obtain effective adaptiveness and the self-learning approach; moreover, there is feedback after answering each question. Other important features of CTQ are the hyperlinks to online learning resources for further reading, a ranking leaderboard to motivate active participation and encourage good performance. and a course management system with automatic data saving. In order to test the effectiveness of the games and the games' features, they performed a study with 107 engineering and ICT students. The statistical analysis performed revealed that the CT knowledge scores after taking CTQ were significantly higher than that before CTQ,

and the post-test results were much better than those of the pre-test. Lastly, it is encouraging that the vast majority of the participants found the game very engaging.

However, it is still difficult to find block-based programming environments that offer adaptive gameplay to fit individual learners' needs (Park et al., 2019). The level of adaptability of the few platforms that already exist consists of limited levels of difficulty; they offer activities that are either easy, medium, or difficult, a classification usually made based on the personal opinion of the activities' creators and not on objective research criteria. In addition, in some cases, the adaptiveness is limited to the feedback given, or the narratives, and the adaptive game worlds and scenarios (or quests) provided are still lacking in broad, consolidated and integrated research focus (Lopes & Bidarra, 2011; Shaker et al., 2016). Specifically, regarding programming learning, due to the difficulties entailed and the necessity of programmers nowadays, there is a great need for effective systems of adaptive difficulty that are specific to the particular demands of programming learning. In Figure 2.8 we summarized the basic concepts and approaches of this section.

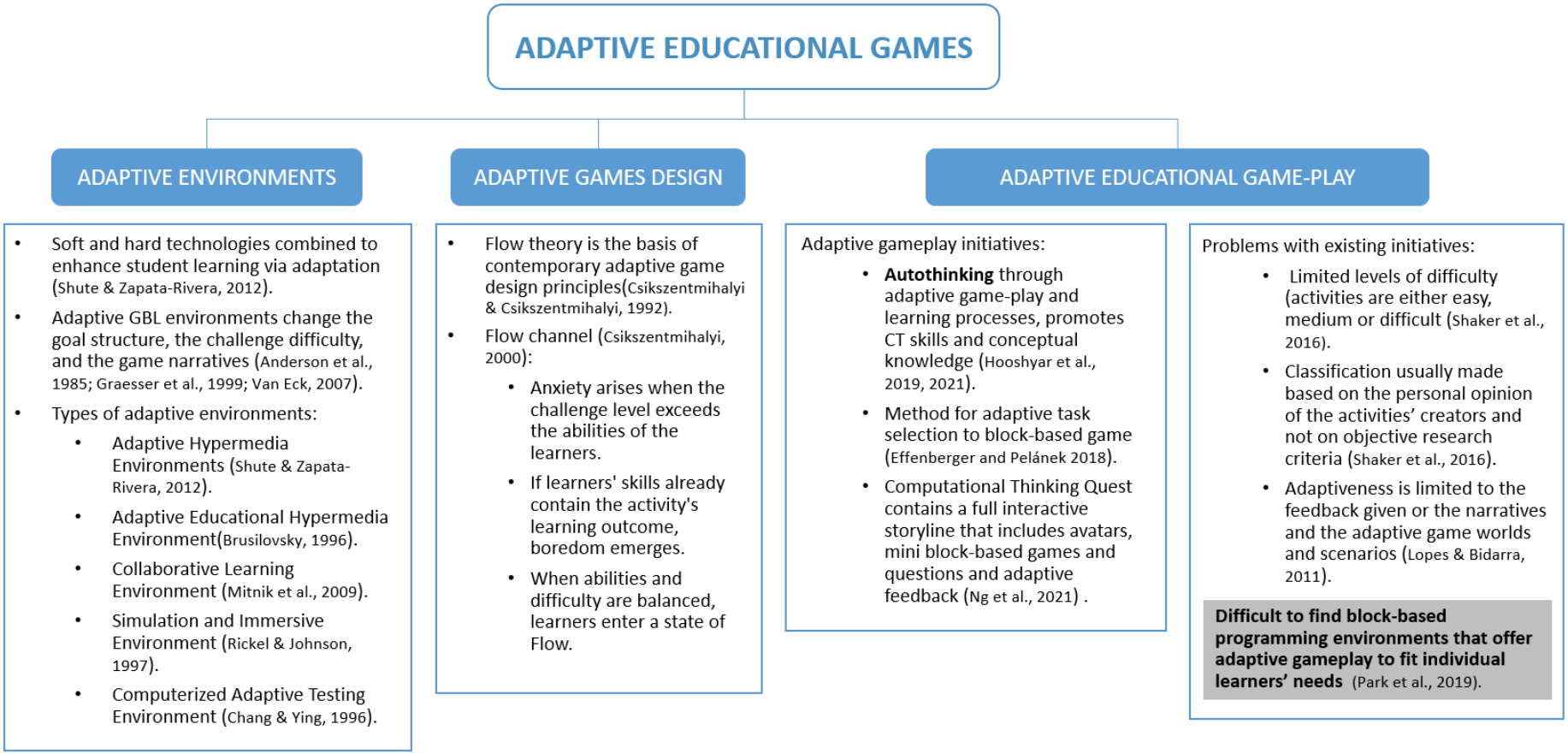

Figure 2.8 Basic concepts of section Adaptive educational games.

2.4. Difficulty in educational games

Difficulty is generally defined as the commitment taken to effectively perform an operation (Gallego-Durán et al., 2018). Li and Belkin (2008) defined task complexity as a subjective perception judged by task doers in their complete task classification scheme. It was operationalized by Cole et al. (2010) as expected task difficulty. Similarly, Kim (2006) defined task difficulty as the perception of a task's complexity by task performers. Many researchers used questionnaires to assess task difficulty based on users' self-reported perceptions of how difficult an activity is. The individuals' judgments of their own competence and the difficulty of the tasks they must do are major predictors of achievement and behavior. However, the implication of any given outcome for one's abilities is hazy at the objective level. Failure, for example, may be defined as "due to the difficulty of the work." This, however, is not easy to differentiate from the notion of "it is too difficult for me" or "I am not smart enough for it." Norms are required to judge ability or task difficulty apart from the subjective perception of difficulty in performing a task (Nicholls & Miller, 1983). Consequently, changes in the learners' ideas of ability, difficulty, and related notions should result in developmental changes in accomplishment and behavior. Evidence shows toddlers under the age of six, who most likely do not grasp the concept of difficulty, exhibit calculated goal setting when concrete difficulty cues are provided. That showed the presence of an objective conception of difficulty (Heckhausen, 2013). Given repeated opportunities to execute diverse tasks, children choose objective difficulty levels with moderate odds of success. This shows that these children comprehend, at some level, variations in objective task difficulty and recognize that more difficult activities necessitate greater skills (Nicholls & Miller, 1983).

Liu et al. (2011) created a theoretical model of user perception of task difficulty and the affecting factors, e.g., task performance, search behaviors, knowledge background, and relevance judgment, among others, based on Vakkari´s (1999) research. The model consists of five sets of components: 1) task completion, 2) task difficulty, 3) user background, 4) user contact with the system and all process behaviors, and 5) task features (Figure 2.9). Regardless of whether the activity will be successfully

solved, the task doer will eventually finish working with it. This is the model's first component, and it serves as the task's endpoint.

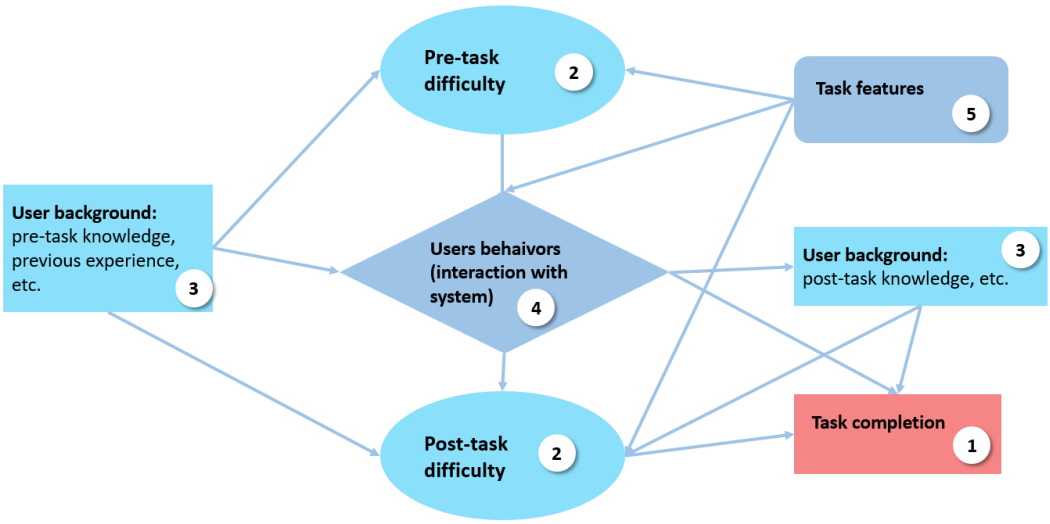

Figure 2.9 A theoretical model of user perception of task difficulty and affecting factors (J. Liu et al., 2011)

The task doer's impression of task difficulty is the subject of the second set of components. Although it is not always explicitly stated, the person performing the task usually has an estimate of the task's complexity, which is the expected task difficulty, before beginning work on it. They have a different perspective of the task's complexity after working on it, which is reflected. The significance of the two forms of task difficulty is that the task doer's impression of task difficulty is likely to alter after they complete the task, which will certainly affect their satisfaction with the system.

The background of the person performing the task, which includes both pre-task and post-task information, is the third group of components. This is related to the "previous knowledge" in Vakkari's (1999) model, which highlights the relationships between task complexity, problem structure, knowledge, and information activities; however, the model presented takes into account more variables than "prior knowledge." The expected task complexity is influenced by the person's pre-task background, such as pre-task topic knowledge, previous experience with the sort of task, and so on. Together with pre-task background elements, the post-task perception of task difficulty will most likely be influenced by the person's post-task background and post-task topic knowledge.

The information system user's behaviors, demonstrated by engaging with the system, are the fourth group of components. This is comparable to Vakkari's (1999) model's "information acts." Users' interactions with the system could be indicators of task difficulty.

The task features are the fifth set of components. It is similar to Vakkari's (1999) model's "problem structure." Users' behaviors and perceptions of difficulty have been found to be influenced by task elements, including different sorts of tasks.

The theoretical model presented above makes clear the importance of systems with adaptive difficulty that will offer personalized learning paths. Determining and quantifying task difficulty is critical for comprehending task performance as well as designing and improving assignments, activities, and games (Tavakoli, 2009).

As was presented and explained in the previous section, adaptive games are considered to be more effective than non-adaptive games as they continuously evaluate the success of the learner and adapt the difficulty of the activities to the individual level (Vanbecelaere et al., 2020).

Samprayo-Vargas et al. (2013) assessed the effectiveness of adaptive difficulty adjustment with 234 secondary school students by creating a Spanish cognates bubble game using early findings from pilot research, learning and motivation theories, and common features of current simple instructional computer games with adaptive difficulty. They created two versions of the game, a simple version and an adaptive version. The method for adaptively adjusting difficulty was based on the Computer Adaptive Test of 'On-Demand Testing' of the Victorian Curriculum and Assessment Authority (VCAA). Based on this information, an adjustable difficulty algorithm was developed to maximize learning by allowing students to proceed to more difficult educational content once they had mastered the topic and to reduce the complexity of the game if they were facing problems. Students were separated into three groups, and each group was given a different activity/game. Two of the groups were identical except for the difficulty adjustment mechanism. The results showed that the adaptive difficulty adjustment game outperformed the simple version of the game and the written exercise in terms of learning outcomes. Students playing the adaptive difficulty adjustment game version were not affected by the time reductions in the gameplay complexity; according to game log data evaluating the

correctness of the students' responses over time, it took them shorter to reach the highest levels of difficulty.

Similarly, Lomas et al. (2017) sought to clarify whether difficulty indeed affects the learners' motivation. To achieve that, they performed three different experiments. In the first experiment, the players were randomly assigned to a particular level of difficulty in the game Battleship Numberline. The game consists of a simple online game where players attempt to explode targets by estimating numbers on a number line. The researchers created five different game levels of difficulty and used a regression model to handle factors predicted to vary in difficulty from very easy to very hard. The second experiment was performed with a chess game, where participants freely choose their opponent, having knowledge of their chess rank. Comparing data from the two experiments, they found that when players could choose their opponent, knowing the level of difficulty that they would face, they had decreased motivation to continue as the levels were becoming more difficult. On the contrary, the participants in the Battleship Numberline adaptive game showed high levels of engagement and motivation. In the final experiment, the researchers investigated the role of different sceneries and suspense in the learners' motivation. The analysis of the data obtained showed that a game level with a small number of easy items was less motivating than a level with a large number of challenging objects. The component of repetition, and not challenging items, plays a key role in a player's motivation.

As described earlier, difficulty is considered a key factor in promoting the motivation of learners in educational games and resulting in better learning outcomes. Several approaches to defining difficulty in games already exist, ranging from measuring the difficulty of video games to assessing the difficulty of educational games. However, current definitions remain mainly intuitive; that is, the difficulty is defined as the ability and the effort necessary to complete an educational task. In particular, Gallego-Durán et al. (2018) proposed a definition of difficulty that depends on the learners´ progress on activities over time and how to measure it. They defined difficulty in a [0,1] range and a score function with a broader range but also an upper limit. Then, they defined an easiness function that depended on the progress/score that the player obtained in a specific timeframe. Finally, difficulty was defined as the inverse of the easiness function, equaling 1 minus the easiness function. Using this definition, they

developed PLMan, an adaptive learning platform that consists of a web application and a game to teach computational logic.

Aponte et al. (2011) claim that the difficulty of a challenge is the probability that the player will fail at it. In the first experiment, they extract objective difficulty metrics from a simple game utilizing a synthetic player and a basic artificial intelligence (AI)-driven player. The concept of difficulty is based on the collection of challenges the player has attempted to accomplish, and it defines a challenge's difficulty as the percentage of losing over time. Later, they explore more in-depth how a player learns while playing a game. More specifically, they assess the player's ability to execute some fundamental behaviors that they learn and practice while playing and that have been identified by the game creator as critical to overcoming a given obstacle. These basic behaviors, which are expected to be observable during a game session, are referred to as skills. Then the researchers calculate the relationship between a player's level of mastery of a particular ability and the likelihood of failing at a given task.

Another effort to investigate difficulty, this time on a video game, was performed by Constant and Levieux (2019). They designed and implemented three games, each representing a different type of difficulty. They defined three categories of game difficulty: sensory, intellectual, and motor. Sensory difficulty refers to the amount of effort required to obtain information about the game's current condition. The amount of effort required to infer or deduce the solution to a problem in terms of action(s) to take from the given knowledge is referred to as logical difficulty. Finally, motor difficulty refers to the physical agility required to do these tasks. Malone's definition of a challenge as a source of uncertainty in video games (Malone, 1982) was the foundation for this research definition of difficulty, as well as Costikyan's reference to uncertainty of outcomes as the uncertainty of success or failure (Costikyan, 2013). To account for individual differences, the authors used mixed-effects logistic regression to estimate the objective difficulty for each challenge. Each challenge's duration and difficulty characteristics (e.g., cursor speed, number of cells) were used as fixed effect parameters, with random intercepts added. They used the random intercepts to predict a coefficient for each player, which they used to assess their overall performance. The difficulty estimation error is the difference between the players' objective difficulty and their estimates of their chances of succeeding. According to the findings, predictive accuracy ranged from 61% for the motor tasks to nearly 70% for the other activities. The researchers also saw a learning impact,

displayed as a negative effect of time on the difficulty for a given difficulty parameter value. This learning effect depended on the nature of the tasks, with logical work having a stronger learning effect.

Szabo et al. (2016) presented a model of an algorithm for game task difficulty estimation based on Bayesian probability theory and existing research on human intelligence. They created a simple model of the scenario to measure the complexity of tasks in educational games. The model is divided into two components, the players and the tasks. The players are represented as simple agents with a certain level of intelligence. The intelligence quotient (IQ) as evaluated by standardized testing corresponds to this intellectual level. A normalized value between 0 and 1 represents the intelligence level. The tasks are given a difficulty value, which is also represented by a normalized value between 0 and 1, with 0 being the easiest and 1 being the most difficult. According to their theory, a random variable D that follows a beta distribution, and is understood as a Bayesian degree-of-belief probability, can be used to model the complexity of a single job. A simulator ran the model to test its effectiveness, and it was also presented in the paper as an example of a real-world application of the method, an educational game made for university students of architecture and civil engineering. However, no further results were presented to deduce the success of the application of the method.

2.4.1. Block-based maze game difficulty

Although there are studies that intent to define the complexity, cognitive load and difficulty of puzzles, microworlds or mazes, as far as block-based maze games are concerned, there is a lack of a definition of difficulty that takes into account the specific characteristics of these programming games and provides maze-based challenges adapted to the learners' performance.

Pelánek and Effenberger (2020) studied the difficulty and complexity of puzzles and microworld elements in order to provide guidelines for the design of block-based programming games. They analyzed basic elements of microworlds and puzzles that were implemented in the open-source project RoboMission. The researchers obtained and analyzed a dataset from 5,800 students. They tracked events performed by the students, such as code modifications, executions, starting and ending of the attempt (student opened or closed the puzzle), code reset, opening a toolbox category, request for a hint, and feedback from the system after an attempt.

They calculated basic difficulty measures, such as failure rate, the median time to solve the puzzle, the median number of actions like edits, executions, or submits, as well as complexity measures based on the solution of the puzzle and the microworld features. They also performed a correlation between these measures. They found that high difficulty metrics were associated with the puzzles that first introduce the if-else concept and those that simply require a sequence of commands, but finding the correct sequence is challenging.

Kelleher and Hnin (2019) performed a study with 75 participants in order to create a model to predict the cognitive load of programming tasks using a block-based programming environment for creating 3D animations and games. They created five sets of code puzzles that covered three programming constructs: executing statements in parallel, count loop, and executing statements in sequence. Along with the dataset obtained and analyzed, the researchers, during the intervention, recorded observations about the problem-solving approach and behaviors of the participants. After linking the observations with the logs, they looked into what elements contribute to a larger cognitive load when solving block-based programming tasks. The results showed that the learners' ability to use feedback effectively, had a large impact on overall performance. In addition, using procedures that the students are unfamiliar with increases cognitive load, whereas locking some blocks in the solution (such that they cannot be modified) reduces cognitive load.

Some common complexity measures of programming tasks in general are based on the resulting program considering the length of the learners' solution (Alvarez & Scott, 2010), the number of programming concepts involved (Ihantola & Petersen, 2019), the number of flow-of-control structures (Alvarez & Scott, 2010), and the number of operators and operands. These approaches consider programming tasks with text-based programming languages.

Regarding the complexity of mazes, Bagnall and Zatuchna (2005), in their effort to classify maze tasks, divided the characteristics of the maze that affect the complexity of a maze task into two categories: 1) the characteristics that are independent of the learning algorithm, such as the number of squares and density of obstacles in the maze task, and 2) those that depend on the learning algorithm–the agent's ability to detect and solve them.

Other approaches, such as the one developed by McClendon (2001), focus on the mathematical measurement of the difficulty of a maze. McClendon used various complexity measures of the hallways in a maze and calculated the overall complexity

and difficulty of the graph of the maze. However, maze-based educational games do not use complex mazes; thus, this function is not applicable to the mazes of this type of game. Consequently, using only the hallway measures of a maze is insufficient to define the difficulty of an educational maze-based game. (For example, maze loops, type of blocks used, and other similar aspects should also be considered.)
Figure 2.10 presents the basic concepts and approaches of this section.

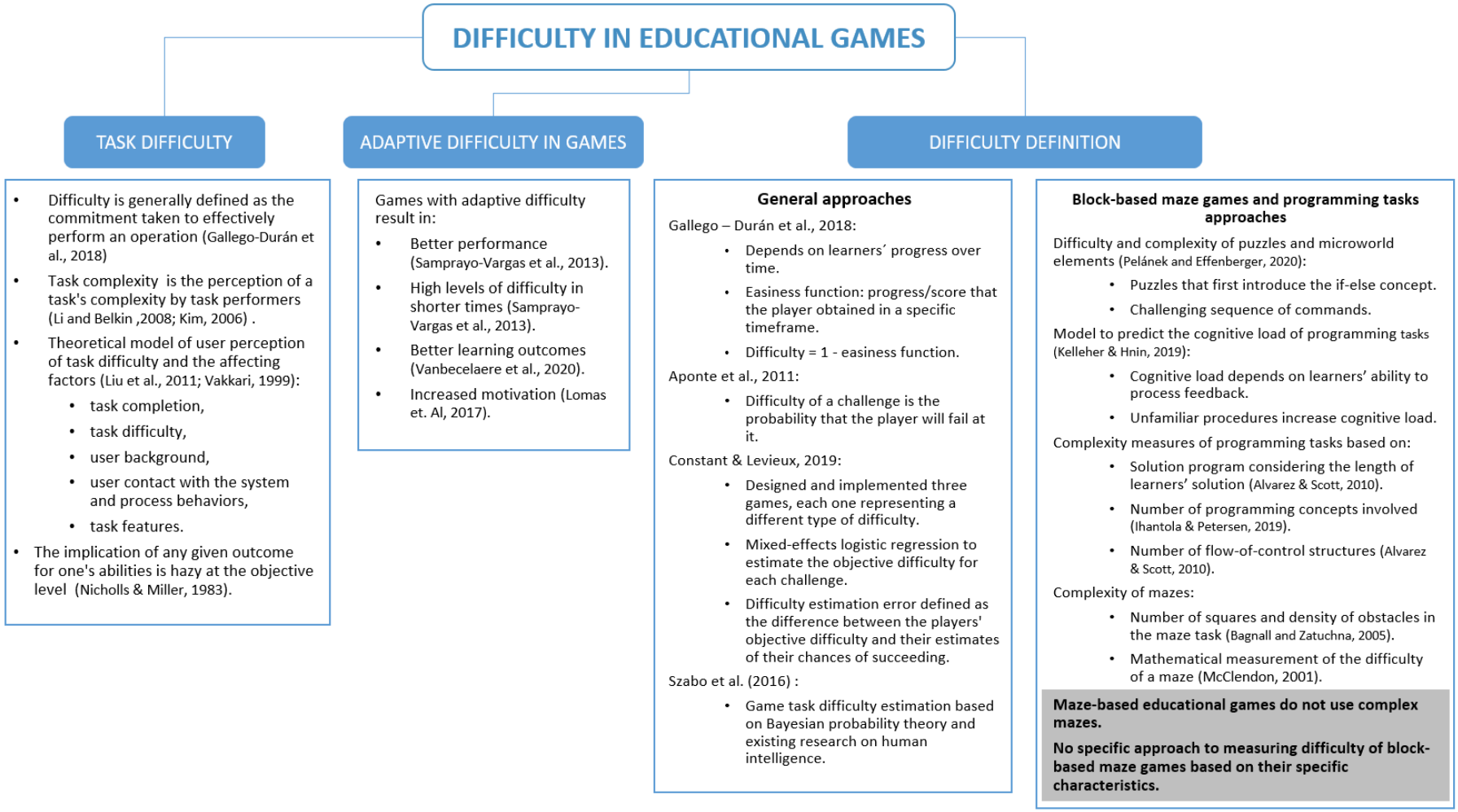

Figure 2.10 Basic concepts of section Difficulty in educational games.

During our research, we focused on developing an adaptive system that will provide learners with learning paths of adaptive difficulty for block-based maze games. To accomplish this, we conducted a review of the literature and state of the art of all related areas. Figure 2.11 describes the path followed to perform the current literature review based on the important concepts of the thesis and the gaps identified. According to the literature review performed, the importance of CT for computer professionals and not only for learners, is well-established by many researchers. The complexity of CT concepts and the many benefits of promoting CT skills and, more specifically, algorithmic thinking (directly related to programming) led to the implementation of GBL by educational stakeholders and the creation of block-based programming games. Although the effectiveness of games of adaptive difficulty is recognized, when it comes to block-based maze games, there is still no specific approach to measuring their difficulty. In our research, we aimed to overcome these limitations and provide a measurement of the difficulty of block-based maze games, considering not only the learner's performance in the game but also the characteristics of the activity. To achieve this, we performed several experiments and analyzed the participants' interaction logs recorded automatically by the tool using learning analytics techniques. In addition, during this literature review, we identified that the adaptive block-based programming games that currently exist are providing limited levels of difficulty (usually only three levels: easy, medium, difficult), and the classification of these programming challenges is made in an intuitive way and sometimes the adaptability is limited to the narratives and feedback given to the learners. To address these gaps, we used the difficulty function measured to create the adaptive block-based maze game and performed a study to compare the adaptive and non-adaptive version of the block-based maze game.

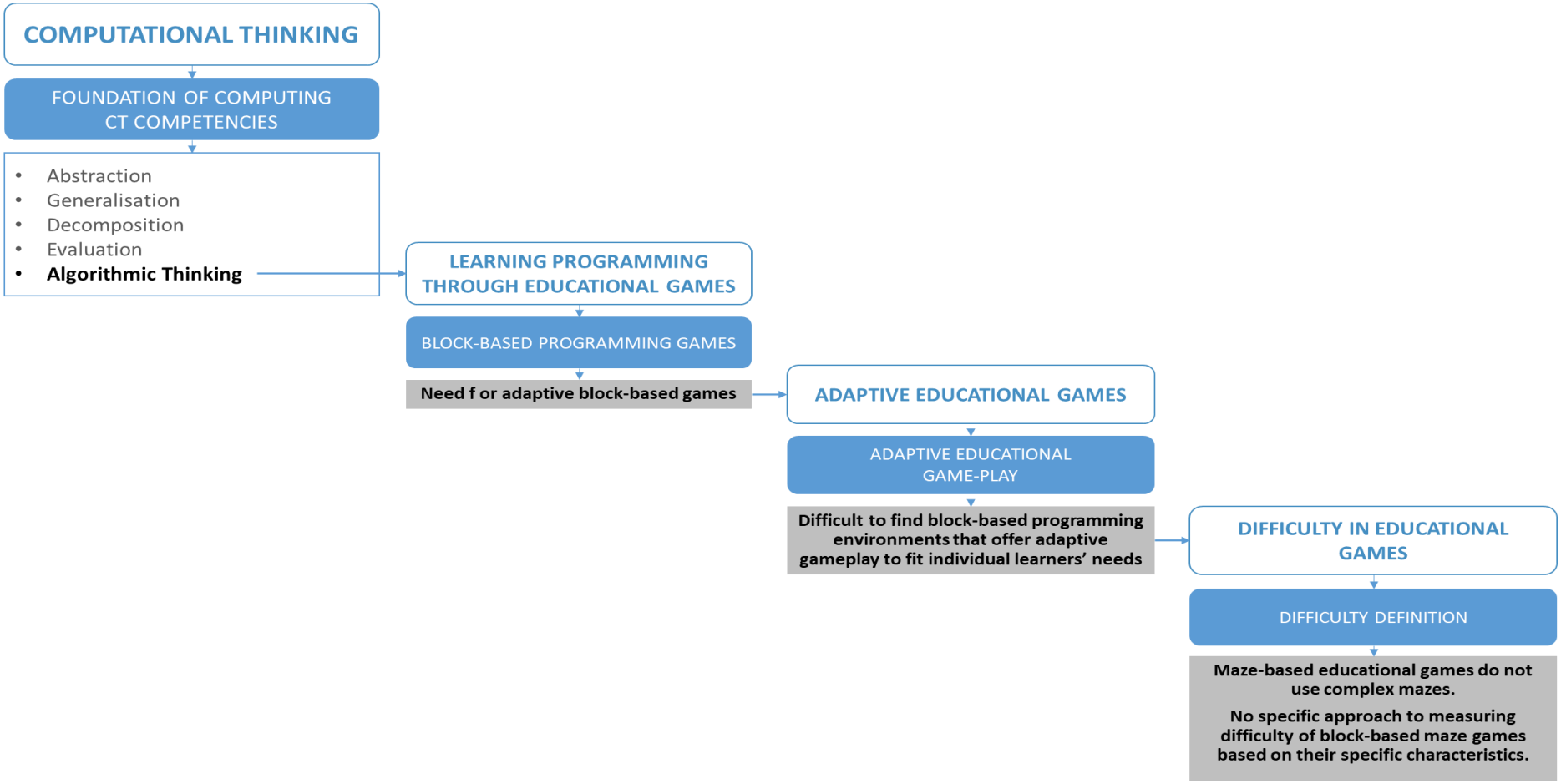

Figure 2.11 Literature review path based on the important concepts and the gaps identified.

Chapter

3

Difficulty of Block-based Maze Games[12]

With the aim to provide an adaptive educational platform that will help develop CT, we tried to design a maze-based online system using block-based programming challenges that matches learners' competence and challenges' needs. However, the lack of a clear definition of the difficulty of this type of challenges led us to first study how we can measure the difficulty in this context. In this chapter, we present the investigation performed to define the difficulty of block-based maze games.

Our study aims to answer the following research questions:

1. How do maze characteristics (width, height, total number of steps in the maze, optimal path, maze loops, turns, and numbers of x-crosses and t-crosses) affect the performance in a maze-based programming challenge?
2. How do coding limitations (blocks provided and block limit) affect the performance in a maze-based programming challenge?
3. How can the difficulty be measured in block-based maze games?

[12] This chapter is an amended version of the published journal article: I. Kanellopoulou, P. Garaizar and M. Guenaga, "First Steps Towards Automatically Defining the Difficulty of Maze-Based Programming Challenges," in IEEE Access, vol. 9, pp. 64211-64223, 2021, doi: 10.1109/ACCESS.2021.3075027.

For that purpose, we conducted 3 studies using Kodetu, a block-based maze game developed by the group LearningLab of the Engineering Faculty of the University of Deusto.

3.1. Kodetu

Kodetu[13] is an online platform based on the Blockly game ''Maze'' where participants must solve challenges by exiting the maze from the initial point, using a block-based programming interface. The aim of Kodetu is to develop basic programming skills by creating visual programs for solving mazes. It is an educational game that allows one to easily create new individual and sequential challenges.

A Kodetu challenge is a maze level where an astronaut is located at an initial position and the exit of the maze is marked at a different point of the same maze. The challenge is solved successfully when the participant leads the astronaut to the exit of the maze using the visual blocks provided. We use the term sequence to define a group of consecutive Kodetu challenges.

The interface of Kodetu consists of three parts: the maze, the blocks provided to solve the challenge, and the workspace (Figure 3.1). The first panel displays the maze that the participants must solve. The participants must lead the astronaut from the beginning of the maze to the endpoint. To achieve this, they use the blocks (programming instructions) provided in the middle part of the interface. There are movement blocks (go forward, turn left, and turn right), loop blocks, and blocks to define one- or two-branch conditionals that check for whether there is a path on the left, on the right, or forward. The user drags and drops these blocks to the third part of the interface, the workspace, where they build the visual program that leads the astronaut to the endpoint. When the user clicks the ''play'' button, the program defined in the workspace is executed, and the astronaut moves according to the programmed instructions. When a new challenge is created, additional features can be added. (For example, we can limit the number of blocks used to build the solution.)

[13] http://kodetu.org/

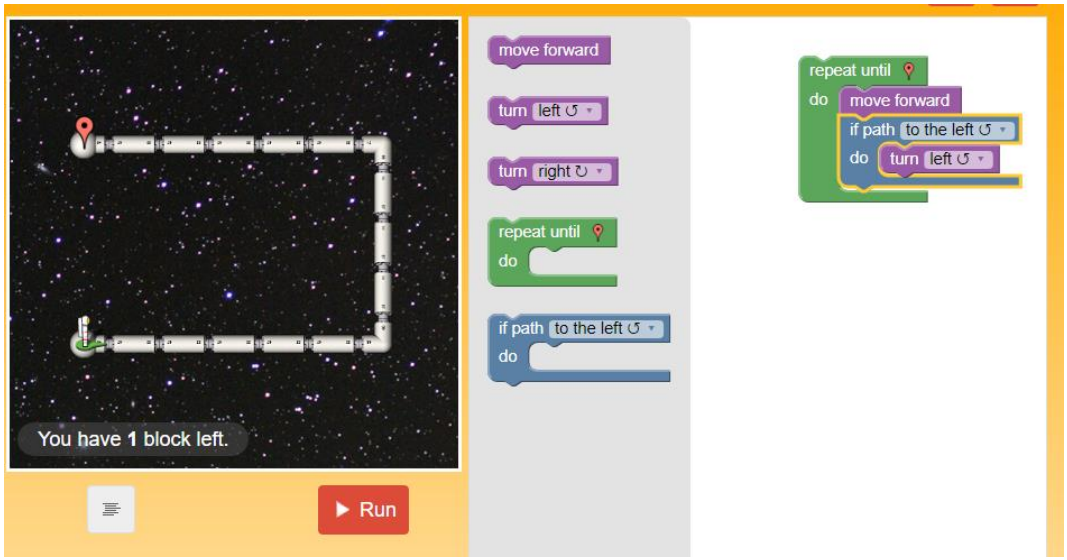

Figure 3.1 The Kodetu interface. Mazed-based challenge (left), available blocks to create the solution (middle) and the workspace where the user builds the program (right).

Kodetu allows the creation of new challenges, as well as the use of gamification techniques in an easy-to-use interface (Figure 3.2). It is simple, using only the necessary buttons and there is consistency between the button icons and their functionality. The elements are carefully placed and the colors are strategically chosen. Furthermore, the system gives immediate feedback when an action is performed.

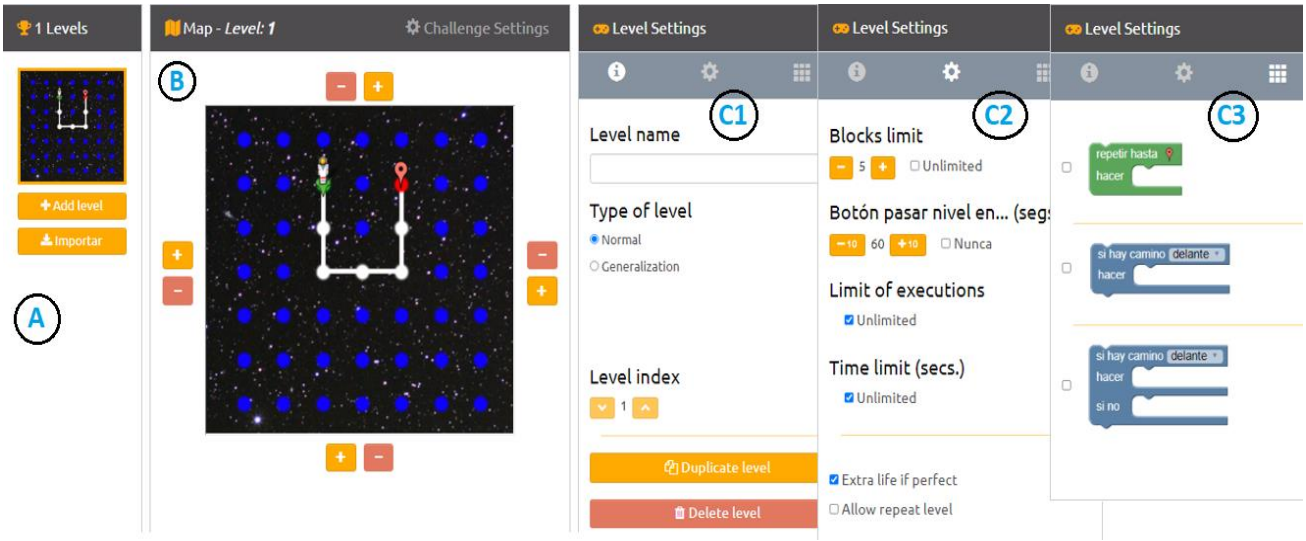

Figure 3.2 Challenge creation interface: A: Add new challenges, B: Create maze, C1: General challenge settings, C2, Challenge limitations, C3: Blocks available.

Every user registered in the platform has access to the challenge creation interface that consists of three panels. Panel A shown in figure 3.2, allows users to add new challenges; for each challenge, a new empty maze appears in Panel B. The users can increase or decrease the size of the maze. By clicking on the dots, the maze path is created. In order to save the challenge, it is obligatory to define the initial point of the maze by placing the astronaut's icon and the endpoint by placing the destination mark. Panel C has three different views; the first view is where the name and the type of the maze are set, the second view allows the user to add several limitations to the challenges such as a limitation on the number of blocks that can be used to solve the maze, and the third view allows the creator of the challenge to choose the blocks provided to the user.

All participants' data are gathered anonymously and stored in the database under a unique identifier that is automatically generated when the user accesses the platform. As the learners play with Kodetu, logs are kept of all the changes in the workspace. From these interactions, we obtain a detailed database that shows the progress of each learner in the different challenges. The information included in each log entry contains the timestamp, the level played, the code contained in the workspace at that time, and the result information of the attempt to solve the challenge (updated on pressing the execute button), with 4 possibilities: 1) success (the astronaut reaches the goal), 2) failure (the code ends without reaching the goal), 3) crash (the astronaut

falls into space) or 4) loop (the path is repeated indefinitely without solving the maze).

Thanks to its interaction logging recording and the challenge creation features, Kodetu is a valuable tool to research in terms of the development of CT in learners. However, we aim to extend its features by adding the ability to adapt to the capabilities and performance as learners progress in the learning path and, ultimately, automatically generate programming challenges personalized for every learner. This adaptive system will allow the complexity of each challenge to be dynamically adapted to learners. However, to achieve this, we must estimate the difficulty of a programming challenge in Kodetu.

3.2. Methodology

To estimate the difficulty of a programming challenge in Kodetu, we defined a 4-step procedure that is presented in detail in the following sections. To begin, we identified the characteristics that may affect the difficulty of programming challenges based on our experience with more than 19,000 participants throughout five years of using Kodetu (Eguiluz et al., 2017; Israel-Fishelson et al., 2020; Olivares-Rodríguez et al., 2018):

- the width and height of the maze,
- the total number of steps in the maze,
- the length of the optimal path (from the starting point to the exit),
- no turns on the optimal path/one-direction turns on the optimal path (that is, only right- or only left-direction turns)/two-direction turns on the optimal path,
- x-crosses (the possibility to move north, south, west, and east from a certain point of the maze),
- t-crosses (the possibility to move south, west and east from a certain point of the maze),
- maze loops (a path that allows users to go from one position in the maze to the same position used in the maze without passing through any previous position),
- blocks available (movement only blocks, loops + movement blocks, and conditionals + loops + movement blocks),

- block limits (the number of blocks allowed to be used in a maze challenge).

After that, we designed a set of challenge pairs that differ in only one of the aforementioned variables (e.g., a challenge pair with an identical maze size and the same number of available blocks but one of the challenges has one more maze loop) and conducted several workshops to make participants solve the challenges.

Later, we studied how the learners' performance varied according to the characteristics of each challenge (e.g., percentage of successes in the challenge, time required to solve the challenge, and number of attempts).

Finally, we estimated the difficulty of the challenges by analyzing the relationships between the characteristics of each challenge and the performance obtained by the participants. The following three sections describe a set of studies that we conducted following this procedure. The data obtained from the three studies allowed us to answer the research questions, measure the performance in the challenges, and obtain the difficulty function (Difficulty calculation 3.6).

3.3. Study 1

3.3.1. Methodology

This study is our first approach to analyzing the difficulty of Kodetu's challenges. We investigated whether and how much maze loops affect the performance in a maze-based programming challenge. We designed a total of 34 challenges in Kodetu (Figure 3.3), separated into pairs that differed only in the number of maze loops. (For example, one challenge is defined as {width: 7, height: 7, optimal path: 24, total steps: 24, maze loops: 0, x-crosses: 0, t-crosses: 0, turns: 2, no block limit, blocks: all available} and another challenge is exactly the same but instead of 0 maze loops, it has 2 maze loops). With these 34 challenges, we prepared 7 sequences of 5 challenges each. (Because $7 \times 5 = 35$, one of the challenges was part of two sequences).

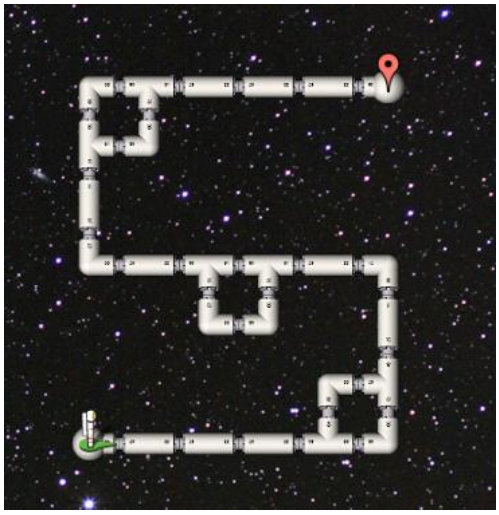

Figure 3.3 Example of a challenge with 3 maze loops.

We designed the sequences (Figure 3.4) following these principles:

- The challenges were arranged in an order in the sequence that aimed to achieve increasing difficulty and a smooth transition from one challenge to the next.
- A special emphasis was devoted to balancing the complexity between the sequences. Levels with similar values of the variables related to the maze, were distributed amongst the sequences to achieve homogeneity between the 7 sequences developed.
- Each sequence is separated into two parts. The first levels of the sequence do not have block limitation and the last challenges have block limitation. Introducing block limitation at a challenge means that participants are not able to use as many blocks as they need and therefore are forced to use the conditional blocks if-then, if-then-else, and loop blocks to successfully solve the maze. This makes the challenges more demanding.
- Each challenge of the sequence must be as less similar as possible to the other challenges in the same sequence so that knowledge obtained from previous challenges will not affect the participant's performance.

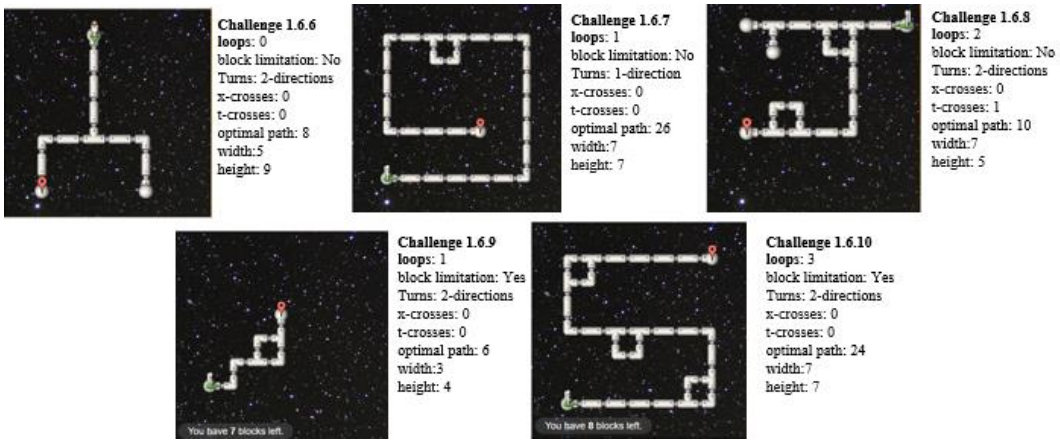

Figure 3.4 Example of a sequence-Study 1.

A total of 70 participants aged between 11 and 15 years old (44% female, 56% male) had 30 minutes to solve ten challenges in Kodetu. To measure the performance in the challenges, we acknowledged the frustration that can be caused to the learners as a result of the lack of familiarity with the platform and how this can affect the performance. Thus, we designed five introductory challenges (Figure 3.5) of minor difficulty that were put at the beginning of the sequences. In these introductory challenges, all the necessary code and game elements are introduced. The first challenge was explained step by step so that the participants would better understand what they had to do at the next levels. Upon the completion of the introductory challenges, participants started playing with the challenges designed to conduct the experiment.

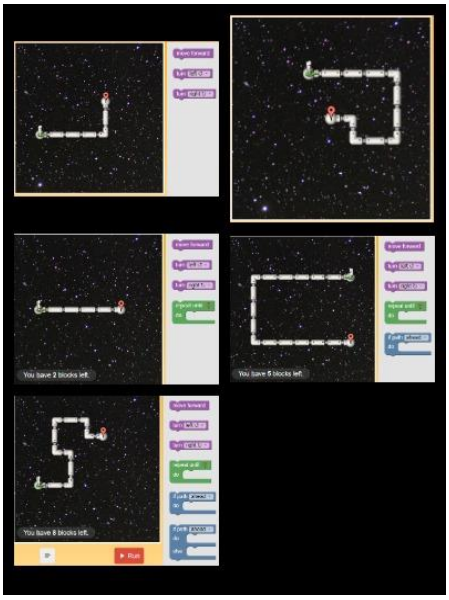

Figure 3.5 Introductory challenges - Study 1.

3.3.2. Data cleansing

After finishing the experiment, we carried out the process of capturing and filtering all information recorded on our server. We performed the following filtering and verification actions on the original dataset:

1. We verified that all learners entered the system by introducing the group code given during the intervention.
2. We verified that the code recorded to solve the maze (and the interactions to create it) effectively solve the corresponding mazes.
3. Due to the fact that several learners use the browser's back button and go back to previous levels and that there are network and connection problems that cause interactions to get lost, we checked that the sequence of interactions of each learner is at the correct challenge and that there is a successful solution to the previous one.
4. All learners' sessions are reviewed, checking if there are excessive time breaks (more than 30 minutes between an interaction and the next), or if sessions are too short (less than 5 interactions on a challenge).

3.3.3. Results and discussion

Table 3.1 presents the percentage of loss in each challenge (the percentage of users who failed to pass a challenge, considering the participants who entered the challenge).

Sequence	Challenge 6	Challenge 7	Challenge 8	Challenge 9	Challenge 10
1.1	11%	0%	13%	71%	50%
1.2	0%	27%	13%	71%	100%
1.3	0%	0%	33%	0%	33%
1.4	0%	0%	0%	0%	38%
1.5	0%	0%	45%	0%	67%
1.6	14%	33%	0%	50%	100%
1.7	11%	13%	0%	29%	40%

Table 3.1 Percentage of participants who failed to solve a challenge, Study 1.

Initially, to determine whether the number of maze loops in a challenge affects success, we conducted Pearson correlation test. The results of the test indicated that there was not a significant association between the maze loops and the percentage of success (p-value = 0.184).

To further investigate whether the number of maze loops affect the success, we performed one-way ANOVA in which the dependent variable was the percentage of success in a challenge and the independent variable was the number of maze loops (four groups: 0, 1, 2, or 3 maze loops in each challenge). The results showed that no statistically significant difference existed between groups [$F(3,31) = 0.705$, $p = 0.556$]. We ran another one-way ANOVA in which the groups of the independent variables were challenges with no maze loops and challenges with maze loops. The results showed that no statistically significant difference existed between the groups [$F(1,33) = 1.604$, $p = 0.214$].

Despite the noticeable decline in the success rate of learners in the last challenges, these results suggest that the presence of maze loops in Kodetu challenges does not result in an added difficulty for participants.

There are several limitations to this study. First, the lack of sufficient time to perform the 10 challenges (5 training and 5 to test) could explain the low success rates in the last challenges of each sequence. Second, the order of the challenges in each sequence

may have prevented us from reaching conclusions regarding the difficulty of each challenge. The challenges were ordered based on our initial assumptions regarding their difficulty, but we found sequences in which 67% of the participants succeeded in one challenge while the next was passed by 100% of participants who reached it. Therefore, this 100% success rate informs us only that the second challenge is not more difficult than the previous one; however, we cannot conclude whether it is perceived as being much easier, slightly easier, or of equivalent difficulty because those who passed it were those who also passed the previous challenge. Considering the above, in the following study, we increased the time available to solve the challenges to 60 minutes, increased the number of test challenges from five to seven (duplicating the session duration is adequate for adding two more challenges), and made an effort to better define sequences of increasingly difficult challenges.

3.4. Study 2

3.4.1. Methodology

Taking into consideration the limitations of the previous study, we performed the next study where we sought to answer the research questions:

1. How do maze characteristics (width, height, total number of steps in the maze, optimal path, turns, and numbers of x-crosses and t-crosses) affect the performance of learners in a maze-based programming challenge?
2. How do coding limitations (blocks provided and block limit) affect the performance in a maze-based programming challenge?

Therefore, we designed 40 challenges following the same principles of Study 1 (for example, 20 pairs of challenges where all the variables were the same except for one), and we created 6 sequences of 7 challenges each. (Because $6 \times 7 = 42$, two of the challenges were part of two sequences).

A total of 197 participants aged 9 to 11 years old (49% female, 49% male, and 2% other) had 60 minutes to solve 12 challenges in Kodetu. The first five challenges were training challenges with no block limit (Figure 3.6), and they were not considered in the analysis. The first challenge was explained step-by-step so that the functioning of Kodetu was understood. The next seven challenges corresponded to one of the six challenge sequences randomly assigned to each participant.

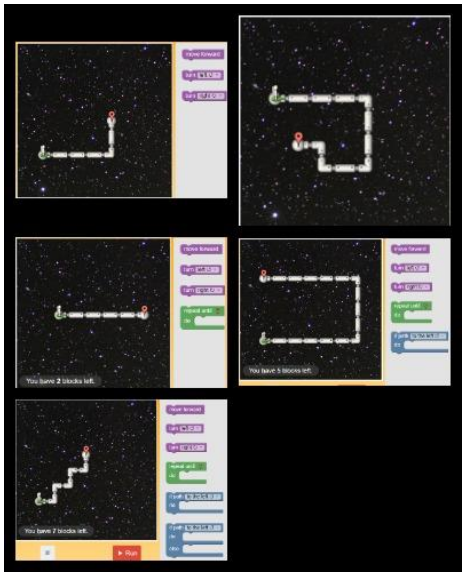

Figure 3.6 Introductory levels - Study 2.

3.4.2. Results and discussion

After the completion of the experiments, we followed the data cleansing process described in section 3.3.2. To investigate the effect of the variables, we performed several statistical tests. We conducted a one-way ANOVA test to compare the effect of the turns on the percentage of success on challenges with no turns, one-direction turns, and two-direction turns. There was a significant effect of the type of turn on the success rate at the p-value$<$0.05 level for the three conditions [$F(2,39) = 3.722$, p-value $= 0.033$]. However, we did not find a significant effect using the Tukey HSD and Duncan post hoc tests in terms of pairwise comparisons.

An analysis of variance showed that the effect of the blocks available was significant [$F(2,39) = 20.032$, p-value $= 0.000$]. Post hoc analyses using the Tukey HSD post hoc criterion for significance indicated that the success percentage was significantly higher in the conditions in which movement only blocks (go forward-turn left-turn right) ($M = 0.934$, $SD = 0.0755$) and loops + movement blocks ($M = 0.945$, $SD = 0.0544$) were provided than in the other condition (conditionals + loops + movement blocks) in which $M = 0.55$ and $SD = 0.2899$.

Regarding the block limit, statistically significant differences in the means of the challenges with and without a block limit were observed with $F(1,40) = 17.902$

with p-value = 0.000. One-way ANOVA showed that the analysis was not significant for the effect of the numbers of x-crosses [$F(3,38) = 0.978$, p-value = 0.413] and t-crosses [$F(3,38) = 2.034$, p-value = 0.125] on success.

As for the remaining variables, considering that they were not divided into groups, we performed correlation tests to evaluate the association between these variables and the success rate. The success rate and maze width were weakly negatively correlated ($r = -0.327$, p-value = 0.035) whereas the success rate and height were not correlated ($r = -0.205$, p-value = 0.194). The success rate and optimal path were also moderately negatively correlated ($r = -0.464$, p-value = 0.002), and the same occurred for the success rate and number of steps in the maze ($r = -0.506$, p-value = 0.001). Furthermore, the percentage of loss in each challenge is presented in Table 3.2.

Sequence	Challenge 1	Challenge 2	Challenge 3	Challenge 4	Challenge 5	Challenge 6	Challenge 7
2.1	0%	9%	3%	10%	4%	59%	100%
2.2	15%	0%	0%	26%	33%	25%	90%
2.3	5%	3%	0%	13%	36%	17%	38%
2.4	8%	0%	0%	13%	26%	0%	73%
2.5	16%	4%	20%	50%	10%	11%	50%
2.6	3%	3%	0%	48%	10%	5%	89%

Table 3.2 Percentage of participants who fail to solve a certain challenge, Study 2.

In Table 3.3, we present additional data metrics for each challenge, which we will use to calculate the participants' performance presented in the results section: average success time (time required to solve a challenge), average number of attempts that each participant needed to solve it (how many times participants pressed the ''play'' button to execute the program in the workspace), and average number of interactions with the workspace in which each participant engaged (blocks added or deleted in the workspace).

Sequence	Data metrics[14]	Challenge 1	Challenge 2	Challenge 3	Challenge 4	Challenge 5	Challenge 6	Challenge 7
2.1	AST	00:06:09	00:01:42	00:01:15	00:03:43	00:00:27	00:08:45	Not solved
	ANA	4.11	2.09	1.91	2.10	1.11	12.56	8.50
	ANI	99.83	59.20	47.84	133.29	14.89	212.04	209.30
2.2	AST	00:05:56	00:02:08	00:01:14	00:04:56	00:03:54	00:01:45	00:12:44
	ANA	4.22	3.35	2.00	3.65	6.67	2.33	8.30
	ANI	102.26	86.13	52.09	168.22	108.22	61.58	144.40
2.3	AST	00:02:30	00:01:31	00:01:02	00:04:27	00:06:55	00:06:29	00:07:35
	ANA	3.84	2.26	2.15	8.41	9.96	9.94	7.23
	ANI	53.57	57.91	41.73	157.91	194.29	169.50	123.38

[14] AST= average success time, ANA= average number of attempts, ANI= average number of interactions

2.4	AST	00:05:50	00:01:10	00:00:38	00:04:04	00:06:42	00:00:25	00:03:41
	ANA	4.52	2.13	1.13	2.61	7.11	1.08	4.55
	ANI	92.00	42.26	28.26	135.00	138.32	15.42	152.91
2.5	AST	00:06:43	00:00:41	00:04:16	00:04:38	00:04:01	00:03:32	00:02:14
	ANA	5.47	1.04	3.68	5.65	8.90	6.89	10.00
	ANI	123.81	28.85	152.32	113.10	129.50	123.56	146.25
2.6	AST	00:02:01	00:02:41	00:01:21	00:07:05	00:01:58	00:00:37	00:15:26
	ANA	2.53	4.62	2.37	9.83	1.86	1.84	10.67
	ANI	36.65	99.03	54.61	184.65	39.67	18.37	205.39

Table 3.3 Data metrics used to calculate performance, Study 2.

From the results shown in Table 3.2 and Table 3.3, we infer that the last challenges of the sequences exceeded the participants' capabilities in many cases. This was not influenced by the time available as 60 minutes was sufficient and an appropriate duration for this study. In the case of sequence 1, 100% of the participants were unable to solve the last challenge, so we could identify a "floor effect" that might be

affected by the participants' age. Considering this, Study 3 replicated Study 2 but increased the age of participants from 9-11 to 15-16 years old.

3.5. Study 3

3.5.1. Methodology

This study is a replica of Study 2 with older participants. A total of 59 participants aged 15-16 (37% female, 59% male, and 3% other) had 60 min to solve twelve challenges in Kodetu. The first five challenges were training challenges with no block limit and were not considered for analysis. (The first challenge was explained step by step so that the functioning of Kodetu was understood.) The last seven challenges corresponded to one of the six challenge sequences prepared before and randomly assigned to each participant.

3.5.2. Results and discussion

Table 3.4 shows the percentage of participants who started the study and failed to overcome each challenge, and Table 3.5 presents additional data metrics regarding the performance of the participants in Study 3.

Sequence	Challenge 1	Challenge 2	Challenge 3	Challenge 4	Challenge 5	Challenge 6	Challenge 7
3.1	0%	0%	11%	0%	0%	13%	57%
3.2	13%	0%	0%	14%	0%	0%	17%
3.3	0%	0%	0%	0%	0%	10%	44%
3.4	0%	0%	8%	0%	0%	0%	27%
3.5	0%	0%	13%	14%	17%	0%	20%
3.6	0%	0%	0%	0%	0%	0%	58%

Table 3.4 Percentage of Participants Who Failed to Complete a Challenge, Study 3.

Sequence	Data metrics[15]	Challenge 1	Challenge 2	Challenge 3	Challenge 4	Challenge 5	Challenge 6	Challenge 7
3.1	AST	00:02:33	00:01:05	00:01:08	00:03:57	00:00:21	00:04:36	00:14:28
	ANA	3.22	2.22	1.89	3.75	1.00	7.75	12.29
	ANI	73.78	66.78	54.44	151.88	11.50	158.13	327.29
3.2	AST	00:02:56	00:01:15	00:00:46	00:01:47	00:02:04	00:00:33	00:09:40
	ANA	2.25	2.00	1.57	1.14	3.67	1.00	13.17
	ANI	74.50	77.86	46.43	128.57	66.33	21.67	233.83

[15] AST= average success time, ANA= average number of attempts, ANI= average number of interactions

3.3	AST	00:04:32	00:00:45	00:00:42	00:02:20	00:04:26	00:02:06	00:09:24
	ANA	3.10	2.10	2.40	5.00	10.10	8.50	12.56
	ANI	39.40	41.30	37.10	94.80	171.20	144.20	266.67
3.4	AST	00:02:15	00:00:47	00:00:37	00:02:07	00:02:19	00:00:16	00:06:32
	ANA	1.33	1.33	1.25	1.09	3.55	1.00	8.18
	ANI	69.33	41.58	35.08	154.45	59.73	7.36	244.82
3.5	AST	00:02:56	00:00:46	00:02:22	00:03:27	00:04:47	00:01:28	00:03:41
	ANA	1.88	1.13	1.75	3.86	6.83	2.00	7.40
	ANI	66.63	36.00	116.38	98.14	141.50	60.80	131.00
3.6	AST	00:00:57	00:01:39	00:00:45	00:02:47	00:01:42	00:00:31	00:15:25
	ANA	2.00	3.25	1.83	3.25	1.67	1.83	17.33
	ANI	28.42	81.75	49.75	83.33	49.42	19.75	372.25

Table 3.5 Data metrics used to calculate performance, Study 3.

As the tables show, the vast majority of participants (85%) succeeded in the first challenges and became stuck in the last challenge, where they consumed the rest of the available time. With the results of Studies 2 and 3, we infer the difficulty associated with each challenge because the young participants of Study 2 suffered from a ''floor effect'' (low success rate in challenges too complex for their level of competence) while the participants of Study 3 suffered from a ''ceiling effect'' (high success rate in challenges too simple for their level of competence).

In order to analyze the differences in the results between Studies 2 and 3, we conducted a one-way ANOVA to compare the effect of age on success in the group of 9- to 11-year-olds and in the group of 15- to 16-year-olds. We found that age had a significant effect on success at the p-value<0.05 level for the two groups [$F(1,82) = 8.437$, p-value = 0.005].

3.6. Difficulty calculation

3.6.1. Methodology

As we saw in the previous analysis, the characteristics of each challenge had an influence on the participant's success rate. However, we must distinguish between the success rate and the participant's performance. Success means having solved a challenge, while performance involves more parameters, such as the time required to succeed, the number of attempts to solve a challenge, and the interactions with the workspace.

To calculate the performance based on the data collected, we created a fuzzy rule-based system (FRBS). The FRBS works by using rules to encode knowledge from a broad area into an automated system (Jiang et al., 2020). Unlike traditional logical systems, fuzzy logic (FL) systems attempt to model the imprecise ways of reasoning which play an essential role in the remarkable human ability to make reasonable decisions in an environment of uncertainty and imprecision (Zadeh, 1988).

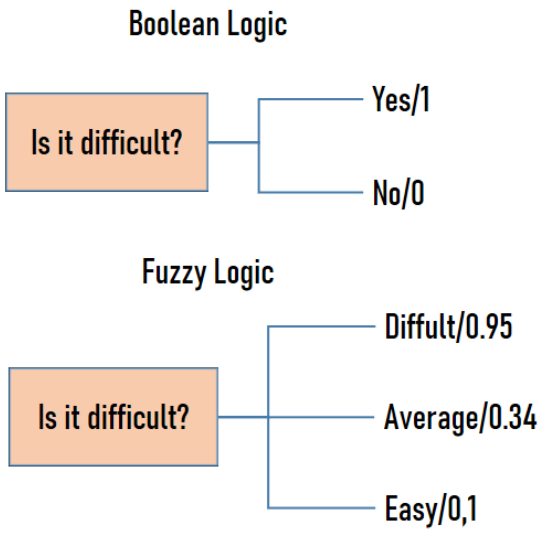

Figure 3.7 Boolean and Fuzzy Logic approach example.

The fuzzy rule-based expert systems are the simplest form of Artificial intelligence. An FRBS is a rule-based system in which fuzzy sets and FL are used to represent various types of information about the situation at hand, as well as to simulate the interactions and relationships that exist between its variables (Magdalena, 2015).

In the field of education, it has been used to create new performance evaluation models (Fourali, 1997). Gokmen et al. (2010) calculated the performance of an exam of students from the Technical Education Faculty of Marmara University, using the classical method (evaluation as success or failure) and an FL method. The findings of the analysis revealed differences between the conventional and FL methodologies. Although FL performance evaluation is complex and requires additional software, it has some advantages. The classical method stuck to fixed mathematical calculation, whereas FL evaluation was flexible and provided several evaluation possibilities. Furthermore, changing different parameters of the FL system, the professor was able to permit a more flexible and objective performance evaluation.

Troussas et al. (2020) used fuzzy-rule based methods to create an intelligent mobile GBL application for assessing and advancing learners' knowledge in the programming language C#. FL was used in order to provide a tailored advice system based on the misconception performed by learners and their current knowledge level. The mobile application was used by university students during an academic semester to aid in the learning of an undergraduate C# course. The results showed that the tailor advice recommendation system is an important aspect that provided learners with individualized learning and improved their knowledge and skills.

In our case, we use the FRBS to solve the ambiguity of defining ‘‘low’’ or ‘‘high’’ performance in a tool such as Kodetu automatically. Using our FRBS, we obtain the value of performance (a number between 0-100) in each challenge as an output variable given the values of the input variables. To define our FRBS, we take four variables as the input: the number of attempts per participant, the number of interactions per participant, the loss per challenge, and the average success time on a challenge. The output is the participant’s performance in each challenge. Then, we map a given input to an output using fuzzy logic. To build the FRBS, we used the frbs R package. We created two identical FRBSs with the only difference being the input

data. In the first system, we used the data from Study 2; while in the other system, we used the data from Study 3.

The FRBS consists of four functional parts. First, the fuzzification interface (**fuzzifier**) transforms the crisp inputs into degrees of membership functions (MFs) of the linguistic label of each variable. The MFs are shown in Figure 3.8 all fuzzy input variables contain four MFs for each of the four associated linguistic labels: low, medium, high, and any. The linguistic label "any" contains all values of the variable and is used when, in a particular fuzzy rule, the corresponding variable is not significant and changes in the value should not be considered as an important factor to determine the performance.

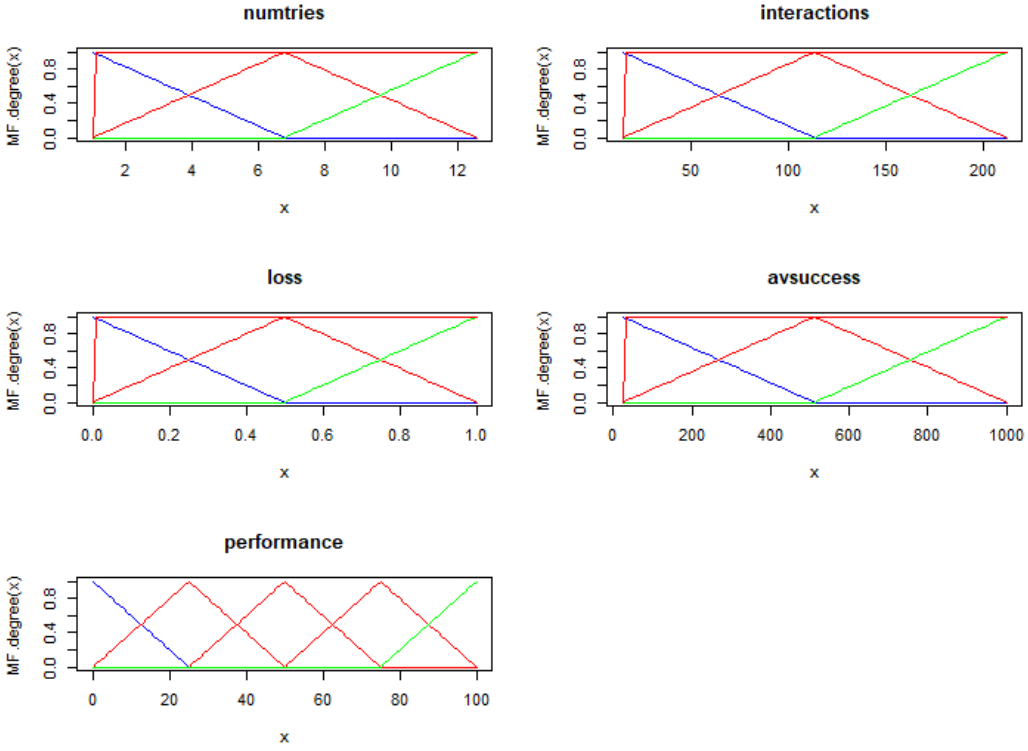

Figure 3.8 Plot of membership functions-FRBS.

Second, the knowledge base consists of the **database** and the **rulebase**. The database shown in Table 3.6 includes the fuzzy set definitions while the rulebase in Table 3.7 contains 12 fuzzy IF-THEN rules. These rules express the experts' knowledge in a form that the system can understand.

Variable	Type	Linguistic label
Number of attempts per participant	Input	Low Medium High Any
Interactions per participant	Input	Low Medium High Any
Loss (percentage)	Input	Low Medium High Any
Average success time	Input	Low Medium High Any
Performance	Output	Very high High Average Low Very low

Table 3.6 Fuzzy Database (Input/Output set variables).

Rule number	Rulebase
1	IF number of attempts is high and interactions is high and loss is high and average success time is high THEN performance is very low
2	IF number of attempts is high and interactions is high and loss is medium and average success time is high THEN performance is very low
3	IF number of attempts is medium and interactions is high and loss is medium and average success time is high THEN performance is low
4	IF number of attempts is medium and interactions is any and loss is high and average success time is any THEN performance is low
5	IF number of attempts is medium and interactions is medium and loss is medium and average success time is medium THEN performance is average
6	IF number of attempts is low and interactions is medium and loss is medium and average success time is medium THEN performance is average
7	IF number of attempts is high and interactions is medium and loss is medium and average success time is medium THEN performance is average
8	IF number of attempts is low and interactions is medium and loss is medium and average success time is low THEN performance is average

9	IF number of attempts is low and interactions is medium and loss is low and average success time is medium THEN performance is high
10	IF number of attempts is medium and interactions is medium and loss is low and average success time is low THEN performance is high
11	IF number of attempts is low and interactions is low and loss is medium and average success time is low THEN performance is very high
12	IF number of attempts is low and interactions is low and loss is low and average success time is low THEN performance is very high

Table 3.7 Fuzzy rulebase specified from experts' knowledge.

Third, the **Mamdani inference engine** performs the inference operations on the fuzzy IF-THEN rules. The Mamdani engine was selected because systems that use the Mamdani engine are designed to incorporate the form of the rulebase that we used in part 3, expressed in natural language.

Fourth, the defuzzification process (**defuzzifier**) center of gravity (**COG**), which is the standard method by which Mamdani systems obtain crisp values from linguistic values, is used.

The loss rate was one of the input variables for the FRBS that we used to measure the performance. However, there is a problem when the loss rate of challenge n-1 is larger than the loss of challenge n. Many factors could cause this issue that prevents us from using the loss rate of challenge n as an indicator of the performance in that challenge. Accordingly, the decision was made to set a low boundary: for the challenges in which the value of the loss rate was equal to or less than 20% of the value of the loss rate at the previous challenge, we reran the FRBS but without using the loss rate as an input variable. In Study 2, three challenges (2.4.6, 2.5.5, and 2.6.5) were affected by this issue.

3.6.2. Results

In order to answer our research questions, we have to calculate the performance of the participants on the maze-based programming challenges using the FRBS system presented in the previous section. Using the Mamdani inference method, the COG for defuzzification (output processor), and the rules based on the 12 linguistic propositions displayed in Table 3.7, we obtained the crisp output values for each of the 42 challenges, which represent the performances of the participants on each challenge. The crisp output values ranged from 0 to 100, with 0 being the lowest performance and 100 being the highest (Table 3.8 and Table 3.9).

Sequence	Challenge 1	Challenge 2	Challenge 3	Challenge 4	Challenge 5	Challenge 6	Challenge 7
2.1	78.23	91.76	95.53	70.06	100	0	20.66
2.2	69.08	81.42	97.04	58.33	59.12	80.17	22.26
2.3	89.85	93.73	97.07	69.86	58.33	56.78	51.79
2.4	78.87	97.1	99.60	69.39	54.48	99.87	32.61
2.5	56.62	96.81	67.59	50.18	44.25	71.21	50
2.6	94.85	77.63	95.95	51.68	86.55	98.36	7.22

Table 3.8 Crisp output of frbs - performance values, Study 2.

Sequence	Challenge 1	Challenge 2	Challenge 3	Challenge 4	Challenge 5	Challenge 6	Challenge 7
3.1	95.23	97.5	88.5	79.4	99.87	62.66	18.13
3.2	83.38	97.6	98.93	66.62	95.16	99.5	36.62
3.3	95.81	98.09	97.76	92.01	75	71.03	31.98
3.4	95.88	99.01	93.52	82.11	95.69	100	51.18
3.5	94.99	99.05	69.44	79.68	61.82	97.5	62.03
3.6	98.46	95.28	98.44	94.45	97.26	99.09	0

Table 3.9 Crisp output of frbs - performance values, Study 3.

Although the one-way ANOVA to compare the effect of age on performance in Studies 2 and 3 is statistically significant [$F(1,82) = 4.674$, p-value $= 0.034$), in the overall ranking of the challenges based on the performance of participants in both studies, we observed that 52% of the challenges differ by less than two positions in the ranking, 31% differ by three to seven positions, and the rest differ by more than eight positions (For instance, the success rate of challenge 2.4.7 is 27% and the success rate of 3.4.7 is 73%; however, both challenges rank 5th on the overall ranking of the challenges).

Once the performance of the challenges is calculated, we conducted a simple regression analysis to determine the correlations between the dependent variable performance and the independent variables defined in the methodology section and investigated them in Studies 2 and 3 (width, height, total number of steps of the maze, length of the optimal path, turns, x-crosses, t-crosses, blocks available and block limit). The relationships between performance and the variables enable us to answer the research questions set at the beginning of the investigation.

Several variables were excluded from the analysis because they did not meet the assumptions of the regression analysis according to (Osbourne & Waters, 2002) (see Table 3.10). Thus, the independent variables that were used were the following: the number of steps, turns, and blocks available.

Variable name	Rejection reason
Width	No linear relationship between the dependent and the independent variable.
Height	No linear relationship between the dependent and the independent variable.
Optimal path	Strong correlation with independent variable number of steps (Pearson coefficient = 0.96)
x-crosses	No linear relationship between the dependent and the independent variable.
t-crosses	No linear relationship between the dependent and the independent variable.
Block limit	Strong correlation with independent variable blocks (Pearson coefficient = 0.83)

Table 3.10 Variables not used in the regression analysis.

We conducted a simple regression analysis using the data of Study 2 and another analysis with those of Study 3. The regression analysis will determine whether and how much these variables affect the performance, described in the form of a function.

Regarding the performance of the participants in Study 2, a significant regression equation was found [$F(3,38) = 25.408$, p-value < 0.000] with an R^2 of 0.667. Considering this, the participants' predicted performance in Study 2 is defined by the following equation:

$$performance_{study2} = 120.848 - 0.985 * num\_steps - 8.289 * turns - 4.076 * blocks \quad (1)$$

Regarding the performance of the participants in Study 3, a regression equation was found [$F(2, 39) = 15.61$, p-value < 0.001] with an R^2 of 0.445. The learners' predicted performance in Study 3 is defined by the following equation:

$$performance_{study3} = 119.122 - 1.274 * num\_steps - 2.752 * blocks \quad (2)$$

To evaluate both models, we calculated the Mean Absolute Error (MAE) (Chai & Draxler, 2014; Willmott & Matsuura, 2005) for (1) ($MAE_1 = 10.528$) and for (2) ($MAE_2 = 11.989$). Since the lower the MAE is the better the model ($MAE_1 < MAE_2$) and the low R^2 (0.445) of (2), we proceeded to define the difficulty function using (1).

Considering (1) and the results from the research of Latham, Seijts, and Crim (2008) indicating that the higher the complexity of a task is, the lower the person's performance in that specific task, we infer that performance can be used to estimate difficulty:

$$dif = -performance \Rightarrow dif = -120.848 + 0.985 * num\_steps + 8.289 * turns + 4.076 * blocks \quad (3)$$

Moreover, given that previous efforts to define difficulty in games (Aponte et al., 2011; Constant et al., 2017; Gallego-Durán et al., 2018; Pelánek & Effenberger, 2020; Szabó et al., 2016) are time-dependent and that research has shown that limiting the time of an activity affects performance (Davidson & Carroll, 1945; DeDonno et al., 2014; Mullane & McKelvie, 2019; Powers & Fowles, 1996), we suggest that difficulty also depends on the time given to solve the challenge. Using linear regression, we predicted the average success time (ASt) given the maze and coding

characteristics of a challenge. A regression equation was found [$F(2, 39) = 14.79$, p-value $= 0.000$] with an R^2 of 0.5012:

$$ASt = 14.303 * num\_steps + 23.891 * blocks \quad (4)$$

Having predicted an estimation of the average success time (ASt), we infer that by introducing a time limit that equals the ASt, half of the participants will be able to succeed at that level. Therefore, if we grant more time, the percentage of participants who succeed in the challenge will be higher and vice versa. However, if only 50% of the participants succeeded in a challenge, we considered it to be difficult. Similarly, we assume that if the time limit corresponds with an extra time of 25% of the ASt, we consider the time limit to have no negative impact on the difficulty of the challenge; in addition, if participants have more than 25% extra time regarding the ASt, the difficulty of the challenge will be lower. Considering this, we moderate the factor of time limit with a coefficient of 0.8. Consequently, our estimation for the difficulty function that considers time is:

$$diftime = dif * \frac{ASt}{0.8 * time\_limit} \quad (5)$$

3.7. Discussion

The research presented in this chapter provides a quantitative analysis of data obtained from the Kodetu platform, which advances our understanding of the maze characteristics and coding limitations that affect learners' performance in maze-based programming challenges. By measuring this effect, we propose an estimation for the difficulty of block-based maze programming challenges. Our results are based on the analysis of the platform log data gathered from 326 learners during 3 studies in which participants were tasked with solving maze-based programming challenges in the online platform Kodetu.

After analyzing the data obtained from Study 1, we found that the existence of maze loops in the challenges did not affect learners' success rate. Thus, the high failure rate, especially in the last challenges of the sequences, cannot be explained by the maze loops. One of the reasons for this finding may be that as long as learners can cognitively solve the challenge, the maze loops do not affect their performance. Furthermore, considering the fact that maze loops have not affected the learners'

performance, we propose that the high failure rate was caused by the limited time given to complete the sequences, as well as the effect of the rest of the variables present in the challenges.

The results from Study 2 indicated that challenges that provide conditionals and loop blocks (in addition to movement blocks), as well as challenges with block limits, are demanding in terms of the time to succeed, the number of interactions with the platform, and the number of attempts to solve them. This confirms the results from prior research (Chan et al., 2020; Pelánek & Effenberger, 2020) as the use of blocks of conditionals and loops to solve a challenge requires challenging CT competencies (Moreno-León & Robles, 2015; Wilson et al., 2012). In addition, the difficulty added by the block limit is because the learners are forced to use conditional and loop blocks to solve the maze instead of using only the sequence of movement blocks. Furthermore, the data analysis shows that turns in the optimal path affect learners' performance in a challenge; however, it is not significant if there are one- or two-direction turns. This suggests that as long as the optimal path is not a straight line, the challenge is complex despite the direction of the turns.

We analyzed the data in more depth by measuring the success rate, the loss rate, the average success time, the number of interactions in a challenge, and the number of attempts to solve a challenge. We also found that the sequences of challenges that we created were too complex for most of the participants (''floor effect''). Considering this, we conducted the same experiment with older participants (Study 3).

Unlike what happened in Study 2, we observed that the success rate in Study 3 during the first challenges was very high, and almost every participant was able to solve them (''ceiling effect''). However, in the last challenges of the sequences, there was a remarkable increase in the time necessary to succeed, the number of interactions, and the number of attempts to solve a challenge. This indicates that age affects the success rate and that the challenges requiring higher CT competencies are also demanding for older participants.

We developed an FRBS to calculate the performance based on the data metrics of the average success time, loss, interactions, and number of attempts. We noted that although the value of the performance in each challenge differs depending on the

learners' age (Navarro et al., 2015), the overall ranking of challenges based on performance is similar, showing that the CT competencies required to solve a maze-based programming challenge are difficult to achieve for both younger and older learners.

Finally, the main finding of this research was the definition of an estimation of the difficulty of block-based maze programming challenges. According to the results of the regression analysis and previous literature, a challenge is difficult when it contains turns, when the total number of steps is substantial, and when movement, conditionals and loop blocks are provided to the learner; thus, our estimation of difficulty is presented in (3).

Considering that we wanted to focus our research on investigating the effect of the maze characteristics and the coding limitations of the game, we designed our three studies without setting a time limit for the completion of each challenge. However, putting a time limitation to solve a challenge is considered a factor that affects the difficulty of the challenge (Davidson & Carroll, 1945; DeDonno et al., 2014; Mullane & McKelvie, 2019; Powers & Fowles, 1996). Therefore, we estimated a time-dependent function of difficulty as (5). We provide an estimation of the average time to succeed in a challenge based on the characteristics of the maze and the coding limitations, and we suggest that this estimation can be used as a threshold for choosing an adequate time limitation. The use of time limitations does not mean that increasing the time limit for a difficult challenge makes it easier; nevertheless, we suggest that the time limit should be considered an additional limitation to the learner.

The limitations of each study are already mentioned because they motivated the main changes in the design of the next study. However, we would like to further mention that due to the lack of specific rules for creating sequences of challenges in block-based maze games, the design of the sequences was conducted mostly in an empirical way which may have affected some participants' performance in the studies. However, we considered this limitation when defining the performance with the FRBS and minimized its further effect in our research.

Considering the prospects and limitations of this research, we next focused on creating an adaptive block-based maze game that will offer personalization of the

learning paths. Our research results encouraged the design of new experiments to explore the effects of providing personalized learning on the development of CT and we present them in detail in the next chapter.

Chapter

4

Block-based Maze Game with Adaptive Learning Paths

The aim of this chapter is twofold: we want to investigate whether adaptive learning paths based on learner's performance enhance the development of CT based on specific data metrics (success time, attempts, interactions and performance in the programming challenges); and by doing that, we also test and validate the difficulty function presented in Chapter 3. To accomplish this purpose, we developed an adaptive version of Kodetu and conducted several experiments with K-12 students. In the following sections, we will detail the algorithm of the system developed, the experimental set- up, as well as the results and conclusions.

4.1. Methodology

Adaptive educational games are used to continuously challenge learners based on their ability as a way to scaffold their learning (Kiili et al., 2012). Based on the Flow Theory (Csikszentmihalyi, 2000), the goal of an adaptive game is to provide learners with challenges that balance the difficulty with their skills so as to prevent boredom and anxiety. The complexity of the game is automatically adjusted in response to the game's continual evaluation of a learner's performance (Sampayo-Vargas et al., 2013). To create the adaptive Kodetu we were based on the approach of CAT systems (Chang & Ying, 1996; Linacre, 2000). Adaptive Kodetu is composed of challenges selected from a collection of challenges, known as challenge bank. The challenges are

chosen to match the estimated ability level of the current learner. They are based on the performance of the previous challenge and, depending on this performance, the next challenge is chosen, decreasing or increasing the difficulty.

4.1.1. System design

The adaptive Kodetu system consists of the following components: (a) the challenge bank, (b) the entry challenges, (c) the challenge selection rule, and (d) the scoring method.

A. Challenge bank.

Essential to the operation of the adaptive system is the Kodetu challenge bank. The bank comprises 110 challenges, 26 were already created and tested to the difficulty experiment presented in Chapter 3 and the rest 84 challenges were new, created for the purposes of this experiment.

The challenges were separated into three categories based on the type of blocks that needed to be solved:

1. Sequential: blocks to go forward, turn right, turn left.
2. Loops: perform loops to move toward the goal.
3. Conditionals: check whether there is a path in front, left, or right of the astronaut.

We calculated the difficulty of each challenge based on the difficulty function presented at Section 3.6:

Challenges without time limit

$$difficulty = -120.848 + 0.985 * num\_steps + 8.289 * turns + 4.076 * blocks \quad (1)$$

Challenges with time limit

$$difficulty_1 = difficulty * \frac{ASt}{0.8 * time\_limit} \quad (2),$$

where

$$ASt = 14.303 * num\_steps + 23.891 * blocks \quad (3),$$ the estimation of the average time to succeed in solving the challenge.

Having calculated the difficulty, we separated the challenges of each category into 4 ranges based on the difficulty value of each challenge. The ranges were defined by the quartiles (Joarder & Firozzaman, 2001); the four quarters that divided our challenges

into quartiles were the lowest 25% of difficulty value, the next lowest 25% up to the median, the second-highest 25% above the median, and the highest 25%. The categories and ranges of the challenge bank, as well as the number of challenges per range, are presented in Table 4.1.

Challenge Category	**Difficulty values**			
Sequential Total N=34	Range 1 <=16.428 N=9	Range 2 16.428<dif<=25.99 N=10	Range 3 25.99<dif<=34.670 N=9	Range 4 dif>34.760 N=6
Loops Total N=23	Range 5 <=22.117 N=7	Range 6 22.117<dif<=28.232 N=5	Range 7 28.232<dif<=34.101 N=5	Range 8 dif>34.101 N=6
Conditionals Total N=53	Range 9 <=56.916 N=14	Range 10 56.916<dif<=62.25 N=14	Range 11 62.25<dif<=71.706 N=13	Range 12 dif>71.706 N=12

Table 4.1 Classification of Kodetu challenges based on category and difficulty value and the number of challenges per Range (N).

Once we had created the challenges, we inserted them into the Kodetu game, using the challenge creation interface (Section 3.1). Four new fields were added (Figure 4.1): 1) the **difficulty** value for each challenge, 2) the prediction of the **average success time** for each challenge, 3) the **block type** i.e. the challenge category, and 4) the **difficulty range** to which the challenge belongs. These fields were necessary in order to provide adaptive learning paths to the learners and they will be more explained in the following paragraphs.

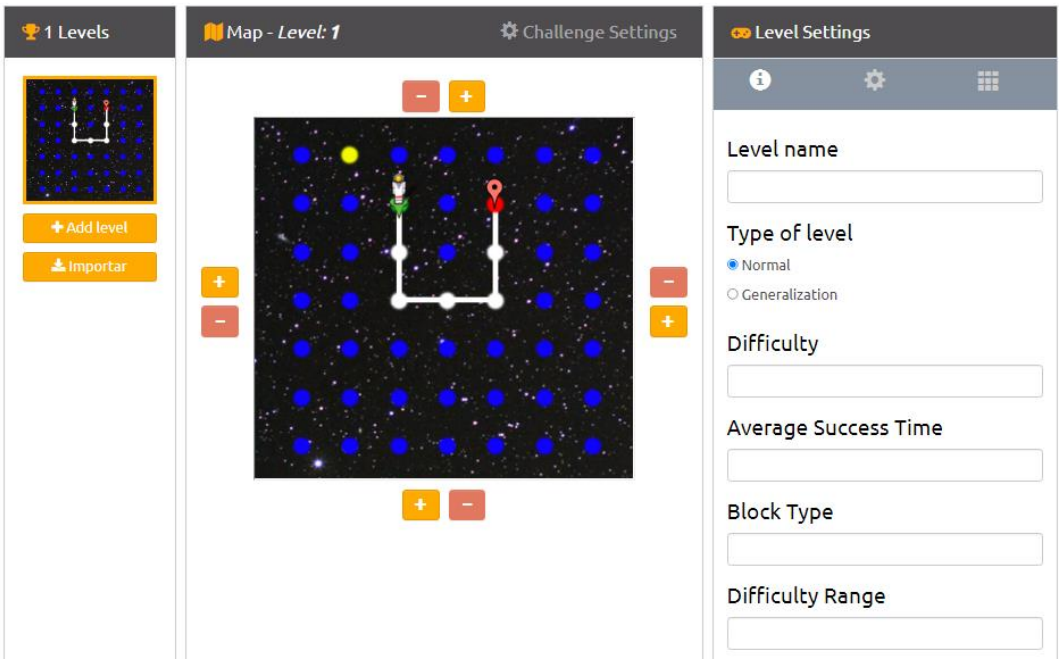

Figure 4.1 Challenge creation interface - Adaptive Kodetu.

B. Entry challenges

In the adaptive system, the challenges are selected based on the learner's performance. However, the system is obviously not able to make any specific estimate of the learner's ability when no challenges have been administered. Therefore, some other initial estimate of the learner's ability is necessary. Specifying accurately the entry challenges is important because accurate entry challenges reduces the number of challenges required to achieve precise learning paths (Linacre, 2000). Furthermore, the level of difficulty of the challenges that are chosen moves to the learner's trait level as the game progresses (Weiss & Kingsbury, 1984).

In our adaptive system, we provided 3 initial challenges that were common to all learners. The 3 challenges belonged to the first category (sequential) and range 2 but they had different difficulty values. Based on the learner's performance in these 3 challenges, we were able to estimate the learner's ability and start the learning path with the appropriate challenge for them.

C. Challenge selection rule and scoring method

The challenge selection rule along with the scoring method are the two steps of the adaptive Kodetu that are repeated each time a new challenge is solved. It determines

the most appropriate challenge to administer given the challenge that is currently shown to the learner and the learner's performance.

The scoring method is the method used to determine the challenge that will be presented next to the learner. It uses the learner's performance to the challenge previously played to provide the appropriate challenge, either a more difficult one in order to continue the learning process or an easier one to improve the learner's skill in this particular challenge category.

In our system, we used the prediction of the average success time of each challenge as the scoring method:

$$ASt = 14.303 * num\_steps + 23.891 * blocks \quad (1)$$

We calculated the prediction of ASt for all the challenges. For the 26 challenges that were already used in the difficulty experiment (Chapter 3), we used the actual AST and not the prediction. Having the estimation of the ASt for all challenges, we were able to create the challenge selection rule: *If the learner's success time to the current challenge is less than or equal to the ASt of the challenge then the challenge's range is increased by 2 and the next challenge is one of the challenges with difficulty value based on the new range. On the contrary, if the learner's success time to the current challenge is more than the ASt of the challenge, then the range is decreased by 1 and a challenge with a smaller difficulty value is provided.* That allows providing a challenging learning path more adequate to the learner's abilities, that increases engagement, motivation and most importantly improves the development of CT (Tenório et al., 2020).

4.1.2. Experimentation

A total of 60 participants aged between 9 and 11 years old (53% female, 43% male, 7% other) had 60 minutes to solve the adaptive learning path provided. At the beginning of the session, all participants were informed of the characteristics of the research and that their participation in the study was voluntary.

The participants had access to the system by using a group code that permitted them to have direct access to the adaptive Kodetu. To avoid a registration system, which is particularly sensitive for minors, who in many cases cannot yet have a personal email address, we defined a hidden identification system that assigns a unique random alphanumeric code to each participant. In addition, no personal identification data were requested and no IP addresses were stored, so privacy and

anonymity were guaranteed. The system asked for demographic data (age, gender, and educational cycle), name of the school, information on whether they have previous knowledge of programming (yes/no), and love for technology (numerical value from 1 -low- to 10 -high-).

Following the completion of the questionnaire, the teacher did a short presentation of the Kodetu game, the basic concepts, and explained the categorization of the challenges based on the blocks provided to them. Once the presentation was finished, the participants entered the game and started solving the three initial challenges. The 4th challenge was different for each participant and depended on the performance at the 3 initial challenges; if the participant failed to solve one or more of the 3 challenges, the next challenge was one from Range 1 and less difficult than the one they failed. If they succeeded in all challenges, then the 4th challenge was from Range 3 and more difficult than the previous 3 challenges. The participants continued playing for the remaining time, creating a learning path adjusted to their performance.

4.2. Results

In this section, we present the results obtained from the dataset created with the learners' interactions when using the Kodetu adaptive block-based maze game. The design of the research also helped us test and validate the difficulty function for block-based maze challenges. Our aims were to investigate the changes in learners' performance and specific data metrics when using adaptive block-based maze games, and to compare the effectiveness of the adaptive Kodetu system with the non-adaptive Kodetu. Specifically, we seeked to answer the following research questions:

4. How does the version (adaptive/non-adaptive) of the block-based maze game affect learners' achievements?

5. How does the version (adaptive/non-adaptive) of the block-based maze game affect the learners' performance in a maze-based programming challenge?

Following the data cleansing procedure outlined in section 3.3.2, we proceeded to analyze the data in order to answer the research questions.

In many games and learning activities, the organization of content according to learners' performance and knowledge, is a fundamental aspect (Hooshyar et al., 2018;

Soflano et al., 2015). The difficulty adjusted to the ability of the learner and the progressive increase of this difficulty facilitates learning and maintains interest (Avi Shena et al., 2019). The average number of the adaptive Kodetu challenges played by each participant is 9.71, the three initial common challenges for all participants and the rest depending on their performance. In the experiments performed with the non-adaptive Kodetu, only 30% of the initial participants reached the 7th (last) challenge of the sequence. In both experiments, the available time to perform the experiments was 60 minutes. In the 60 minutes available to perform the experiment, 65% of the participants reached challenges from the last three ranges of difficulty values (60.28≤difficulty≤84.7135), and 57% of them solved the challenges successfully. The corresponding results of the experiments with the non-adaptive Kodetu with same age participants (Section 3.4), showed that of the 62% of the participants that played with challenges with difficulty higher than 60.28, 40% managed to solve them successfully.

Table 4.2 shows the mean and standard deviation (between brackets) of the data metrics **success time** (time required to solve a challenge), **attempts per user** (average number of attempts that each participant needed to solve the challenge), **interactions per user** (blocks added or deleted in the workspace), and the **difficulty** of the challenges in each category. Data were obtained by the experiments of Study 2 (Section 3.4) for the non-adaptive Kodetu and by the experiments performed with the adaptive Kodetu. The participants in both experiments were 9 to 11 years old. The challenges were separated and compared by category (sequential, loops, conditionals). Values in bold indicate the best result in each category and data metric.

Challenge Category	**Non-adaptive Kodetu (Study 2)**				**Adaptive Kodetu**			
	Success time	Attempts	Interactions	Difficulty	Success time	Attempts	Interactions	Difficulty
Sequential	175.78 (120.222)	**2.98 (1.275)**	80.71 (42.537)	**18.89 (11.627)**	**99.87 (76.284)**	3.27 (2.300)	**58.81 (41.697)**	17.78 (8.329)
Loops	**88.94 (119.112)**	3.10 (3.567)	**51.50 (71.014)**	21.27 (8.915)	175.86 (144.784)	**3.01 (3.047)**	87.14 (74.633)	**31.01 (6.782)**
Conditionals	402.50 (269.377)	17.72 (14.368)	144.47 (48.976)	61.91 (9.981)	**190.70 (317.882)**	**13.61 (12.257)**	**126.33 (142.691)**	**62.65 (9.727)**

Table 4.2 Mean and Standard Deviation of data metrics Success time, Attempts and Interactions and the Difficulty obtained by non-adaptive Kodetu and adaptive Kodetu.

Observing Table 4.2, we can point out the following results:

1. A Pearson correlation coefficient was computed to assess the relationship between the difficulty of the challenges and the data metrics success time, attempts and interactions of the adaptive Kodetu. There was a positive correlation between the difficulty and the interactions, $r(96) = 0.358$, $p = 0.000$. Also, positive correlation was observed between the difficulty and the number of attempts, $r(96) = 0.430$, $p = 0.000$. No relationship was found between the difficulty and the success time ($p = 0.488$).
2. We performed the non-parametric Mann-Whitney U Test to determine if the difference between the mean difficulty of each challenge category is statistically significant in the two versions of Kodetu. For the sequential

challenge category, the test specified that there is no statistically significant difference in the difficulty of the challenges of the adaptive and the non-adaptive version given that the p-value returned by this test is 0.20. The Mann-Whitney test for the loops challenge category indicated that the challenges were statistically significantly more difficult for the adaptive Kodetu (M=32.28) than for the non-adaptive version (M=19.45), U=15.5, z=-2.028, p-value=0.043. For the last challenge category (conditionals), the difficulty of the challenges in the adaptive version was not statistically significantly different from the one of the non-adaptive version, given the p-value=0.902.

3. The adaptive Kodetu gave better results than the non-adaptive regarding the sequential challenge category. Participants playing the adaptive version, needed less time to succeed in a challenge, and in the meantime, they interacted less with the platform. Taking into account the "three As" iterative process of CT based on the three stages (a) abstraction (problem formulation), (b) automation (solution expression), and (c) analysis (solution execution and evaluation) (Repenning et al., 2016), we can deduct that they can quickly and effectively formulate the problem, express the solution easier and execute it.
4. Similarly, in the conditionals category, the data metrics showed that the participants not only needed less time, and fewer interactions to succeed in a challenge, they also needed fewer attempts to solve it. It is interesting to point out that despite the fact that the challenges at the adaptive Kodetu are more difficult than the ones of the non-adaptive, we observed better results in the data metrics, indicating that by playing the adaptive version, learners are able to solve the more complicated problems in a more effective way.
5. On the contrary, in the loops challenge category, we observed better results in the non-adaptive Kodetu. Participants needed less time to succeed as well as fewer interactions with the game's interface when they were not playing with the adaptive version. Nevertheless, loops challenges at the adaptive Kodetu are more difficult than the ones at the non-adaptive, thus the better results in the data metrics do not mean necessarily that the non-adaptive Kodetu is more effective than the adaptive one in the development of the CT skills associated with the loops challenge category.

Table 4.3 is similar to Table 4.2, showing the mean and standard deviation of the data metrics mentioned above. The difference lies in the age difference of the learners. This time we compared the data metrics of the adaptive Kodetu (participants' age 9 to 11 years old) with the results of the experiments with the non-adaptive Kodetu where the participants were 15 to 16 years old (Section 3.5).

In the sequential and the conditionals challenge categories the data metrics gave better results at the adaptive Kodetu despite the age difference. Similar to the results of the comparison between the adaptive and non-adaptive Kodetu with same age participants, in the loops category, the participants in the non-adaptive system performed better. This result is expected not only because the challenges are easier, but because the participants are older too.

Challenge Category	**Non-adaptive Kodetu (Study 3), 15-16 y.o.**				**Adaptive Kodetu, 9-11, y.o.**			
	Success time	Attempts	Interactions	Difficulty	Success time	Attempts	Interactions	Difficulty
Sequential	104.67 (68.062)	**1.95 (0.805)**	70 (37.873)	**18.89 (11.627)**	**99.87 (76.284)**	3.27 (2.300)	**58.81 (41.697)**	17.78 (8.329)
Loops	**51.94 (58.943)**	**2.25 (1.893)**	**33.50 (41.348)**	21.27 (8.915)	175.86 (144.784)	3.01 (3.047)	87.14 (74.633)	**31.01 (6.782)**
Conditionals	315.54 (267.165)	17.24 (14.590)	154.71 (83.969)	61.91 (9.981)	**190.70 (317.882)**	**13.61 (12.257)**	**126.33 (142.691)**	**62.65 (9.727)**

Table 4.3 Mean and Standard Deviation of data metrics Success time, Attempts and Interactions and the Difficulty obtained by Kodetu normal version (15-16 y.o.) and adaptive Kodetu (9-11 y.o.).

Lastly, we used the FRBS presented in Chapter 3 to calculate the performance on the adaptive Kodetu challenges played. All the system's characteristics were kept identical to be able to compare the performance in both systems. The input variables are the average success time, the number of attempts per participant as well as the number of interactions per participant.

In Figure 4.2 Plot of membership functions-FRBS adaptive Kodetu. we present the MFs that differ from the ones presented in Chapter 3. As explained in Section 3.6, for each of the linguistic labels associated with the three input variables, there are four MFs: low, medium, high, and any, where the linguistic label "any" encompasses all values of the variable and is used when the associated variable is not significant in a fuzzy rule and changes in its value should not be taken into account when determining performance.

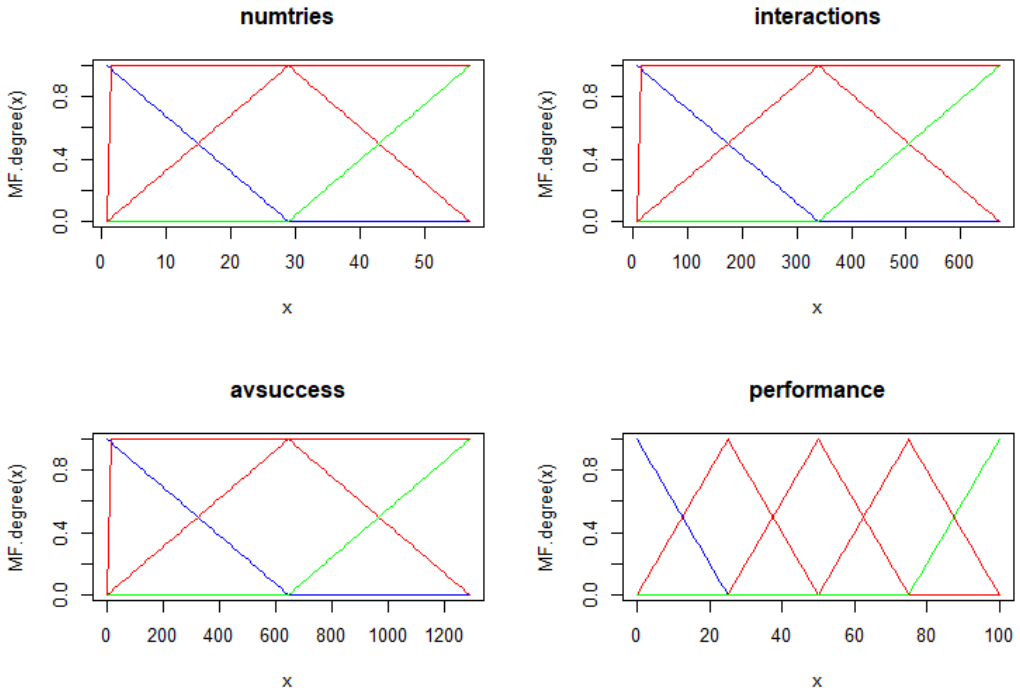

Figure 4.2 Plot of membership functions-FRBS adaptive Kodetu.

The output is the participant's performance in each challenge. With the FRBS, we obtain the value of performance in each challenge of the adaptive Kodetu as an output variable given the values of the input variables, a number from 0 to 100, where 0 represents the poorest performance and 100 the best. The results are presented in Table 4.4.

Challenge Category	Performance values per challenge separated by Range												
Sequential													
	Range 1	99.83	90.30	99.66	99.35	93.52	91.98	93.08	96.96	97.76			
	Range 2	89.51	94.02	91.23	97.08	93.99	92.68	93.79	88.29				
	Range 3	86.23	79.70	58.20	89.92	88.90	92.02	92.88	81.94	84.80			
	Range 4	87.34	97.84	97.65	87.37	88.45							
Loops													
	Range 5	46.66	78.10	55.84	99.52	86.97	86.46						
	Range 6	78.13	100	97.50	57.27	92.90							
	Range 7	89.38	91.51	75.62	90.57	73.84							
	Range 8	95.48	97.93	91.27	87.68	87.49	95.82						
Conditionals													
	Range 9	0.10	70.43	79.42	75	25	41.98	67.55	57.61	39.60	86.63	86.35	94.99	82.54
	Range 1	72.82	72.17	94.77	88.64	39.76	81.81	96.92	94.43	75	62.20	75	94.89	51.53
	Range 1	73.07	80.04	91.49	27.67	97.16	73.02	29.04						
	Range 1	92.05	46.86	72.35	99.81	84.52	92.72	90.98	99.22	85.25	95.98			

Table 4.4 Crisp output of frbs - performance values, Adaptive Kodetu.

To assess whether the differences in performance observed in Study 2, Study 3 and the adaptive version are significant or not, we made use of non-parametric statistical tests. We performed the Mann-Whitney U test to compare the distributions of performance values obtained from the independent participants' groups of Study 2 and the adaptive version. The test indicated that the performance of the participants was statistically significantly higher for the adaptive Kodetu (M=74.61) than for the non-adaptive (M=57.81), U=1525, z=-2.272, p-value=0.023. Similarly, we performed the Mann-Whitney U test for Study 3 and adaptive version performance values. The test specified that there was no statistically significant difference in the performance of the participants in the two groups given that the p-value returned by this test is 0.09.

In regards to the difficulty function, performing Pearson's correlation test we found that participants' performance was negatively correlated with the difficulty of the challenges with $r(96) = -0.318$, $p = 0.000$, meaning that the more difficult the challenges the worse the performance.

4.3. Discussion

Throughout the last 2 chapters we have exposed the methodology of our research. In total, we performed 4 experiments with more than 380 participants from 9 to 16 years old and analyzed sessions that included more than 500.000 interactions. Starting from the development of a large investigation containing 3 studies, we have detailed each step of the research, the creation of new challenges that the game consists of to investigate its specific characteristics, the experimentation process, the data collection, and analysis, as well as an extensive presentation of the research results. We have also described in detail the development of the adaptive game, the science behind it, the experimentation process and we have presented the influence of the game version (adaptive/non-adaptive) on learner's performance.

The effectiveness of the adaptive system is an important indicator of the validity of our estimation of the difficulty of block-based maze games. The new dataset obtained offered us practical information, by performing statistical tests, to further validate the difficulty function. After performing Pearson correlation tests, the results indicated that the more difficult the challenges, the more interactions the learners have with the platform and the more attempts to solve the maze are needed, which is the expected result based on how the difficulty function was calculated. Similar results were found regarding the performance, it is negatively correlated with the difficulty of the challenges, i.e. the more difficult the challenges the lowest the performance.

The results presented in this chapter help us understand how adaptive learning paths can affect learners' performance and can also serve as a consideration for designing better and more effective CT tools. We reviewed the conclusions associated with our research objectives and we present them below.

Research question 4: How does the version (adaptive/non-adaptive) of the block-based maze game affect learners' achievements?

Analyzing the data gathered from the experimentation with the adaptive Kodetu, we found that participants played almost 10 challenges in the 60 minutes provided, whereas, in the non-adaptive Kodetu, 30% of the participants reached the 7th challenge (last challenge of the sequence). Consequently, the results suggest that the

participants in the adaptive Kodetu experiment were more engaged with the game, and the learning path provided to them helped them progress quicker. This is consistent with the results of previous research that adapting the challenges provided to the learners can lead to bigger engagement (Andersen, 2012; Lomas et al., 2017; Rubio-Tamayo et al., 2017), motivation (Kiili et al., 2012; Sampayo-Vargas et al., 2013), thus better skills acquisition (All et al., 2015); the more experience is gained in the educational game (more challenges played), the more their abilities are increased (Ali & Sah, 2017).

It is noteworthy that the majority of the challenges played (65%) belong to the 3 most difficult challenge ranges (60.28≤difficulty≤84.7135), while in the non-adaptive the corresponding number is 62%; therefore, the increase in the number of challenges played is not caused due to the fact that participants played easier challenges. Furthermore, the success rate for these challenges to the adaptive Kodetu was 57% and to the non-adaptive was 40%. Interacting with more difficult challenges, participants were exposed to more difficult programming concepts and were able to train to more demanding problems, enhancing their CT development (D. Barr et al., 2011). Succeeding in more challenges, self-confidence is increasing and learners feel more confident to go to the next step and handle more complicated tasks (Cherney, 2008; Markowitz, 2004).

Research Question 5: How does the version (adaptive/non-adaptive) of the block-based maze game affect the learners' performance in a maze-based programming challenge?

Looking deeper into specific data metrics, we found out that the adaptive version has a significant effect on the success time, the interactions, and attempts on a block-based maze challenge. We compared the results from the experiments with the adaptive Kodetu where the participants were 9 to 11 years old, with the results from Study 2 (Section 3.4) and Study 3 (Section 3.5), with participants 9 to 11 and 15 to 16 years old respectively. In particular, observing the data metrics for each challenge category, our results showed that regarding the sequential and conditionals category, despite the fact that the challenges were of similar difficulty (results from the Mann-Witney U test showed a non-statistically significant difference), the participants performed significantly better in the data metrics. Specifically, participants in the non-adaptive Kodetu experiment of the same age participants needed almost twice as

long time to solve the challenges and with more interactions, despite the fact that the number of attempts to solve the challenge remained similar.

Regarding the challenges that include conditionals blocks, thus complicated programming concepts are introduced, research has shown that it seems challenging for learners to make efficient conditionals statements (Kwon & Cheon, 2019). Our results showed that, when playing with the adaptive system, despite the age of the participants, they succeeded in less time, with fewer interactions with the game and attempts. These results showed that it is more effective to use an adaptive block-based maze game compared to a non-adaptive, learners achieve better results in less time and with less effort, allowing them to continue playing and developing CT.

It is important to discuss the results regarding the loops challenge category in both comparisons, with same age participants in both experiments and with older participants interacting with the non-adaptive Kodetu. It is the only challenge category where the participants interacting with the non-adaptive Kodetu succeeded in less time and their interactions with the platform were fewer. The Mann-Whitney U test showed that the difference in the difficulty of the challenges were statistically significant (the adaptive Kodetu challenges were significantly more difficult). Although in the second comparison, with the older participants in the non-adaptive Kodetu, the better success time and fewer interactions are the expected result and are consistent with previous research, due to the age difference and the less difficult challenges (Navarro et al., 2015), further investigation is needed when comparing the data metrics of the experiments with same age participants. In these experiments, where there is no age difference between the participants, we cannot safely conclude that the better results are due to the fact that the non-adaptive Kodetu is more efficient for teaching the loop related CT concepts. The difference in the success time and interactions can be caused because the challenges in the adaptive system are more difficult, in line with previous studies (Lynch et al., 2019; Sampayo-Vargas et al., 2013). Connecting that with the fact that conditional challenges integrate loops, and in the conditional category we got better data metrics in the adaptive Kodetu, we believe that there should be conducted more experiments in the future, comparing the adaptive and non-adaptive block-based maze games, focusing on the loops challenge category.

Calculating the performance for each level with the use of the FRBS we showed that learners of the same age perform better in the adaptive block-based

maze game than in the non-adaptive one. Statistical tests showed that the difference in the performance is significant, meaning that learning is acquired more efficiently when we use the adaptive version. This result is in accordance with findings reported by Hooshyar et al. (2021) and van Oostendorp et al. (2014). The interest lies in the fact that the statistical tests did not show a significant difference between the performance of 15-16 years old participants and the 9-11 years old. Plenty of research indicates that the older the learners the better they perform in block-based maze games (Eguiluz et al., 2017; Guenaga et al., 2021; S.-Y. Wu, 2018) and that it is essential to provide differentiated tools and challenges for different ages (Alves et al., 2019). As the results show, we can overcome these challenges by offering adaptive games that provide personalized learning paths based on learners' performance.

4.4. Summary

It is important to know and comment on the limitations of this part of our research. Although the interventions with the learners are supervised, it cannot be avoided that some learners interact with each other and this cooperation very likely influences the results. We minimized these events by giving special attention to this issue, without preventing learners from enjoying playing. Furthermore, the log analysis can be enriched and complemented with more assessment techniques such as interviews and questionnaires, which can give us a broader view of the skills acquired.

Nevertheless, the results presented in this chapter show that adaptive tools like the adaptive Kodetu, play an important role in assisting and developing CT in primary and secondary education. The conclusions help us validate our estimation of the difficulty function and provide educational stakeholders with important information and a tool to promote adaptive learning. In the next and final chapter of this investigation, we will summarize all the conclusions and we will state the future lines of work.

Chapter

5

Conclusions

This thesis aimed to investigate the development of CT through an educational game of adaptive difficulty based on the learners' performance. We presented the work conducted to measure the difficulty of block-based maze programming challenges based on the maze and coding characteristics. Based on the difficulty measurement, we presented the investigation performed to demonstrate how an adaptive block-based maze game can be developed with which we can obtain better CT development than with a non-adaptive one. For this purpose, studies about CT, game-based learning, adaptive games, and their difficulty definitions have been reviewed and described. During this process, we identified the gaps regarding the development of adaptive block-based maze games and the measurement of the difficulty of block-based maze programming challenges. The necessity of providing learning paths of adaptive difficulty was established. This led us to perform four experiments with more than 380 participants from 9 to 16 years old and analyzed sessions that included more than 500,000 interactions. In particular, three studies were performed with 326 participants 9 to 16 years old, to measure the difficulty of block-based maze games and then create a block-based maze game that will provide challenges of adaptive difficulty to the learners based on their performance, that was tested during experimentation with 60 participants 9 to 11 years old. In this chapter, we present the conclusions of this research, summarizing the process carried out, indicating its limitations, summarizing the results obtained, outlining some of the possible applications, and pointing out the most important future lines of research that we propose.

5.1. Summary of the investigation process

The research conducted was divided into six closely linked milestones, beginning with an exploratory approach during the structuring of the study and ending with a block-based maze game that offers adaptive sequences of challenges. The following is a description of the work carried out at each milestone.

Milestone 1. Establishment of the research hypothesis.

The increasing use of technology in our daily lives has heightened the need for computer science professionals, and CT is a very important skill not only for them but for most of the citizens of the 21st century. The learning paths provided in block-based maze games are usually fixed without taking into account the learners' abilities (Park et al., 2019), and there is no specific definition of the difficulty of block-based challenges that would help adapt the level of difficulty of the learning path offered based on the learners' performance. In this thesis, we proposed to improve the development of CT by personalizing the learning path dynamically through programming challenges in a block-based maze game.

Milestone 2. Review of the state of the art.

In this thesis, we reviewed studies on the definition of CT, how to develop and promote CT, what tools are available, and the characteristics of these tools that make them ideal for that purpose. We investigated how learning to program through games is beneficial for learners and examined the variety of tools and games that were created so that early programmers can develop CT. Furthermore, we investigated the different methodological approaches of adaptive games, their use in learning programming, and how they use different difficulty definitions to provide the adaptive features. We examined the different approaches to measure the difficulty of block-based programming games. This literature review made it possible to understand that the difficulty of block-based maze games depends on factors that go beyond the percentage of failures and the complexity of maze hallways, to determining the important characteristics that would make them efficiently adaptive. That knowledge was used during the whole experimentation process of this thesis.

Milestone 3. Experimentation to measure the difficulty of block-based maze games.

Having clarified the importance of calculating the difficulty in order to create the adaptive block-based maze game and having identified what is missing to calculate the difficulty, the design and implementation of the experiments began. Initially, the possible parameters of this type of game that affect the performance of learners and make them more difficult were identified. Based on these parameters, the maze challenges with which the participants played, were created. Then, we designed the sequences from challenges, i.e., the learning path of the participants, taking into account various design principles that helped in the best and most objective collection of data for analysis. Finally, we introduced the sequences in the block-based maze game Kodetu, which was used to carry out this research. By completing the design of the experiments, three studies were conducted with a total of 326 participants from 9 to 16 years old. After each study, data analysis, statistical tests, and interpretation of the results were performed. These results, as well as the limitations, served as design rules for the next study.

Milestone 4. Calculation of the difficulty function.

The data obtained from the experiments were used to measure the difficulty of block-based maze games. Professionals and experts in the field of education can characterize the performance at each challenge empirically and based on various metrics, such as success time, number of interactions, and attempts. Our aim was to use the knowledge and experience of experts in a generalized algorithm so that performance can be calculated in the different challenges. Various algorithms were tested to find the most appropriate one to measure the performance of the learners and, therefore, the difficulty of the challenges. We ended up using a FRBS, which uses various rules written by experts, to obtain a value of performance (a number between 0 and 100) in each challenge using various data metrics obtained from the three studies previously mentioned. The value of performance for each challenge was used to perform linear regression and determine if and how much the coding and maze characteristics of the block-based maze game influence the difficulty of a challenge. Based on the results of the linear regression and the theoretical background studied in the bibliography, we defined the function of the difficulty estimation and an estimation of how much time on average is needed to succeed in a challenge.

Milestone 5. Design and development of the adaptive Kodetu.

The calculation of the difficulty function of the block-based maze game allowed us to proceed with the development of the adaptive system. The basic principles of CAT systems were used for the development. Initially, 110 Kodetu challenges were designed and introduced to the system based on various features necessary for the research. The challenges were divided into three categories (sequential, loops, and conditionals), based on the necessary blocks to solve them. For all the challenges, their difficulty was calculated, with the easiest having a difficulty of 0.259 and the most difficult at 81.115, as well as the estimation of the average success time. Challenges of every category were separated into four ranges based on their difficulty value, so the adaptive Kodetu was functioning as follows: if the learner was able to solve the challenge in less time than the average success time estimation, a challenge chosen from the two ranges above is the next one provided; if the challenge is solved in more time than the average success time, then an easier challenge is provided of one difficulty range less. Having designed how the system will work, we incorporated our algorithm into Kodetu and developed the adaptive Kodetu.

Milestone 6. Experimentation and validation of adaptive Kodetu.

New experiments were designed and performed in order to test the adaptive Kodetu and the difficulty function. Sixty students from 9 to 11 years old participated in the experiment, and they played with the adaptive Kodetu, creating their learning path based on their performance. The dataset obtained by this experiment was analyzed; specific data metrics were calculated and compared with the relative values of the non-adaptive Kodetu. With these results, we were able to validate our hypothesis.

5.2. Research results

This thesis aimed to investigate the effect of adaptive learning paths on the development of CT. In the first part, presented in Chapter 3, we saw that the results help to understand the characteristics of block-based maze programming challenges that affect the learners' performance and the difficulty of the challenges. We were able to measure the difficulty function of block-based maze programming challenges. The development of the adaptive Kodetu and the comparison between the adaptive and the non-adaptive version indicated that better results are achieved when learning occurs with block-based maze games that adapt the difficulty of the challenges based

on the learners' performance. Considering this, we review the conclusions associated with our research questions:

Research question 1: How do maze characteristics (width, height, total number of steps in the maze, optimal path, maze loops, turns, and numbers of x-crosses and t-crosses) affect the performance in a maze-based programming challenge?

After performing three studies with 9- to 16-year-old K-12 students, we found that there are specific maze characteristics in a block-based maze programming challenge that, depending on their value, are affecting the performance of the learners. These characteristics are 1) the turns in the optimal path, i.e., whether to solve the maze, the learner has to use the turn left and turn right blocks and 2) the number of total steps in a maze. When performing experiments with challenges that contain maze loops (Study 1), the performance was very low, and there was a significant failure rate in the challenges. However, statistical tests revealed that the high failure rate and poor performance were not due to the maze loops (no statistically significant difference between success and maze loops [$F(3,31) = 0.705$, $p = 0.556$]). Therefore, this characteristic should not be considered a factor that can increase the difficulty of a challenge.

Research question 2: How do coding limitations (blocks provided and block limit) affect the performance in a maze-based programming challenge?

Coding limitations in a maze-based programming challenge were found to significantly affect the performance of the learners. The three types of blocks that are provided to the learners are directly connected with specific concepts: 1) the sequential blocks (move forward, turn left, turn right) that can execute linearly the steps of an algorithm, 2) the loop block (repeat until) which permits a series of steps to be executed repeatedly, and 3) the conditionals blocks (if-then, if-then-else) where the learners need to find the correct conditions for the steps to be executed. The loop and conditionals block categories are more complicated and require more mental effort. Therefore, it was found that the performance was affected by the challenges where, in order to be solved, blocks from these two categories had to be used (statistically significant effect of the blocks available [$F(2,39) = 20.032$, p-value $= 0.000$]). As for the block limit characteristic, it is directly related to the block type; by

limiting the number of blocks that the learner can use to solve the challenge, the learners cannot only use sequential solutions to exit the maze; instead, they are forced to use the loop and conditionals blocks (statistically significant effect of the block limit [$F(1,40) = 17.902$, p-value $= 0.000$]).

Research question 3: How can the difficulty be measured in block-based maze games?

Having answered the two previous research questions, we presented our difficulty function in Section 3.6.2:

$dif = -120.848 + 0.985 * num\_steps + 8.289 * turns + 4.076 * blocks$ (1).

A challenge is difficult when it involves turns and a significant number of steps, and when the learner has to necessarily use the sequential, loop, and conditional blocks to solve the challenge.

Furthermore, considering that limiting the time to solve a challenge consists of an added difficulty, we presented a second difficulty function for challenges that contain time limitations:

$$diftime = dif * \frac{ASt}{0.8 * time\_limit} \quad (2)$$

where $ASt = 14.303 * num\_steps + 23.891 * blocks$ (3)

After finishing this part of the investigation, we were ready to develop the adaptive Kodetu. The system's design was based on the basic principles of CAT systems. Its main goal was to investigate if having adaptive learning paths affects performance in block-based maze games, resulting in better CT acquisition, and to test and validate the difficulty function for block-based maze challenges. With the results of this study and the discussion presented in Chapter 4, we comment on the conclusions to our research questions:

Research question 4: How does the version (adaptive/non-adaptive) of the block-based maze game affect learners' achievements?

Our results showed that by playing with the adaptive block-based maze game, learners solve more challenges in the same period of time than with the non-adaptive game. 65% of the learners playing with the adaptive Kodetu reached challenges from the last three ranges of difficulty ($60.28 \leq difficulty \leq 84.7135$), and 57% of them

successfully completed them, where only 40% of the learners participating in the experimentation with the non-adaptive Kodetu, were successful in solving the challenges. The regulation of the difficulty based on the learners' performance provided in the adaptive game led not only to playing more difficult challenges than in the non-adaptive one but also to achieving higher success rates in these difficult challenges.

Research question 5: How does the version (adaptive/non-adaptive) of the block-based maze game affect the learners' performance in a maze-based programming challenge?

Specific data metrics that are indicative of how well the learners are performing showed that the version of the system is affecting the success in the challenges. For the challenges that belong to the category of sequential and conditionals, when learners played with the adaptive system, they needed less time to succeed in a challenge and performed fewer interactions with the platform and similar attempts, meaning that the algorithm was clear to them. They did not need to add and delete blocks that were not necessary for the solution of the challenge. In addition, when comparing the data metrics from the experimentation involving the 9- to 11-year-old participants playing with the adaptive version of the games from study 3 to the 15- to 16-year-old participants playing with the non-adaptive version, we contrasted that playing with the adaptive version can result in better values of data metrics, i.e., they succeed in less time, with fewer failed attempts, and performing fewer interactions with the platform.

The FRBS presented in Chapter 3 allowed us to calculate the performance in different block-based maze challenges. Our results indicated that the performance is significantly better in the adaptive game (learners' performance was statistically significantly higher for the adaptive Kodetu (M=74.61) than for the non-adaptive (M=57.81), U=1525, z=-2.272, p-value=0.023). Even when comparing the data obtained from experimentation with the adaptive Kodetu and 9- to 11-year old participants to the performance of 15- to 16-year-old participants playing with the non-adaptive system the results showed that despite the age difference, they performed similar in both versions (not statistically significant difference, p-value=0.09).

5.2.1. Research implications and publications

From the research process developed and described, the necessary evidence is provided to answer the hypothesis. We found that adaptive learning paths based on the learners' performance on block-based maze games led to better success rates in difficult programming challenges and higher performance in less time and with less effort. Hence, considering previous research presented in Chapter 2 concerning the effectiveness of block-based programming in CT development and the results from our investigation, we consider that the hypothesis of this thesis is valid:

Learners can improve their performance in a set of block-based maze challenges for the development of computational thinking through a Computerized Adaptive Testing system that estimates the difficulty of each challenge according to maze and code characteristics.

Specifically, when creating programming challenges in block-based maze games, we recommend taking into account the maze characteristics: the turns in the optimal path, the number of total steps in the maze, and the coding characteristics and block limitation of blocks provided. As mentioned in previous studies (J. Kim, 2006; Li & Belkin, 2008; J. Liu et al., 2011), learners can perceive a task as being difficult when their performance is low, which leads to behaviors that have no learning benefits, thus having identified the characteristics that affect their performance is of high value.

The above contribution was described and presented at the following conference:

Title: What makes a maze-based programming challenge difficult?
Authors: Ioanna Kanellopoulou, Pablo Garaizar, Mariluz Guenaga.
Conference: Learning Analytics Summer Institute Spain 2021.
Proceedings URL: http://ceur-ws.org/Vol-3029/#paper02

We are considering the difficulty function as one of the most important contributions of this thesis. Given the lack of a difficulty function for block-based maze games, this research provides the first estimation of difficulty that depends on the maze and coding characteristics of block-based programming challenges. Nowadays, block-based games are being increasingly introduced to the curriculums of K-12 education,

highlighting the importance of the CT development in primary and secondary education. Having a function that calculates the difficulty of maze programming challenges and that can be used when designing the learning paths will enable teachers to design personalized learning paths with increasing difficulty, even when they have no access to an adaptive system, helping their students to acquire CT in a more efficient and enjoyable manner.

The investigation process to obtain the difficulty function was published in the peer-reviewed journal:

Title: First Steps Towards Automatically Defining the Difficulty of Maze-Based Programming Challenges.
Authors: Ioanna Kanellopoulou, Pablo Garaizar, Mariluz Guenaga.
Journal: IEEE Access (Impact Factor = 3.367 -> Q1).
Status: Published. Vol. 9, pp. 64211-64223, 2021.

The development of the adaptive Kodetu and the promising results of the experiment performed is an important step toward more efficient adaptive block-based maze games. The adaptive Kodetu is still online and offered free, available in English, Spanish, Basque, and Hebrew. Educational stakeholders that organize interventions with the aim of improving CT can use it for free and offer personalized learning to as many learners as they want. Moreover, the adaptive Kodetu can be personalized, and new challenges can be added and offered, based on the user's needs; thus, it can serve different purposes. For example, researchers can use it to perform experiments and research in various fields of pedagogy and education, and teachers can create their own learning paths that fit their classes' needs. We are planning to publish the investigation process and the results of our research regarding the adaptive Kodetu to a peer-reviewed journal relevant to our topic.

5.3. Limitations and future lines of work

Before explaining the future lines of work of our investigation, we review its most important limitations.

In order to achieve accurate and efficient data collection, we organized all the experiments in a supervised environment, i.e., making organized interventions in student classes of different schools. Achieving this requires preparation that began months in advance and involved contacting schools, applying for and granting permits to conduct experiments, finding the appropriate department based on the purpose of the research, and adjusting the availability of students to limited months of the school year.

The above led to the following limitation: the available sample size. Especially for the experiments performed with the adaptive Kodetu (Chapter 4), the sample size was not ideal (60 participants). Nevertheless, the analysis of the data did not show any abnormalities due to the sample size and we were able to draw safe conclusions, but it is generally accepted that the sample size is important in order to obtain accurate results.

Furthermore, the variety of the sample of participants in our experiments is another possible limitation of this study. All participants attend schools in the Basque country region. Although this can lead to a homogenous sample that helps us perform accurate comparisons between the data of the studies, it is not possible to measure whether the common geographical characteristics have an effect on the data collected and the results.

Another limitation involves the fact that prior research studies that are relevant to the current thesis are limited, especially with the difficulty definition for block-based maze games. Specifically, during the experimentation described in Chapter 3, there was a lack of specific rules on how to design the challenges and the learning paths, so the experimentation was mostly conducted in an empirical way. However, performing the next study after analyzing the data from the previous one, as well as the choice of the tools to proceed to the definition of the difficulty (FRBS and regression analysis), minimized the further effect of this aspect in our research.

There is clearly future work to be done to explore the implementation of block-based maze games in CT learning and skills development via adaptive learning paths. The research in this thesis has addressed some of the fundamental issues with difficulty definition and development of adaptive block-based maze games. These give the direction for further work in these areas. This section provides an overview of some areas of future interest.

We consider this research to be the first step in identifying and defining a proper estimation of difficulty in this type of game. To achieve this, we identified ten variables that may affect the difficulty and designed our experiments and research to investigate whether and how much they affect the performance and difficulty of Kodetu challenges. Future work should focus on identifying more variables that may affect the difficulty of this type of game, such as the number of blocks of the optimal solution or the initial position of the character. These additional variables should be investigated and added to the difficulty function in order to increase the function's accuracy.

Increasing the difficulty function's accuracy will lead to more accurate learning paths for the adaptive Kodetu. Thus, by calculating and inserting the new values of difficulty of the challenges in the adaptive system, new experiments can be organized, this time without the aforementioned limitations. Having a larger and more diverse sample will lead to the collection of a valuable dataset, fundamental to foster the development of knowledge and a powerful and important tool for researchers. This dataset, along with our research so far, can assist not only in exploring in greater depth the inconclusive results we discovered regarding the performance in the loops challenge category but, more importantly, in assessing the adaptive system, defining new design rules, and recommending the most effective design of a block-based maze game with adaptive learning paths.

Nevertheless, the major future work that derives from this thesis is the dynamic generation of challenges based on the learners' performance. Our research results encourage us to design new experiments and explore the effects of providing automatically generated personalized learning paths for the acquisition of CT. Focus should be given to the way to automatically create adapted learning paths to improve the development of CT. Procedural Content Generation techniques and Genetic algorithms can be used in order to automatically generate the challenges, with the difficulty function serving as the fitness function.

The COVID-19 pandemic imposed the fast and efficient total implementation of digital learning at all levels of education. We are already in the aftermath of the pandemic, and many of these digital learning practices used in the last two years have been permanently integrated into education. Thus, creating adaptive learning paths and, in general, adaptive block-based games to develop CT and computer science competencies is only the beginning of their implementation in modern education and

teaching. Teachers, researchers, and general education stakeholders are constantly making efforts to advance learning acquisition of the important 21st-century skills, using the state-of-the-art tools and techniques that are continuously developed and widely provided. Research like the one conducted and presented in this thesis consists of one more step to the progress of providing effective and pioneering educational tools and techniques.

Bibliography

Abdellatif, A. J., McCollum, B., & McMullan, P. (2018). Serious games: Quality characteristics evaluation framework and case study. *2018 IEEE Integrated STEM Education Conference (ISEC)*, 112–119. https://doi.org/10.1109/ISECon.2018.8340460

Abrahamson, D. (2006). The Shape of Things to Come: The Computational Pictograph as a Bridge From Combinatorial Space to Outcome Distribution. *International Journal of Computers for Mathematical Learning*, *11*(1), 137–146. https://doi.org/10.1007/s10758-006-9102-y

Adcock, A. B., Watson, G. S., Morrison, G. R., & Belfore, L. A. (2011). Effective knowledge development in game-based learning environments: Considering research in cognitive processes and simulation design. In *Gaming and Simulations: Concepts, Methodologies, Tools and Applications* (pp. 409–425). IGI Global.

Adcock, A., & Van Eck, R. (2012). Adaptive Game-Based Learning. In N. M. Seel (Ed.), *Encyclopedia of the Sciences of Learning* (pp. 106–110). Springer US. https://doi.org/10.1007/978-1-4419-1428-6_4

Aho, A. V. (2012). Computation and Computational Thinking. *The Computer Journal*, *55*(7), 832–835. https://doi.org/10.1093/comjnl/bxs074

Ali, A., & Sah, M. (2017). Adaptive game-based e-leaming using semantic web technologies. *2017 International Conference on Open Source Systems & Technologies (ICOSST)*, 15–23.

All, A., Castellar, E. P. N., & Van Looy, J. (2015). Towards a conceptual framework for assessing the effectiveness of digital game-based learning. *Computers & Education*, *88*, 29–37.

Alsawaier, R. S. (2018). The effect of gamification on motivation and engagement. *The International Journal of Information and Learning Technology*.

Alvarez, A., & Scott, T. A. (2010). Using student surveys in determining the difficulty of programming assignments. *Journal of Computing Sciences in Colleges*, *26*(2), 157–163.

Alves, D. C., Wangenheim, N. V., Hauck, C. R., & Jean. (2019). Approaches to Assess Computational Thinking Competences Based on Code Analysis in K-12 Education: A Systematic Mapping Study. *Informatics in Education*, *18*(1), 17–39.

Andersen, E. (2012). Optimizing adaptivity in educational games. *Proceedings of the International Conference on the Foundations of Digital Games*, 279–281.

Anderson, J. R., Boyle, C. F., & Reiser, B. J. (1985). Intelligent tutoring systems. *Science*, *228*(4698), 456–462.

Angeli, C. (2022). The effects of scaffolded programming scripts on pre-service teachers' computational thinking: Developing algorithmic thinking through programming robots. International Journal of Child-Computer Interaction, 31, 100329. https://doi.org/10.1016/j.ijcci.2021.100329

Aponte, M.-V., Levieux, G., & Natkin, S. (2011). Measuring the level of difficulty in single player video games. *Entertainment Computing*, *2*(4), 205–213. https://doi.org/10.1016/j.entcom.2011.04.001

Arnheim, R. (1997). *Visual thinking*. Univ of California Press.

Avi Shena, B. S., Sitohang, B., & Rukmono, S. A. (2019). Application of Dynamic Difficulty Adjustment on Evidence-centered Design Framework for Game Based Learning. *2019 International Conference on Data and Software Engineering (ICoDSE)*, 1–6. https://doi.org/10.1109/ICoDSE48700.2019.9092725

Azevedo, R., Cromley, J. G., Moos, D. C., Greene, J. A., & Winters, F. I. (2011). Adaptive content and process scaffolding: A key to facilitating students' self-regulated learning with hypermedia. *Psychological Test and Assessment Modeling*, *53*(1), 106.

Barab, S. A., Ingram-Goble, A., & Warren, S. (2009). Conceptual play spaces. In *Handbook of research on effective electronic gaming in education* (pp. 989–1009). IGI Global.

Barbosa, L. L. da S., & Maltempi, M. V. (2019). Recognizing Possibilities of Computational Thinking When Teaching First-degree Equations: A Classroom Case. *Proceedings of FabLearn 2019*, 57–64. https://doi.org/10.1145/3311890.3311898

Barnes, T., Richter, H., Powell, E., Chaffin, A., & Godwin, A. (2007). Game2Learn: Building CS1 learning games for retention. *Proceedings of the 12th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education*, 121–125.

Barr, D., Harrison, J., & Conery, L. (2011). Computational Thinking: A Digital Age Skill for Everyone. *Learning & Leading with Technology*, *38*(6), 20–23.

Barr, V., & Stephenson, C. (2011). Bringing Computational Thinking to K-12: What is Involved and What is the Role of the Computer Science Education Community? *ACM Inroads*, *2*(1), 48–54. https://doi.org/10.1145/1929887.1929905

Bashir, G. Md. M., & Hoque, A. S. Md. L. (2016). An effective learning and teaching model for programming languages. *Journal of Computers in Education*, *3*(4), 413–437. https://doi.org/10.1007/s40692-016-0073-2

Bau, D., Gray, J., Kelleher, C., Sheldon, J., & Turbak, F. (2017). Learnable programming: Blocks and beyond. *Communications of the ACM*, *60*(6), 72–80.

Begel, A., & Klopfer, E. (2007). Starlogo TNG: An introduction to game development. *Journal of E-Learning*, *53*(2007), 146.

Bennedsen, J., & Caspersen, M. E. (2008). Exposing the Programming Process. In J. Bennedsen, M. E. Caspersen, & M. Kölling (Eds.), *Reflections on the Teaching of Programming: Methods and Implementations* (pp. 6–16). Springer. https://doi.org/10.1007/978-3-540-77934-6_2

Bergeron, B. (2005). *Developing serious games (game development series)*. Charles River Media, Inc.

Bonar, J. G., & Liffick, B. W. (1987). *A Visual Programming Language for Novices.* PITTSBURGH UNIV PA LEARNING RESEARCH AND DEVELOPMENT CENTER.

Bontchev, B., & Panayotova, R. (2018). Towards Automatic Generation of Serious Maze Games for Education. *Serdica Journal of Computing*, *11*(3–4), 249-278–278.

Bra, P. D., & Calvi, L. (1998). AHA! An open adaptive hypermedia architecture. *New Review of Hypermedia and Multimedia*, *4*(1), 115–139.

Brennan, K., & Resnick, M. (2012). *New frameworks for studying and assessing the development of computational thinking*. 25.

Brusilovsky, P. (1996). Adaptive hypermedia: An attempt to analyze and generalize. *International Conference on Multimedia, Hypermedia, and Virtual Reality*, 288–304.

Brusilovsky, P. (2001). Adaptive hypermedia. *User Modeling and User-Adapted Interaction*, *11*(1), 87–110.

Brusilovsky, P., Eklund, J., & Schwarz, E. (1998). Web-based education for all: A tool for development adaptive courseware. *Computer Networks and ISDN Systems*, *30*(1–7), 291–300.

Buitrago Flórez, F., Casallas, R., Hernández, M., Reyes, A., Restrepo, S., & Danies, G. (2017). Changing a Generation's Way of Thinking: Teaching Computational Thinking Through Programming. *Review of Educational Research*, *87*(4), 834–860. https://doi.org/10.3102/0034654317710096

Chaffin, A., Doran, K., Hicks, D., & Barnes, T. (2009). Experimental evaluation of teaching recursion in a video game. *Proceedings of the 2009 ACM SIGGRAPH Symposium on Video Games*, 79–86.

Chai, T., & Draxler, R. R. (2014). *Root mean square error (RMSE) or mean absolute error (MAE)?* [Preprint]. Numerical Methods. https://doi.org/10.5194/gmdd-7-1525-2014

Chan, S.-W., Looi, C.-K., & Sumintono, B. (2020). Assessing computational thinking abilities among Singapore secondary students: A Rasch model measurement analysis. *Journal of Computers in Education*. https://doi.org/10.1007/s40692-020-00177-2

Chang, H.-H., & Ying, Z. (1996). A global information approach to computerized adaptive testing. *Applied Psychological Measurement*, *20*(3), 213–229.

Chen, G., & Chiu, M. M. (2008). Online discussion processes: Effects of earlier messages' evaluations, knowledge content, social cues and personal information on later messages. *Computers & Education*, *50*(3), 678–692. https://doi.org/10.1016/j.compedu.2006.07.007

Cherney, I. D. (2008). Mom, Let Me Play More Computer Games: They Improve My Mental Rotation Skills. *Sex Roles*, *59*(11), 776–786. https://doi.org/10.1007/s11199-008-9498-z

Chiu, M. M. (2008). Effects of argumentation on group micro-creativity: Statistical discourse analyses of algebra students' collaborative problem solving. *Contemporary Educational Psychology*, *33*(3), 382–402. https://doi.org/10.1016/j.cedpsych.2008.05.001

Chong, E. K. M. (2019). Teaching and Learning Music through the Lens of Computational Thinking. *Proceedings of the International Conference on Art and Arts Education (ICAAE 2018)*. Proceedings of the International Conference on Art and Arts Education (ICAAE 2018), Yogyakarta, Indonesia. https://doi.org/10.2991/icaae-18.2019.1

Çoban, E., & Korkmaz, Ö. (2021). An alternative approach for measuring computational thinking: Performance-based platform. *Thinking Skills and Creativity*, *42*, 100929. https://doi.org/10.1016/j.tsc.2021.100929

Coelho, A., Kato, E., Xavier, J., & Gonçalves, R. (2011). *Serious Game for Introductory Programming*. *6944*, 61–71. https://doi.org/10.1007/978-3-642-23834-5_6

Cole, M. J., Zhang, X., Liu, J., Liu, C., Belkin, N. J., Bierig, R., & Gwizdka, J. (2010). Are self-assessments reliable indicators of topic knowledge? *Proceedings of the American Society for Information Science and Technology*, *47*(1), 1–10. https://doi.org/10.1002/meet.14504701285

Computer and Information Technology Occupations: Occupational Outlook Handbook: : U.S. Bureau of Labor Statistics. (n.d.). Retrieved September 30, 2020, from https://www.bls.gov/ooh/computer-and-information-technology/home.htm

Constant, T., & Levieux, G. (2019). Dynamic Difficulty Adjustment Impact on Players' Confidence. *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, 1–12. https://doi.org/10.1145/3290605.3300693

Constant, T., Levieux, G., Buendia, A., & Natkin, S. (2017). From Objective to Subjective Difficulty Evaluation in Video Games. In R. Bernhaupt, G. Dalvi, A. Joshi, D. K. Balkrishan, J. O'Neill, & M. Winckler (Eds.), *Human-Computer Interaction—INTERACT 2017* (Vol. 10514, pp. 107–127). Springer International Publishing. https://doi.org/10.1007/978-3-319-67684-5_8

Conway, M., Audia, S., Burnette, T., Cosgrove, D., & Christiansen, K. (2000). Alice: Lessons learned from building a 3D system for novices. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems - CHI '00*, 486–493. https://doi.org/10.1145/332040.332481

Conway, M. J. (1998). *Alice: Easy-to-Learn 3D Scripting for Novices* [University of Virginia]. https://doi.org/10.18130/v3m85j

Costikyan, G. (2013). *Uncertainty in games*. Mit Press.

Council, N. R., Sciences, D. on E. and P., Board, C. S. and T., & Thinking, C. for the W. on C. (2010). *Report of a Workshop on the Scope and Nature of Computational Thinking*. National Academies Press.

Csikszentmihalyi, M. (2000). *FLOW: The Psychology of Optimal Experience*. 6.

Csikszentmihalyi, M., & Csikszentmihalyi, I. S. (1992). *Optimal Experience: Psychological Studies of Flow in Consciousness*. Cambridge University Press.

Csizmadia, A., Curzon, P., Dorling, M., Humphreys, S., Ng, T., Selby, C., & Woollard, J. (2015). *Computational thinking—A guide for teachers* [Monograph]. Computing at School. https://eprints.soton.ac.uk/424545/

Davidson, W. M., & Carroll, J. B. (1945). Speed and Level Components in Time-Limit Scores: A Factor Analysis. *Educational and Psychological Measurement*, *5*(4), 411–427. https://doi.org/10.1177/001316444500500408

De Freitas, S., & Jarvis, S. (2006). *A framework for developing serious games to meet learner needs*.

DeDonno, M. A., Rivera-Torres, K., Monis, A., & Fagan, J. F. (2014). The influence of a time limit & bilingualism on scholastic school assessment test performance. *North American Journal of Psychology*, *16*(2), 211–223.

Demirkiran, M. C., & Tansu Hocanin, F. (2021). An investigation on primary school students' dispositions towards programming with game-based learning. *Education and Information Technologies*, *26*(4), 3871–3892.

Denning, P. J. (2017). Remaining trouble spots with computational thinking. *Communications of the ACM*, *60*(6), 33–39. https://doi.org/10.1145/2998438

Dicheva, D., Dichev, C., Agre, G., & Angelova, G. (2015). Gamification in education: A systematic mapping study. *Journal of Educational Technology & Society*, *18*(3), 75–88.

Djambong, T., Freiman, V., Gauvin, S., Paquet, M., & Chiasson, M. (2018). Measurement of Computational Thinking in K-12 Education: The Need for Innovative Practices. In D. Sampson, D. Ifenthaler, J. M. Spector, & P. Isaías (Eds.), *Digital Technologies: Sustainable Innovations for Improving Teaching and Learning* (pp. 193–222). Springer International Publishing. https://doi.org/10.1007/978-3-319-73417-0_12

Domagk, S., Schwartz, R. N., & Plass, J. L. (2010). Interactivity in multimedia learning: An integrated model. *Computers in Human Behavior*, *26*(5), 1024–1033.

Dowker, A. (2017). Interventions for primary school children with difficulties in mathematics. *Advances in Child Development and Behavior*, *53*, 255–287.

Duncan, C., & Bell, T. (2015). A Pilot Computer Science and Programming Course for Primary School Students. *Proceedings of the Workshop in Primary and Secondary Computing Education on ZZZ - WiPSCE '15*, 39–48. https://doi.org/10.1145/2818314.2818328

Echeverria, L., Cobos, R., Morales, M., Moreno, F., & Negrete, V. (2019). Promoting Computational Thinking Skills in Primary School Students to Improve Learning of Geometry. *2019 IEEE Global Engineering Education Conference (EDUCON)*, 424–429. https://doi.org/10.1109/EDUCON.2019.8725088

Effenberger, T., & Pelánek, R. (2018). Towards making block-based programming activities adaptive. *Proceedings of the Fifth Annual ACM Conference on Learning at Scale*, 1–4. https://doi.org/10.1145/3231644.3231670

Eguíluz, A., Garaizar, P., & Guenaga, M. (2018). An Evaluation of Open Digital Gaming Platforms for Developing Computational Thinking Skills. In D. Cvetković (Ed.), *Simulation and Gaming*. InTech. https://doi.org/10.5772/intechopen.71339

Eguiluz, A., Guenaga, M., Garaizar, P., & Olivares-Rodriguez, C. (2017). Exploring the progression of early programmers in a set of computational thinking challenges via clickstream analysis. *IEEE Transactions on Emerging Topics in Computing*, *PP*(99), 1–1. https://doi.org/10.1109/TETC.2017.2768550

Fagerlund, J., Häkkinen, P., Vesisenaho, M., & Viiri, J. (2021). Computational thinking in programming with Scratch in primary schools: A systematic review. *Computer Applications in Engineering Education*, *29*(1), 12–28. https://doi.org/10.1002/cae.22255

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. https://doi.org/10.1016/j.compedu.2012.11.016

Fourali, C. (1997). Using Fuzzy Logic in Educational Measurement: The Case of Portfolio Assessment. *Evaluation & Research in Education*, *11*(3), 129–148. https://doi.org/10.1080/09500799708666923

Francisco-Aparicio, A., Gutiérrez-Vela, F. L., Isla-Montes, J. L., & Sanchez, J. L. G. (2013). Gamification: Analysis and application. In *New trends in interaction, virtual reality and modeling* (pp. 113–126). Springer.

Futschek, G. (2006). Algorithmic Thinking: The Key for Understanding Computer Science. In R. T. Mittermeir (Ed.), *Informatics Education – The Bridge between Using and Understanding Computers* (pp. 159–168). Springer. https://doi.org/10.1007/11915355_15

Gallego-Durán, F. J., Molina-Carmona, R., & Llorens-Largo, F. (2018). Measuring the difficulty of activities for adaptive learning. *Universal Access in the Information Society*, *17*(2), 335–348. https://doi.org/10.1007/s10209-017-0552-x

Gardner, H., & Davis, K. (2013). The App Generation: How Today's Youth Navigate Identity, Intimacy, and Imagination in a Digital World. In *The App Generation*. Yale University Press. https://doi.org/10.12987/9780300199185

Garner, S. (2003). Learning resources and tools to aid novices learn programming. *Informing Science & Information Technology Education Joint Conference (INSITE)*, 213–222.

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

Gautam, A., Bortz, W., & Tatar, D. (2020). Abstraction Through Multiple Representations in an Integrated Computational Thinking Environment. In *Proceedings of the 51st ACM Technical Symposium on Computer Science Education* (pp. 393–399). Association for Computing Machinery. https://doi.org/10.1145/3328778.3366892

Geldreich, K., Simon, A., & Starke, E. (2019). Which Perceptions Do Primary School Children Have about Programming? *Proceedings of the 14th Workshop in Primary and Secondary Computing Education*, 1–7. https://doi.org/10.1145/3361721.3361728

Gokmen, G., Akinci, T. Ç., Tektaş, M., Onat, N., Kocyigit, G., & Tektaş, N. (2010). Evaluation of student performance in laboratory applications using fuzzy logic. *Procedia - Social and Behavioral Sciences*, *2*(2), 902–909. https://doi.org/10.1016/j.sbspro.2010.03.124

Gouws, L., Bradshaw, K., & Wentworth, P. (2013). First year student performance in a test for computational thinking. *Proceedings of the South African Institute for Computer Scientists and Information Technologists Conference on - SAICSIT '13*, 271. https://doi.org/10.1145/2513456.2513484

Graesser, A. C., Wiemer-Hastings, K., Wiemer-Hastings, P., Kreuz, R., & Group, T. R. (1999). AutoTutor: A simulation of a human tutor. *Cognitive Systems Research*, *1*(1), 35–51.

Grover, S., & Pea, R. (2013). Computational Thinking in K–12: A Review of the State of the Field. *Educational Researcher*, *42*(1), 38–43. https://doi.org/10.3102/0013189X12463051

Guenaga, M., Eguíluz, A., Garaizar, P., & Gibaja, J. (2021). How do students develop computational thinking? Assessing early programmers in a maze-based online game. *Computer Science Education*, *31*(2), 259–289. https://doi.org/10.1080/08993408.2021.1903248

Guzdial, M. (2015). Learner-Centered Design of Computing Education: Research on Computing for Everyone. *Synthesis Lectures on Human-Centered Informatics*, *8*(6), 1–165. https://doi.org/10.2200/S00684ED1V01Y201511HCI033

Heckhausen, H. (2013). *The anatomy of achievement motivation* (Vol. 1). Academic press.

Henderson, P. B. (2009). Ubiquitous Computational Thinking. *Computer*, *42*(10), 100–102. https://doi.org/10.1109/MC.2009.334

Henze, N., & Nejdl, W. (2003). Logically characterizing adaptive educational hypermedia systems. *International Workshop on Adaptive Hypermedia and Adaptive Web-Based Systems (AH 2003)*, 20–24.

Hidi, S., & Renninger, K. A. (2006). The four-phase model of interest development. *Educational Psychologist*, *41*(2), 111–127.

Hoffman, B., & Nadelson, L. (2010). Motivational engagement and video gaming: A mixed methods study. *Educational Technology Research and Development*, *58*(3), 245–270.

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

Hooshyar, D., Malva, L., Yang, Y., Pedaste, M., Wang, M., & Lim, H. (2021). An adaptive educational computer game: Effects on students' knowledge and learning attitude in computational thinking. *Computers in Human Behavior*, *114*, 106575.

Hooshyar, D., Pedaste, M., Yang, Y., Malva, L., Hwang, G.-J., Wang, M., Lim, H., & Delev, D. (2021). From Gaming to Computational Thinking: An Adaptive Educational Computer Game-Based Learning Approach. Journal of Educational Computing Research, 59(3), 383–409. https://doi.org/10.1177/0735633120965919

Hooshyar, D., Yousefi, M., & Lim, H. (2018). A Procedural Content Generation-Based Framework for Educational Games: Toward a Tailored Data-Driven Game for Developing Early English Reading Skills. *Journal of Educational Computing Research*, *56*(2), 293–310. https://doi.org/10.1177/0735633117706909

Hou, H.-T., Wu, C.-S., & Wu, C.-H. (2022). Evaluation of a mobile-based scaffolding board game developed by scaffolding-based game editor: Analysis of learners' performance, anxiety and behavior patterns. *Journal of Computers in Education*. https://doi.org/10.1007/s40692-022-00231-1

Hsieh, Y.-H., Lin, Y.-C., & Hou, H.-T. (2015). Exploring elementary-school students' engagement patterns in a game-based learning environment. *Journal of Educational Technology & Society*, *18*(2), 336–348.

Hung, C.-M., Huang, I., & Hwang, G.-J. (2014). Effects of digital game-based learning on students' self-efficacy, motivation, anxiety, and achievements in learning mathematics. *Journal of Computers in Education*, *1*(2), 151–166.

Hutchins, N. (2018). A DSML for a Robotics Environment to Support Synergistic Learning of CT and Geometry. Kong, S. C., Sheldon, J., & Li, K. Y.. (Eds.). Conference. *Proceedings of International Conference on Computational Thinking Education 2018.* https://par.nsf.gov/biblio/10072963-dsml-robotics-environment-support-synergistic-learning-ct-geometry-kong-sheldon-li-eds-conference

ICT specialists in employment—Statistics Explained. (n.d.). Retrieved September 30, 2020, from https://ec.europa.eu/eurostat/statistics-explained/index.php/ICT_specialists_in_employment

Ihantola, P., & Petersen, A. (2019). *Code complexity in introductory programming courses*.

Israel-Fishelson, R., Hershkovitz, A., Eguíluz, A., Garaizar, P., & Guenaga, M. (2020). The Associations Between Computational Thinking and Creativity: The Role of Personal Characteristics: *Journal of Educational Computing Research*. https://doi.org/10.1177/0735633120940954

J. Bagnall, A., & V. Zatuchna, Z. (2005). On the Classification of Maze Problems. In L. Bull & T. Kovacs (Eds.), *Foundations of Learning Classifier Systems* (Vol. 183, pp. 305–316). Springer Berlin Heidelberg. https://doi.org/10.1007/11319122_12

Jeon, I., & Song, K.-S. (2019). The Effect of Learning Analytics System towards Learner's Computational Thinking Capabilities. *Proceedings of the 2019 11th International Conference on Computer and Automation Engineering*, 12–16. https://doi.org/10.1145/3313991.3314017

Jiang, H., Song, Q., Gao, K., Song, Q., & Zhao, X. (2020). Rule-based expert system to assess caving output ratio in top coal caving. *PLOS ONE*, *15*(9), e0238138. https://doi.org/10.1371/journal.pone.0238138

Joarder, A. h., & Firozzaman, M. (2001). Quartiles for Discrete Data. *Teaching Statistics*, *23*(3), 86–89. https://doi.org/10.1111/1467-9639.00063

Jung, J. H., Schneider, C., & Valacich, J. (2010). Enhancing the motivational affordance of information systems: The effects of real-time performance feedback and goal setting in group collaboration environments. *Management Science*, *56*(4), 724–742.

Kalelioglu, F., Gulbahar, Y., & Kukul, V. (2016). A Framework for Computational Thinking Based on a Systematic Research Review. *Baltic Journal of Modern Computing*, *4*, 583–596.

Kanellopoulou, I., Garaizar, P., & Guenaga, M. (2021). First Steps Towards Automatically Defining the Difficulty of Maze-Based Programming Challenges. *IEEE Access*, *9*, 64211–64223.

Kapur, M. (2008). Productive failure. *Cognition and Instruction*, *26*(3), 379–424.

Kapur, M., & Bielaczyc, K. (2012). Designing for productive failure. *Journal of the Learning Sciences*, *21*(1), 45–83.

Kapur, M., & Kinzer, C. K. (2009). Productive failure in CSCL groups. *International Journal of Computer-Supported Collaborative Learning*, *4*(1), 21–46.

Kazimoglu, C., Kiernan, M., Bacon, L., & Mackinnon, L. (2012). A serious game for developing computational thinking and learning introductory computer programming. *Procedia-Social and Behavioral Sciences*, *47*, 1991–1999.

Kelleher, C., Cosgrove, D., Culyba, D., Forlines, C., Pratt, J., & Pausch, R. (2002). Alice2: Programming without syntax errors. *User Interface Software and Technology*, *2*(2), 35–36.

Kelleher, C., & Hnin, W. (2019). Predicting Cognitive Load in Future Code Puzzles. *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, 1–12. https://doi.org/10.1145/3290605.3300487

Khaleel, F. L., Tengku Wook, T. S. M., & Ismail, A. (2016). Gamification elements for learning applications. *International Journal on Advanced Science, Engineering and Information Technology*, *6*(6), 868–874.

Khan, A., Ahmad, F. H., & Malik, M. M. (2017). Use of digital game based learning and gamification in secondary school science: The effect on student engagement, learning and gender difference. *Education and Information Technologies*, *22*(6), 2767–2804. https://doi.org/10.1007/s10639-017-9622-1

Kiili, K., de Freitas, S., Arnab, S., & Lainema, T. (2012). The Design Principles for Flow Experience in Educational Games. *Procedia Computer Science*, *15*, 78–91. https://doi.org/10.1016/j.procs.2012.10.060

Kim, B., Park, H., & Baek, Y. (2009). Not just fun, but serious strategies: Using meta-cognitive strategies in game-based learning. *Computers & Education*, *52*(4), 800–810.

Kim, J. (2006). Task difficulty as a predictor and indicator of web searching interaction. *CHI '06 Extended Abstracts on Human Factors in Computing Systems*, 959–964. https://doi.org/10.1145/1125451.1125636

Kingsbury, G. G., & Weiss, D. J. (1983). A comparison of IRT-based adaptive mastery testing and a sequential mastery testing procedure. In *New horizons in testing* (pp. 257–283). Elsevier.

Kinshuk, T. L., & Lin, T. (2004). Application of learning styles adaptivity in mobile learning environments. *Third Pan Commonwealth Forum on Open Learning*, 4–8.

Kirkley, J., Kirkley, S., & Heneghan, J. (2007). Building bridges between serious game design and instructional design: A blueprint for now and the future. In *The design and use of simulation computer games in education* (pp. 61–83). Brill Sense.

Koedinger, K. R. (2001). Cognitive tutors as modeling tools and instructional models. In *Smart machines in education: The coming revolution in educational technology* (pp. 145–167).

Koupritzioti, D., & Xinogalos, S. (2020). PyDiophantus maze game: Play it to learn mathematics or implement it to learn game programming in Python. *Education and Information Technologies*, *25*(4), 2747–2764. https://doi.org/10.1007/s10639-019-10087-1

Kuittinen, M., & Sajaniemi, J. (2004). Teaching roles of variables in elementary programming courses. *Proceedings of the 9th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education*, 57–61. https://doi.org/10.1145/1007996.1008014

Kwon, K., & Cheon, J. (2019). Exploring Problem Decomposition and Program Development through Block-Based Programs. *International Journal of Computer Science Education in Schools*, *3*(1). https://eric.ed.gov/?id=EJ1214684

Landers, R. N., & Landers, A. K. (2014). An empirical test of the theory of gamified learning: The effect of leaderboards on time-on-task and academic performance. *Simulation & Gaming*, *45*(6), 769–785.

Latham, G. P., Seijts, G., & Crim, D. (2008). The effects of learning goal difficulty level and cognitive ability on performance. *Canadian Journal of Behavioural Science / Revue Canadienne Des Sciences Du Comportement*, *40*(4), 220–229. https://doi.org/10.1037/a0013114

Lee, M., & Lee, J. (2021). Enhancing computational thinking skills in informatics in secondary education: The case of South Korea. *Educational Technology Research and Development*, *69*(5), 2869–2893. https://doi.org/10.1007/s11423-021-10035-2

Lee, Y.-J. (2011). Empowering teachers to create educational software: A constructivist approach utilizing Etoys, pair programming and cognitive apprenticeship. *Computers & Education*, *56*(2), 527–538. https://doi.org/10.1016/j.compedu.2010.09.018

Leutner, D. (1993). Guided discovery learning with computer-based simulation games: Effects of adaptive and non-adaptive instructional support. *Learning and Instruction*, *3*(2), 113–132.

Li, Y., & Belkin, N. J. (2008). A faceted approach to conceptualizing tasks in information seeking. *Information Processing & Management*, *44*(6), 1822–1837. https://doi.org/10.1016/j.ipm.2008.07.005

Linacre. (2000). *Computer-adaptive testing: A methodology whose time has come. MESA Memorandum No 9*.

Liu, H., Wu, Z., Lu, Y., & Zhu, L. (2022). Exploring the Balance Between Computational Thinking and Learning Motivation in Elementary Programming Education: An Empirical Study with Game-based Learning. *IEEE Transactions on Games*.

Liu, J., Liu, C., Yuan, X., & Belkin, N. J. (2011). Understanding searchers' perception of task difficulty: Relationships with task type. *Proceedings of the American Society for Information Science and Technology*, *48*(1), 1–10. https://doi.org/10.1002/meet.2011.14504801152

Lomas, D., Koedinger, K., Patel, N., Shodhan, S., Poonawala, N., & L. Forlizzi, J. (2017, May 6). *Is Difficulty Overrated? The Effects of Choice, Novelty and Suspense on Intrinsic Motivation in Educational Games*. https://doi.org/10.1145/3025453.3025638

Lopes, R., & Bidarra, R. (2011). Adaptivity challenges in games and simulations: A survey. *IEEE Transactions on Computational Intelligence and AI in Games*, *3*(2), 85–99.

Ludi, S. (2015). Position paper: Towards making block-based programming accessible for blind users. *2015 IEEE Blocks and Beyond Workshop (Blocks and Beyond)*, 67–69. https://doi.org/10.1109/BLOCKS.2015.7369005

Lui, D., Walker, J. T., Hanna, S., Kafai, Y. B., Fields, D., & Jayathirtha, G. (2020). Communicating computational concepts and practices within high school students' portfolios of making electronic textiles. *Interactive Learning Environments*, *28*(3), 284–301. https://doi.org/10.1080/10494820.2019.1612446

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

Lynch, R., Hurley, A., Cumiskey, O., Nolan, B., & McGlynn, B. (2019). Exploring the relationship between homework task difficulty, student engagement and performance. *Irish Educational Studies*, *38*(1), 89–103.

Magdalena, L. (2015). Fuzzy Rule-Based Systems. In J. Kacprzyk & W. Pedrycz (Eds.), *Springer Handbook of Computational Intelligence* (pp. 203–218). Springer. https://doi.org/10.1007/978-3-662-43505-2_13

Magis, D., & Barrada, J. R. (2017). Computerized Adaptive Testing with *R*: Recent Updates of the Package **catR**. *Journal of Statistical Software*, *76*(Code Snippet 1). https://doi.org/10.18637/jss.v076.c01

Malik, S., Al-Emran, M., Mathew, R., Tawafak, R., & AlFarsi, G. (2020). Comparison of E-learning, M-learning and game-based learning in programming education–a gendered analysis. *International Journal of Emerging Technologies in Learning (IJET)*, *15*(15), 133–146.

Malliarakis, C., Satratzemi, M., & Xinogalos, S. (2017). CMX: The Effects of an Educational MMORPG on Learning and Teaching Computer Programming. *IEEE Transactions on Learning Technologies*, *10*(02), 219–235. https://doi.org/10.1109/TLT.2016.2556666

Malone, T. W. (1982). Heuristics for designing enjoyable user interfaces: Lessons from computer games. *Proceedings of the 1982 Conference on Human Factors in Computing Systems*, 63–68.

Maloney, J., Burd, L., Kafai, Y., Rusk, N., Silverman, B., & Resnick, M. (2004). Scratch: A sneak preview [education]. *Proceedings. Second International Conference on Creating, Connecting and Collaborating through Computing, 2004.*, 104–109.

Mannila, L., Dagiene, V., Demo, B., Grgurina, N., Mirolo, C., Rolandsson, L., & Settle, A. (2014). Computational Thinking in K-9 Education. *Proceedings of the Working Group Reports of the 2014 on Innovation & Technology in Computer Science Education Conference - ITiCSE-WGR '14*, 1–29. https://doi.org/10.1145/2713609.2713610

Markowitz, D. G. (2004). Evaluation of the Long-Term Impact of a University High School Summer Science Program on Students' Interest and Perceived Abilities in Science. *Journal of Science Education and Technology*, *13*(3), 395–407. https://doi.org/10.1023/B:JOST.0000045467.67907.7b

Mathrani, A., Christian, S., & Ponder-Sutton, A. (2016). PlayIT: Game based learning approach for teaching programming concepts. *Journal of Educational Technology & Society*, *19*(2), 5–17.

Mayes, T., & De Freitas, S. (2007). Learning and e-learning. *Rethinking Pedagogy for a Digital Age*, 13–25.

McBride, J. R., & Martin, J. T. (1983). Reliability and validity of adaptive ability tests in a military setting. In *New horizons in testing* (pp. 223–236). Elsevier.

McClendon, M. S. (2001). *The Complexity and Difficulty of a Maze*. Bridges: Mathematical Connections in Art, Music, and Science. http://at.yorku.ca/c/a/h/d/25.htm

McCormick, K. I., & Hall, J. A. (2021). Computational thinking learning experiences, outcomes, and research in preschool settings: A scoping review of literature. *Education and Information Technologies*. https://doi.org/10.1007/s10639-021-10765-z

McKinley, R. L., & Reckase, M. D. (1980). Computer applications to ability testing. *AEDS Journal*, *13*(3), 193–203.

Mekler, E. D., Brühlmann, F., Tuch, A. N., & Opwis, K. (2017). Towards understanding the effects of individual gamification elements on intrinsic motivation and performance. *Computers in Human Behavior*, *71*, 525–534.

Mestadi, W., Nafil, K., Touahni, R., & Messoussi, R. (2018). An assessment of serious games technology: Toward an architecture for serious games design. *International Journal of Computer Games Technology*, *2018*.

Michael, D. R., & Chen, S. L. (2005). *Serious games: Games that educate, train, and inform*. Muska & Lipman/Premier-Trade.

Min, W., Mott, B., Park, K., Taylor, S., Akram, B., Wiebe, E., Boyer, K. E., & Lester, J. (2020). Promoting Computer Science Learning with Block-Based Programming and Narrative-Centered Gameplay. *2020 IEEE Conference on Games (CoG)*, 654–657. https://doi.org/10.1109/CoG47356.2020.9231881

Mitamura, T., Suzuki, Y., & Oohori, T. (2012). Serious games for learning programming languages. *2012 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, 1812–1817.

Mitnik, R., Recabarren, M., Nussbaum, M., & Soto, A. (2009). Collaborative robotic instruction: A graph teaching experience. *Computers & Education*, *53*(2), 330–342. https://doi.org/10.1016/j.compedu.2009.02.010

Mohamad, S. N. M., Salam, S., & Bakar, N. (2017). *An analysis of gamification elements in online learning to enhance learning engagement*.

Moreno-León, J., & Robles, G. (2015). Dr. Scratch: A Web Tool to Automatically Evaluate Scratch Projects. *Proceedings of the Workshop in Primary and Secondary Computing Education on ZZZ - WiPSCE '15*, 132–133. https://doi.org/10.1145/2818314.2818338

Mulholland, P. (1998). *A framework for describing and evaluating software visualisation systems: A case-study in Prolog*.

Mullane, J., & McKelvie, S. (2019). Effects of Removing the Time Limit on First and Second Language Intelligence Test Performance. *Practical Assessment, Research, and Evaluation*, *7*(1). https://doi.org/10.7275/ph8y-yz89

Muratet, M., Torguet, P., Jessel, J.-P., & Viallet, F. (2009). Towards a Serious Game to Help Students LearnComputer Programming. *International Journal of Computer Games Technology*, *2009*.

Navarro, J.-J., García-Rubio, J., & Olivares, P. R. (2015). The Relative Age Effect and Its Influence on Academic Performance. *PLoS ONE*, *10*(10). https://doi.org/10.1371/journal.pone.0141895

Ng, A. K., Atmosukarto, I., Cheow, W. S., Avnit, K., & Yong, M. H. (2021). Development and implementation of an online adaptive gamification platform for learning computational thinking. *2021 IEEE Frontiers in Education Conference (FIE)*, 1–6.

Nicholls, J. G., & Miller, A. T. (1983). The differentiation of the concepts of difficulty and ability. *Child Development*, 951–959.

Oblinger, D. G. (2006). Games and learning. *Educause Quarterly*, *29*(3), 5–7.

O'Kelly, J., & Gibson, J. P. (2006). RoboCode & problem-based learning: A non-prescriptive approach to teaching programming. *Proceedings of the 11th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education*, 217–221.

Olivares-Rodríguez, C., Guenaga, M., & Garaizar, P. (2018). Using children's search patterns to predict the quality of their creative problem solving. *Aslib Journal of Information Management*, *70*(5), 538–550. https://doi.org/10.1108/AJIM-05-2018-0103

Osbourne, J. W., & Waters, E. (2002). Four Assumptions of Multiple Regression That Researchers Should Always Test. *Practical Assessment, Research & Evaluation*, *8*(2).

Palts, T. (2020). A Model for Developing Computational Thinking Skills. *Informatics in Education*, *19*(1), 113–128.

Panskyi, T., & ROWIŃSKA, Z. (2021). A holistic digital game-based learning approach to out-of-school primary programming education. *Informatics in Education*, *20*(2), 255–276.

Papadakis, S., & Kalogiannakis, M. (2019). Evaluating the effectiveness of a game-based learning approach in modifying students' behavioural outcomes and competence, in an introductory programming course. A case study in Greece. *International Journal of Teaching and Case Studies*, *10*(3), 235–250.

Papoušek, J., & Pelánek, R. (2015). Impact of Adaptive Educational System Behaviour on Student Motivation. In C. Conati, N. Heffernan, A. Mitrovic, & M. F. Verdejo (Eds.), *Artificial Intelligence in Education* (pp. 348–357). Springer International Publishing. https://doi.org/10.1007/978-3-319-19773-9_35

Park, K., Mott, B. W., Min, W., Boyer, K. E., Wiebe, E. N., & Lester, J. C. (2019). Generating Educational Game Levels with Multistep Deep Convolutional Generative Adversarial Networks. *2019 IEEE Conference on Games (CoG)*, 1–8. https://doi.org/10.1109/CIG.2019.8848085

Pelánek, R., & Effenberger, T. (2020). Design and analysis of microworlds and puzzles for block-based programming. *Computer Science Education*, *0*(0), 1–39. https://doi.org/10.1080/08993408.2020.1832813

Perkins, K., Adams, W., Dubson, M., Finkelstein, N., Reid, S., Wieman, C., & LeMaster, R. (2006). PhET: Interactive Simulations for Teaching and Learning Physics. *The Physics Teacher*, *44*(1), 18–23. https://doi.org/10.1119/1.2150754

Plass, J. L., Chun, D. M., Mayer, R. E., & Leutner, D. (1998). Supporting visual and verbal learning preferences in a second-language multimedia learning environment. *Journal of Educational Psychology*, *90*(1), 25.

Plass, J. L., Homer, B. D., & Kinzer, C. K. (2015). Foundations of Game-Based Learning. *Educational Psychologist*, *50*(4), 258–283. https://doi.org/10.1080/00461520.2015.1122533

Plass, J. L., Perlin, K., & Nordlinger, J. (2010). The games for learning institute: Research on design patterns for effective educational games. *Game Developers Conference, San Francisco, CA*.

Powers, D. E., & Fowles, M. E. (1996). Effects of Applying Different Time Limits to a Proposed GRE Writing Test. *Journal of Educational Measurement*, *33*(4), 433–452. https://doi.org/10.1111/j.1745-3984.1996.tb00500.x

Prensky, M. (2003a). Digital game-based learning Computers in Entertainment (CIE). *Listen to the Natives Educational Leadership*.

Prensky, M. (2003b). Digital game-based learning. *Computers in Entertainment*, *1*(1), 21. https://doi.org/10.1145/950566.950596

Price, B. A., Baecker, R. M., & Small, I. S. (1993). A principled taxonomy of software visualization. *Journal of Visual Languages & Computing*, *4*(3), 211–266.

Przybylski, A. K., Murayama, K., DeHaan, C. R., & Gladwell, V. (2013). Motivational, emotional, and behavioral correlates of fear of missing out. *Computers in Human Behavior*, *29*(4), 1841–1848. https://doi.org/10.1016/j.chb.2013.02.014

Relkin, E., de Ruiter, L. E., & Bers, M. U. (2021). Learning to code and the acquisition of computational thinking by young children. *Computers & Education*, *169*, 104222. https://doi.org/10.1016/j.compedu.2021.104222

Repenning, A., Basawapatna, A., & Escherle, N. (2016). Computational thinking tools. *2016 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, 218–222. https://doi.org/10.1109/VLHCC.2016.7739688

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

Rickel, J., & Johnson, W. L. (1997). Integrating pedagogical capabilities in a virtual environment agent. *Proceedings of the First International Conference on Autonomous Agents*, 30–38.

Robins, A., Rountree, J., & Rountree, N. (2003). Learning and Teaching Programming: A Review and Discussion. *Computer Science Education*, *13*(2), 137–172. https://doi.org/10.1076/csed.13.2.137.14200

Rotgans, J. I., & Schmidt, H. G. (2011). Situational interest and academic achievement in the active-learning classroom. *Learning and Instruction*, *21*(1), 58–67.

Rubio-Tamayo, J. L., Gértrudix, M., & García, F. (2017). Immersive Environments and Virtual Reality: Systematic Review and Advances in Communication, Interaction and Simulation. *Multimodal Technologies and Interaction*, *1*. https://doi.org/10.3390/mti1040021

Rudder, A., Bernard, M., & Mohammed, S. (2007). Teaching programming using visualization. *Proceedings of the Sixth Conference on IASTED International Conference Web-Based Education - Volume 2*, 487–492.

Rushkoff, D. (2010). *Program Or Be Programmed: Ten Commands for a Digital Age*. OR Books.

Sampayo-Vargas, S., Cope, C. J., He, Z., & Byrne, G. J. (2013). The effectiveness of adaptive difficulty adjustments on students' motivation and learning in an educational computer game. *Computers & Education*, *69*, 452–462. https://doi.org/10.1016/j.compedu.2013.07.004

Saqr, M., & Tedre, M. (2019). Should we teach computational thinking and big data principles to medical students? *International Journal of Health Sciences*, *13*(4), 1–2.

Sarkar, N. I. (2006). Teaching computer networking fundamentals using practical laboratory exercises. *IEEE Transactions on Education*, *49*(2), 285–291.

Selby, C., & Woollard, J. (2013). *Computational thinking: The developing definition* [Monograph]. University of Southampton (E-prints). https://eprints.soton.ac.uk/356481/

Seoane Pardo, A. M. (2018). Computational Thinking Between Philosophy and STEM—Programming Decision Making Applied to the Behavior of "Moral Machines" in Ethical Values Classroom. *IEEE Revista Iberoamericana de Tecnologias Del Aprendizaje*, *13*(1), 20–29. https://doi.org/10.1109/RITA.2018.2809940

Shaker, N., Liapis, A., Togelius, J., Lopes, R., & Bidarra, R. (2016). Constructive generation methods for dungeons and levels. In *Procedural Content Generation in Games* (pp. 31–55). Springer.

Shute, V. J., & Zapata-Rivera, D. (2012). Adaptive educational systems. *Adaptive Technologies for Training and Education*, *7*(27), 1–35.

Smith, P. A., & Webb, G. F. (1999). Evaluation of Low-Level Program Visialization for Teaching Novice C Programmers. *Proceedings of ICCE*, *99*(7), 1.

Soflano, M. (2011). Modding in serious games: Teaching structured query language (sql) using neverwinter nights. In *Serious Games and edutainment applications* (pp. 347–368). Springer.

Soflano, M., Connolly, T. M., & Hainey, T. (2015). An application of adaptive games-based learning based on learning style to teach SQL. *Computers & Education*, *86*, 192–211. https://doi.org/10.1016/j.compedu.2015.03.015

Steinkuehler, C., & Duncan, S. (2008). Scientific habits of mind in virtual worlds. *Journal of Science Education and Technology*, *17*(6), 530–543.

Sun, L., Hu, L., & Zhou, D. (2021). Improving 7th-graders' computational thinking skills through unplugged programming activities: A study on the influence of multiple factors. *Thinking Skills and Creativity*, *42*, 100926. https://doi.org/10.1016/j.tsc.2021.100926

Szabó, M., Pomázi, K. D., Radostyán, B., Szegletes, L., & Forstner, B. (2016). Estimating task difficulty in educational games. *2016 7th IEEE International Conference on Cognitive Infocommunications (CogInfoCom)*, 000397–000402. https://doi.org/10.1109/CogInfoCom.2016.7804582

Tavakoli, P. (2009). Investigating task difficulty: Learners' and teachers' perceptions. *International Journal of Applied Linguistics*, *19*(1), 1–25.

Tengler, K., Kastner-Hauler, O., & Sabitzer, B. (2021). A Robotics-based Learning Environment Supporting Computational Thinking Skills—Design and Development. *2021 IEEE Frontiers in Education Conference (FIE)*, 1–6. https://doi.org/10.1109/FIE49875.2021.9637351

Tenório, K., Chalco Challco, G., Dermeval, D., Lemos, B., Nascimento, P., Santos, R., & Pedro da Silva, A. (2020). Helping Teachers Assist Their Students in Gamified Adaptive Educational Systems: Towards a Gamification Analytics Tool. In I. I. Bittencourt, M. Cukurova, K. Muldner, R. Luckin, & E. Millán (Eds.), *Artificial Intelligence in Education* (pp. 312–317). Springer International Publishing. https://doi.org/10.1007/978-3-030-52240-7_57

Ternik, Ž., Koron, A., Koron, T., & Šerbec, I. N. (2017). Learning Programming Concepts Through Maze Game in Scratch. *European Conference on Games Based Learning; Reading*, 661–670. https://search.proquest.com/docview/1967728949/abstract/573F8E28B0344D95PQ/1

Thomsen, B. (2008). Using on-line tutorials in introductory IT courses. In *Reflections on the Teaching of Programming* (pp. 68–74). Springer.

Tikva, C., & Tambouris, E. (2021). Mapping computational thinking through programming in K-12 education: A conceptual model based on a systematic literature Review. *Computers & Education*, *162*, 104083. https://doi.org/10.1016/j.compedu.2020.104083

Tinker, R. F., & Xie, Q. (2008). Applying Computational Science to Education: The Molecular Workbench Paradigm. *Computing in Science Engineering*, *10*(5), 24–27. https://doi.org/10.1109/MCSE.2008.108

Tlili, A., Denden, M., Essalmi, F., Jemni, M., Kinshuk, Chen, N., & Huang, R. (2019). Does Providing a Personalized Educational Game Based on Personality Matter? A Case Study. *IEEE Access*, *7*, 119566–119575. https://doi.org/10.1109/ACCESS.2019.2936384

Tlili, A., Essalmi, F., Ayed, L. J. B., Jemni, M., & Kinshuk. (2017). A Smart Educational Game to Model Personality Using Learning Analytics. *2017 IEEE 17th International Conference on Advanced Learning Technologies (ICALT)*, 131–135. https://doi.org/10.1109/ICALT.2017.65

Tomlinson, C. A., Brighton, C., Hertberg, H., Callahan, C. M., Moon, T. R., Brimijoin, K., Conover, L. A., & Reynolds, T. (2003). Differentiating instruction in response to student readiness, interest, and learning profile in academically diverse classrooms: A review of literature. *Journal for the Education of the Gifted*, *27*(2–3), 119–145.

Torres-Torres, Y.-D., Román-González, M., & Pérez-González, J.-C. (2021). Specific Didactic Strategies Used for the Development of Computational Thinking in the Female Collective in Primary and Secondary Education: A Systematic Review Protocol. *Ninth International Conference on Technological Ecosystems for Enhancing Multiculturality (TEEM'21)*, 25–29. https://doi.org/10.1145/3486011.3486414

Troussas, C., Krouska, A., & Sgouropoulou, C. (2020). Collaboration and fuzzy-modeled personalization for mobile game-based learning in higher education. *Computers & Education*, *144*, 103698. https://doi.org/10.1016/j.compedu.2019.103698

Turkay, S., Kinzer, C. K., & Adinolf, S. (2013). The effects of customization on game experiences of a massively multiplayer online game's players. *Proceedings of GLS*, *9*, 330–337.

Vakkari, P. (1999). Task complexity, problem structure and information actions: Integrating studies on information seeking and retrieval. *Information Processing & Management*, *35*(6), 819–837.

van der Linden, W. J., & Glas, C. A. W. (Eds.). (2010). *Elements of Adaptive Testing*. Springer New York. https://doi.org/10.1007/978-0-387-85461-8

Van Eck, R. (2006). Digital game-based learning: It's not just the digital natives who are restless. *EDUCAUSE Review*, *41*(2), 16.

Van Eck, R. (2007). Building artificially intelligent learning games. In *Games and simulations in online learning: Research and development frameworks* (pp. 271–307). IGI global.

Van Eck, R. (2015). Digital game-based learning: Still restless, after all these years. *EDUCAUSE Review*, *50*(6), 13.

van Oostendorp, H., Van der Spek, E. D., & Linssen, J. (2014). *Adapting the Complexity Level of a Serious Game to the Proficiency of Players.(2014)*.

Vanbecelaere, S., Berghe, K. V. den, Cornillie, F., Sasanguie, D., Reynvoet, B., & Depaepe, F. (2020). The effectiveness of adaptive versus non-adaptive learning with digital educational games. *Journal of Computer Assisted Learning*, *36*(4), 502–513. https://doi.org/10.1111/jcal.12416

Wainer, H., Dorans, N. J., Flaugher, R., Green, B. F., & Mislevy, R. J. (2000). *Computerized adaptive testing: A primer*. Routledge.

Wang, X., Schneider, C., & Valacich, J. S. (2015). Enhancing creativity in group collaboration: How performance targets and feedback shape perceptions and idea generation performance. *Computers in Human Behavior*, *42*, 187–195.

Weintrop, D. (2019). Block-based programming in computer science education. *Communications of the ACM*, *62*(8), 22–25. https://doi.org/10.1145/3341221

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

Weintrop, D., & Wilensky, U. (2015). To Block or Not to Block, That is the Question: Students' Perceptions of Blocks-based Programming. *Proceedings of the 14th International Conference on Interaction Design and Children*, 199–208. https://doi.org/10.1145/2771839.2771860

Weiss, D. J. (1982). Improving measurement quality and efficiency with adaptive testing. *Applied Psychological Measurement*, *6*(4), 473–492.

Weiss, D. J., & Kingsbury, G. G. (1984). Application of Computerized Adaptive Testing to Educational Problems. *Journal of Educational Measurement*, *21*(4), 361–375. https://doi.org/10.1111/j.1745-3984.1984.tb01040.x

Wilensky, U., & Reisman, K. (2006). Thinking Like a Wolf, a Sheep, or a Firefly: Learning Biology Through Constructing and Testing Computational Theories—An Embodied

Modeling Approach. *Cognition and Instruction*, *24*(2), 171–209. https://doi.org/10.1207/s1532690xci2402_1

Willmott, C. J., & Matsuura, K. (2005). Advantages of the mean absolute error (MAE) over the root mean square error (RMSE) in assessing average model performance. *Climate Research*, *30*(1), 79–82. https://doi.org/10.3354/cr030079

Wilson, A., Hainey, T., & Connolly, T. (2012). Evaluation of Computer Games Developed by Primary School Children to Gauge Understanding of Programming Concepts. *Proceedings of the 6th European Conference on Games Based Learning*, 549–558.

Wing, J. M. (n.d.). *Research Notebook: Computational Thinking—What and Why?* 8.

Wing, J. M. (2006). Computational Thinking. *Commun. ACM*, *49*(3), 33–35. https://doi.org/10.1145/1118178.1118215

Wing, J. M., & Stanzione, D. (2016). Progress in computational thinking, and expanding the HPC community. *Communications of the ACM*, *59*(7), 10–11. https://doi.org/10.1145/2933410

Wu, S.-Y. (2018). The Development and Challenges of Computational Thinking Board Games. *2018 1st International Cognitive Cities Conference (IC3)*, 129–131. https://doi.org/10.1109/IC3.2018.00-45

Wu, W.-H., Hsiao, H.-C., Wu, P.-L., Lin, C.-H., & Huang, S.-H. (2012). Investigating the learning-theory foundations of game-based learning: A meta-analysis. *Journal of Computer Assisted Learning*, *28*(3), 265–279. https://doi.org/10.1111/j.1365-2729.2011.00437.x

Xinogalos, S., Satratzemi, M., & Malliarakis, C. (2017). Microworlds, games, animations, mobile apps, puzzle editors and more: What is important for an introductory programming environment? *Education and Information Technologies*, *22*(1), 145–176. https://doi.org/10.1007/s10639-015-9433-1

Zadeh, L. A. (1988). Fuzzy logic. *Computer*, *21*(4), 83–93. https://doi.org/10.1109/2.53

Zapušek, M., & Rugelj, J. (2013). Learning programming with serious games. *EAI Endorsed Transactions on Serious Games*, *1*(1).

Zhao, D., Muntean, C. H., Chis, A. E., Rozinaj, G., & Muntean, G.-M. (2022). Game-Based Learning: Enhancing Student Experience, Knowledge Gain, and Usability in Higher Education Programming Courses. *IEEE Transactions on Education*.

Zyda, M. (2005). From visual simulation to virtual reality to games. *Computer*, *38*(9), 25–32.