Stephan Keller, BSc

Enhancing Computational Thinking: A Mobile Augmented Reality Game-Based Approach

Master's Thesis
to achieve the university degree of
Diplom-Ingenieur
Master's degree programme: Computer Science

submitted to

Graz University of Technology

Supervisor
Assoc.Prof. Dipl.-Ing. Dr.techn. Christian Gütl
Co-Supervisor
Ass.Prof. Dipl.-Ing. Dr.techn. Johanna Pirker, BSc.

Institute for Interactive Systems and Data Science
Head: Univ.-Prof. Dipl.-Inf. Dr. Stefanie Lindstaedt

Graz, October 2019

Stephan Keller, BSc

Förderung von informatischem Denken durch ein mobiles Augmented Reality Lernspiel

Masterarbeit
zur Erlangung des akademischen Grades
Diplom-Ingenieur
Masterstudium: Informatik

eingereicht an der

Technischen Universität Graz

Betreuer
Assoc.Prof. Dipl.-Ing. Dr.techn. Christian Gütl
Co-Betreuer
Ass.Prof. Dipl.-Ing. Dr.techn. Johanna Pirker, BSc.

Institute for Interactive Systems and Data Science
Univ.-Prof. Dipl.-Inf. Dr. Stefanie Lindstaedt

Graz, Oktober 2019

Affidavit

I declare that I have authored this thesis independently, that I have not used other than the declared sources/resources, and that I have explicitly indicated all material which has been quoted either literally or by content from the sources used. The text document uploaded to TUGRAZonline is identical to the present master's thesis.

Date	Signature

Eidesstattliche Erklärung

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst, andere als die angegebenen Quellen/Hilfsmittel nicht benutzt, und die den benutzten Quellen wörtlich und inhaltlich entnommenen Stellen als solche kenntlich gemacht habe. Das in TUGRAZonline hochgeladene Textdokument ist mit der vorliegenden Masterarbeit identisch.

Datum	Unterschrift

Acknowledgements

First of all, I would like to express my sincere appreciation to my supervisors Prof. Christian Gütl and Prof. Johanna Pirker, for their support and for believing in me.

I am grateful to Prof. Bernadette Reiter for her assistance and for making the evaluation in this thesis possible. I would like to extend my gratitude to all the voluntary participants of the evaluation.

Special thanks to Prof. Foaad Khosmood, the main organizer of FDG 2019. The grant made it possible for me to present my work at the Games Educator Summit during the conference in San Luis Obispo, California.

I want to thank Graz University of Technology and Curtin University for offering me an international research experience in Perth, Western Australia.

I wish to acknowledge the support of my family, my friends and my colleagues. Thank you for the constructive pressure, which you knew I needed.

Last but not least, I wish to express my deepest gratitude to my partner Nicole. Her endless support and encouragement were essential for the completion of this thesis.

Abstract

The modern technologized working world is becoming increasingly complex. It demands problem-solving skills and a way of thinking that bridges the gap between humans and computers. Several thinking patterns have emerged from software development and computer science in the last decades to deal with complex problems. These patterns and the respective skills are summarized under the term computational thinking (CT). For today's young students, these skills will be as essential as mathematics. These students belong to the so-called Generation Z, growing up surrounded by quickly evolving mobile technology. There are efforts to promote CT in education internationally. However, national curricula adapt slowly while technology evolves rapidly. It is still a matter of debate, what exactly should be learned and how to assess it.

This thesis investigates how to make CT education interesting for young students. We want to know if a mobile augmented reality game-based approach encourages young students to engage with CT concepts. Augmented reality on mobile devices is relatively new. There is little research on mobile augmented reality in combination with CT education. Alongside this thesis, the mobile augmented reality game *ARobot* was designed and developed. The game was evaluated with a group of young students and a group of university students. The play-test was recorded on-screen and through a think-aloud protocol. A post-play-test questionnaire measured the usability of the game. In the concluding interview, the participants reflected and expressed their experience with the game. Most participants made quick progress and finished the game in the assigned time. The mean of all system usability scores is considered excellent. The comparison group with graduate and undergraduate students showed that usability was perceived very well independent of age. Analysis of the play-test recordings and the interviews showed that mobile augmented reality was very well accepted. It is yet unclear how well in-game experiences transfer to more general applications. The results confirm the assumption that students with prior programming knowledge progress faster in the game. Based on these results, we conclude that mobile augmented reality and game-based learning are promising tools to accompany CT education and deserve more attention.

Kurzfassung

Die moderne technologisierte Arbeitswelt wird zunehmend komplexer. Es werden Fachkräfte gebraucht, die Technologie verstehen und mitgestalten. Informatisches Denken (vom englischen Computational Thinking) ist ein Überbegriff für Fähigkeiten die an die Arbeitsweise von Computern angelehnt sind um komplexe Probleme zu lösen. Diese Fähigkeiten haben sich aus der Informatik und Softwareentwicklung der letzten Jahrzente herauskristallisiert und werden für die Jugend von heute in Zukunft eine große Rolle spielen. Die Kinder der sogenannten Generation Z sind mit rasanten Technologieentwicklungen aufgewachsen und auch ihre Art zu Lernen hat sich verändert.

Diese Arbeit untersucht, wie man Informatisches Denken für junge Schüler interessant gestalten kann. Wir wollen wissen, ob ein mobiles Augmented Reality Lernspiel junge Schüler dazu motivieren kann, sich mit Informatischem Denken auseinanderzusetzen. Augmented Reality auf mobilen Geräten ist eine relativ neue Entwicklung. Dementsprechend gibt es bisher wenig Forschung auf diesem Gebiet in Kombination mit informatischem Denken. In dieser Arbeit wurde das mobile Augmented Reality Spiel *ARobot* designt und entwickelt. Das Spiel wurde mit einer Gruppe Schülern und einer Gruppe Studenten evaluiert. Während des Tests wurden Display und Ton (Thinking Aloud) aufgezeichnet. Nach dem Test beantworteten die Teilnehmer einen Fragebogen zur System-Gebrauchstauglichkeit. In einem abschließenden Interview reflektierten die Teilnehmer ihre Erfahrung mit dem Spiel. Die meisten Teilnehmer haben sich schnell in das Spiel eingelebt und haben es in der vorgegebenen Zeit fertig gespielt. Der Mittelwert der Ergebnisse des Fragebogens zur System-Gebrauchstauglichkeit wird als exzellent eingestuft. Die Analyse der Aufzeichnungen und der Interviews zeigte, dass der mobile Augmented Reality Ansatz sehr gut angenommen wurde. Basierend auf diesen Ergebnissen folgern wir, dass die Kombination aus Augmented Reality und mobilem Lernspiel einen vielversprechenden Ansatz zum Lernen von Informatischem Denken darstellt.

Contents

1. Introduction

Industry 4.0, Autonomous Vehicles, Internet of Things, Artificial Intelligence and mobile devices are just a few phenomena of modern technology. The world today is connected, smart and digital. New STEM (short for science, technology, engineering and mathematics) jobs emerge, future employees with IT understanding and problem-solving skills are needed in many fields. Education curricula are adapting to this development (García-Peñalvo & Cruz-Benito, 2016). Computational Thinking (CT) as part of STEM education, has been an ongoing topic for the last decade. There have been efforts to promote CT in education research and curricula all around the world. However, CT education is still an open topic with little consensus.

1.1. Motivation

When teaching CT, it is non-trivial to clarify what exactly is learned and measure how well it is understood. The different concepts to be learned might be very theoretical and dry, so motivating students is an additional non-trivial task. Literature suggests that game-based learning can be utilized to increase motivation (Papastergiou, 2009). For creating a digital educational game, it is essential to understand the underlying principles to make it entertaining and thereby motivating to learn. CT education, with the assistance of mobile augmented reality, is a relatively new field with little existing research.

In this thesis, introductory CT learning targets are identified and embedded in a mobile augmented reality game. The game prototype is evaluated with two groups. The evaluation gives insights on mobile augmented reality game-based learning for CT. Analysis of thinking aloud and screen recordings from the participants provide interesting observations on the learning process.

The game prototype and the evaluation bring insights into the following research questions:

Q1 Is our mobile augmented reality game suitable to engage students in classroom with computational thinking concepts?
Q2 Does marker-based AR enhance the mobile learning experience?
Q3 How do lessons learned in-game transfer to an on-paper task?

1.2. Outline

This thesis is structured into three segments: research, prototype and outcome. The comprehensive literature research in Chapter 2 reveals interesting spots and the current status of the relevant topics. At first, the history of CT is revisited up to the current status of CT in education. Then literature on learning and teaching is reviewed with a focus on modern forms such as technology-enhanced learning and game-based learning. A section on game development illustrates which state-of-the-art tools can be used to develop a game. The chapter concludes with a section on augmented reality, which is becoming increasingly interesting.

Chapter 3 covers related work separated into three categories: code, play and augmented reality. First constructivist approaches are reviewed, such as block-based programming. The user creates animations or programs through simplified code. The second part describes games, which embed coding as a mechanic. The player's comprehension of the embedded concepts increases by progressing through the game. The chapter concludes with a look at interesting educational AR projects.

Based on the research block, a mobile augmented reality game prototype was designed and developed. Chapter 4 describes the design process of the prototype. First, the functional and non-functional requirements are defined. Then a conceptual design and design decisions build the foundation for the development.

Chapter 5 presents the development process and implementation details. First, the development environment is explained, including the decision-making for a development framework. Then the architectural layout is discussed. Details of the individual level implementations conclude the chapter.

The last segment describes the evaluation and discusses the outcomes. Chapter 6 presents the design and the procedure of the evaluation with two different groups. The results are analyzed thoroughly and discussed according to the research objectives.

Chapter 7 revises what worked well throughout this thesis and what could be improved. The concluding chapter 8 sums up the outcome and provides an outlook for future work.

2. Background

This chapter covers relevant literature and background for this thesis. At first existing literature on computational thinking is reviewed. Then we look at the historical background and current topics on learning and teaching. Technology-enhanced learning and game-based learning leads us to the essentials of game development. The chapter concludes with a look at augmented reality from early experiments to modern applications in various fields.

2.1. Computational Thinking

Computer science is getting a lot of attention in today's education. *"All of today's students will go on to live a life heavily influenced by computing"* (Barr & Stephenson, 2011). Computational Thinking (CT), as a set of problem-solving skills, builds the foundation to understand complex processes and challenges. Wing (2008) notes that in computing abstraction into layers and *"well-defined interfaces between layers enable us to build large, complex systems"*. CT has become one of the key competences in the 21st century for young students to acquire. They learn how to solve complex problems by breaking them down into parts, which are easier to understand.

2.1.1. CT History

Researchers at the beginning of computer science understood that they had developed new ways of thinking. They foresaw the importance of computing for the future. Dijkstra (1979) for instance, describes three ways of thinking used when programming:

1. effective use of abstraction
2. design and use of notations
3. avoiding analyses with too much overhead

Seymour Papert dedicated his work to computing and education. He worked with Jean Piaget, who specialized in cognitive child development. Papert was strongly influenced by Piaget, considering children as active builders of their own thinking. Papert described computers as a powerful education tool with a more significant impact than books or videos. In 1967 he created Logo, which he describes as *"a programming language plus a philosophy of education"* (Papert, 1999). In Logo, children can program a turtle to draw patterns by defining movement sequences and iterations. He notes that students can learn *"mechanical thinking"*, which he describes as *"the art* *of deliberately thinking like a computer"* (Papert, 1980). Long after Papert's initial work on Logo, Wing (2006) attracted a lot of attention towards computational thinking in education. Much like Papert, Wing emphasizes that CT skills are essential for future working environments and that they are beneficial in all disciplines, not just computer science. *"Thinking like a computer scientist means more than being able to program a computer. It requires thinking at multiple levels of abstraction"* Wing explains. Her interpretation of computational thinking is not restricted to programming. Denning (2017) points out that *"[...] in Traditional CT programming ability produces CT, and in New CT learning certain concepts produces programming ability"*.

Due to its historical development, CT encompasses a wide variety of concepts. There are various definitions on which skills are embraced by CT. Barr and Stephenson (2011) describe CT as *"a problem solving methodology that can be automated and transferred and applied across subjects"*. Further, they specify the following core CT concepts: data collection, data analysis, data representation, decomposition, abstraction, algorithms, automation, parallelization and simulation. According to Wing (2006), CT relevant skills are conditional logic, distributed processing, debugging, simulation and algorithm building. Different interpretations and practical approaches make it a non-trivial task to measure CT learning progress. Gouws, Bradshaw, and Wentworth (2013) created the computational thinking framework to benchmark applications that are designed to teach CT skills. The framework distinguishes six distinct areas of CT:

- Processes and transformations
- Models and abstractions
- Patterns and algorithms
- Tools and resources
- Inference and logic
- Evaluations and improvements

Gouws et al. separate these areas in the understanding of a given problem from the active use of tools for solving it. As a result, the framework is able to analyze CT applications and provide a structured overview.

Other research focuses on the learner's perspective and how CT skills can be acquired. Barr and Stephenson (2011) note that *"Students become not merely tool users but tool builders. They use a set of concepts such as abstraction, recursion and iteration to process and analyse data"*. CT improves collaboration between humans and computesr. Problems are formulated *"in a way that enables us to use a computer and other tools to help solve them"* (Vallance, 2017). He identifies the characteristics of CT as organization and analysis of data, abstractions such as models, automation through algorithmic thinking, and generalization.

2.1.2. CT in Education

Since there are many different perspectives and interpretations of what CT embraces, it is difficult to find a universal definition that fits all possible scenarios. Denning

(2017) points out that there is little empirical evidence that computational thinking benefits everyone. He notes that too many different definitions cause inconsistencies in educational setups. Further Denning investigates the open issue of proper CT skill assessment. Nevertheless, most definitions overlap and the high interest in educational research indicates a strong demand for CT in education. With her article Wing (2006) initiated discussions all around the globe. The strong interest lead to implementations of CT in curricula such as K-12 in the US or in the UK national curriculum. In 2015 the British Computer Society published a CT guide for teachers (Csizmadia et al., 2015). They discuss the following CT concepts:

- Algorithmic thinking
- Decomposition
- Patterns
- Evaluation
- Abstractions and representations

The International Society for Technology in Education (ISTE, 2016) describes the following characteristics of CT:

- Abstraction
- Collecting data
- Identify relevant data
- Analyze data
- Represent data
- Decomposition
- Create automated solutions by using algorithmic thinking

Projects to encourage CT in schools

Besides national curricula, there are other efforts to increase CT activities in school. *"IT contests may be a key to the potential of new knowledge and an attractive way of binding up technology and education"* (Dagienė, 2006). In 2004 Dagienė established the first Bebras[1] challenge. The goal was to make computer science more attractive to students through a competition. Over time the competition evolved and many countries joined. In 2018 more than 2.780.000 participants from 54 countries participated in the contest[2].

Code.org[3] is a US-based non-profit organization that aims to improve the accessibility of computer science for all students. In 2013 Code.org launched the initiative *Hour of Code* (Partovi & Sahami, 2013). The goal of this initiative is to engage students to acquire programming skills and to motivate schools to include more programming activities in their curricula.

Initiatives like *Hour of Code* and *Bebras* encourage CT education in schools. Clearly, teachers must be involved in the process. Lack of CT knowledge must be trained in

[1]bebras.org https://www.bebras.org/, accessed 2019-07-17
[2]bebras statistics https://www.bebras.org/?q=statistics, accessed 2019-07-17
[3]Code.org https://code.org/, accessed 2018-08-28

order to include CT in education successfully. *"As computational thinking is central to the syllabus, it is important that teachers both have a deep understanding of it and have ways to help students develop the skills"* (Curzon, McOwan, Plant, & Meagher, 2014). They implemented unplugged interactive workshops to introduce CT concepts to teachers and improve their CT knowledge. A different more hands-on approach was shown by Marcelino, Pessoa, Vieira, Salvador, and Mendes (2018). They created an online course to improve CT knowledge of elementary school teachers by programming Scratch.

2.2. Learning and Teaching

Coon and Mitterer (2004) define learning as *"a relatively permanent change in behavior due to past experience"*. Different models exist, which describe how learners receive, process and store information. With advancing technology the possibilities of transporting learning content increase. How can these new possibilities be used to motivate and engage learners?

2.2.1. Learning Preferences

In the past decades, research in educational sciences has shown various models of learning styles with different perspectives. The categorizations are based on the diverse ways of perceiving and processing information and the learning preferences of the individual.

However, there is little to no empirical evidence supporting learning style hypothesis. Coffield, Moseley, Hall, and Ecclestone (2004) reviewed 13 of the most influential learning style models. They found shortcomings in validity, consistency or reliability in nearly all of them. The sheer number of 30 different learning styles that were discussed suggest that categorizing an individual in one specific style might not be optimal. Kirschner and van Merriënboer (2013) state that *"learning styles poorly classify learners"*. Kirschner (2017) criticizes the way how students are categorized into groups. He questions the validity and reliability of learning type tests, which are mostly self-report assessments. *"The self-reported preferred way of learning is often a bad predictor of the way people learn most effectively"*. Other research criticizes that learning styles are being implemented in education without empirical proof. *"Students may have preferences about how to learn, but no evidence suggests that catering to those preferences will lead to better learning"* (Riener & Willingham, 2010). Despite the legitimate critics, learning style models contribute to relevant concepts on learning preferences. In the following three relevant models to this work are described shortly.

Kolb's model

According to Kolb (1984, p. 38) *"Learning is the process whereby knowledge is created through the transformation of experience"*. Kolb based his learning style theory on a learning cycle model with four stages.

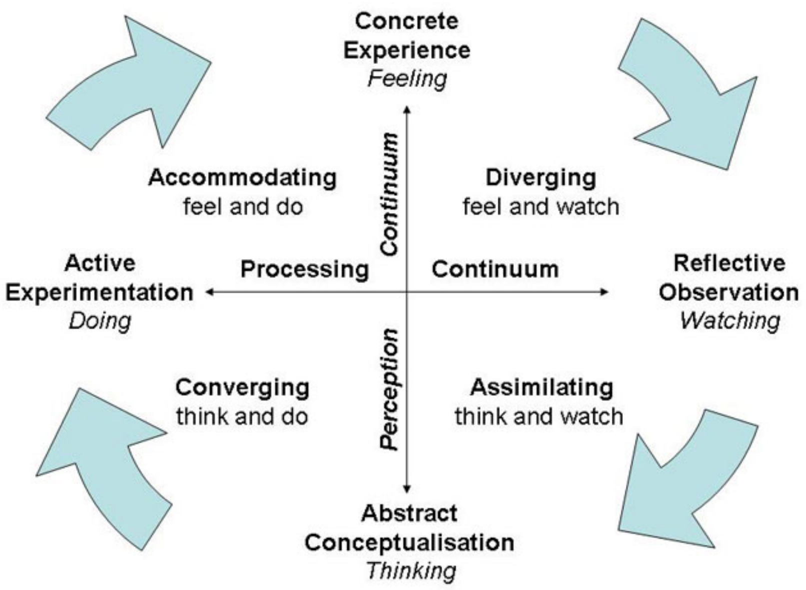


Figure 2.1.: Visualization of learning styles according to Kolb (McLeod, 2017)[4].

1. Concrete Experience
2. Reflective Observation
3. Abstract Conceptualisation
4. Active Experimentation

Kolb's model states that actual learning happens when these four stages are followed in a logical order. Kolb claims that each individual prefers one out of four different learning styles: Accommodating, Diverging, Converging and Assimilating. These preferences are a combination of the learning stages (as shown in Figure 2.1).

VARK

According to Fleming (2011) there are four different learning types and learners have different preferences among these. Based on these definition Fleming created the VARK model. VARK is an acronym for four learning preferences: visual, auditive, read/write, and kinesthetic. While visual learners prefer visual learning content (e.g. pictures, diagrams), auditory learners learn more effectively with sounds, read/write with written information and kinesthetic learners by experiencing. Fleming considers learners with multiple preferences as *multimodal* learners.

[4]Retrieved June 28, 2018 from www.simplypsychology.org/learning-kolb.html

Felder–Silverman Learning/Teaching Style Model

Felder, Silverman, et al. (1988) found shortcomings, particularly in engineering education. They note that *"mismatches exist between common learning styles of engineering students and traditional teaching styles of engineering professors. In consequence, students become bored and inattentive in class, do poorly on tests, get discouraged about the courses, the curriculum, and themselves, and in some cases change to other curricula or drop out of school"*.

They further propose to adapt the teaching style in a way, such that all learning preferences could be satisfied. In their model, they define what they call "preferred learning styles" and the corresponding teaching styles which should be used to overcome the mismatches. For instance, for learners who prefer visual input, a visual presentation is recommended. The model is based on earlier research such as Jung's theory of psychological types, the Myer-Briggs Type Indicator and the VARK model. It combines research results on how learners perceive and process information differently, with a focus on engineering education. Based on their model the online questionnaire *Index of Learning Styles*[5] was created for self-assessment of learning preferences.

2.2.2. Technology-Enhanced Learning

Educational research has found various ways to use technology as a beneficial complement to traditional teaching. As technology keeps changing, there are new possibilities appearing every day. The term technology-enhanced learning (TEL) is used to describe the application of information and communication technologies for teaching and learning (Kirkwood & Price, 2014). This definition includes a wide range of technologies. *"It is not simple to define which technologies are 'learning technologies' or 'e-learning'"* (Dror, 2008). The applications of TEL range from simply using computers to acquire information to interactive whiteboards to digital game-based learning to a mobile technology-enhanced flipped classroom (Hwang, Lai, & Wang, 2015) and much more. Universities such as MIT or Harvard offer so-called *Massive Open Online Courses* (MOOCs) for everyone. *"In most cases participants sign up for MOOCs free of charge and in some cases for a small or minimal fee to obtain a completion certificate"* (Hew & Cheung, 2014). In this section, the most relevant aspects of TEL for this work are discussed more in detail.

Mobile Learning

Mobile technology advanced a lot in the last decades. Most people who own a smartphone have a computer in their pocket with more computing power than the computers in the Apollo 11 mission ship, which carried the first humans to the moon. One of the relevant aspects of this thesis is the inclusion of mobile learning in school education. First, there is a big choice of low-cost devices, which are affordable for schools. Secondly, many school students have their own mobile devices. With a focus on smart

[5]ILS https://www.webtools.ncsu.edu/learningstyles/, accessed 2018-08-28

mobile devices, mobile learning offers improved usability and increased simplicity compared to desktop applications. Berge and Muilenburg (2013) investigate the impact of the adaptivity of mobile technologies. They note that this development leads to new learning situations. The app stores (e.g. iTunes, Google Play, Microsoft Store) are accessible for everyone with a matching device. The stores allow developers to publish their apps with a low threshold and reach a large group of potential users. There exist apps for nearly every field of interest. Learning apps such as Duolingo[6] use the benefits of mobile devices (Finardi, Leao, & Amorim, 2016). The devices are used frequently within short time frames. To use these short time frames in a convenient way, learning apps usually implement small learning blocks and short feedback cycles. Mobile learning is of relevance and renders important need because access to mobile technologies is widespread. The number of smartphone users worldwide is tending towards 3 billion in the near future (eMarketer, 2015). Together with laptops, tablets and other devices the number of actively used mobile devices is much higher. Projects like One Laptop Per Child (Negroponte, Bender, Battro, & Cavallo, 2006) aim to increase this access especially in areas where people can't afford costly hardware. Given the access to a device and internet, users have the possibility to learn autonomously with open resources such as MOOCS.

Gamification

Game designers are pioneers when it comes to creative use of technology and innovative mechanics. These ideas can be adapted to other areas, which is often described as gamification. A widely used definition describes gamification as *"the use of game design elements in non-game contexts"* (Deterding, Dixon, Khaled, & Nacke, 2011). They further note that because of their subjective nature sometimes games and gamified systems can be difficult to distinguish. Relevant game design elements can be for instance, points, badges, leaderboards, progress or rewards. Gamification can be implemented in all kinds of non-game contexts. Deterding et al. suggest not to limit gamification to specific contexts. They note that even games can be gamified e.g. with achievements (as long as it is not part of the game design). Seaborn and Fels (2015) analyze existing research on gamification and summarize the diverse range of results. Their survey shows that *"Gamification has been applied and researched across many domains, from sustainability to health and wellness to education"*. According to their findings the top field of research on gamification is education. Educational research is particularly interested in whether gamification can increase student motivation and engagement.

In an approach to gamify a university course Pirker, Riffnaller-Schiefer, and Gütl (2014) introduced the pedagogical model Motivational Active Learning (MAL). MAL combines Technology-Enabled Active Learning (TEAL[7]) with motivational game design elements and included interactive collaborative tasks. MAL was included in the design of a Computer Science course at Graz University of Technology. It was evaluated

[6]Duolingo https://duolingo.com/, accessed 2018-04-17
[7]TEAL http://icampus.mit.edu/projects/TEAL.shtml, accessed 2019-03-10

through pre- and post-questionnaires and a transparent intermediate grading point system. Pirker et al. (2014) found that students felt motivated through immediate feedback in form of (grading) points, the interactivity, collaborative tasks and the progress through the adaptive course design.

Other research suggests that specific game design elements (e.g. competition, leaderboard, badges) might be harmful for educational outcomes (Domínguez et al., 2013; Hanus & Fox, 2015). Existing research results differ strongly. Depending on the context, the design and the particular focus they are mostly not comparable. As Hanus and Fox pointed out, future studies should look at specific elements only, in order to have a more meaningful and comparable result. For the same reason, they suggest to take into consideration empirical results and concerns before implementing gamification in an educational environment.

2.2.3. Game-Based Learning (GBL)

Games have been part of human cultures all over the world for thousands of years. One of the oldest known board games is called Senet (Piccione, 1980) and findings show that it was played in ancient Egypt around 3100 B.C. Various different definitions of the term game exist in literature and it depends on the context, which one suits best. Koster (2013) describes games in his attempt to find a universal definition as *"exceptionally tasty patterns for the brain to eat up"*. An extended definition is given in the classic game model by Juul (2003): *"A game is a rule-based formal system with a variable and quantifiable outcome, where different outcomes are assigned different values, the player exerts effort in order to influence the outcome, the player feels attached to the outcome, and the consequences of the activity are optional and negotiable"*.

Playing a game means understanding the rules and mechanics of the game and how to use them. These rules and mechanics must be learned. Learning is an essential part of games. Game-based learning (GBL) builds upon the idea, that the process of learning is more engaging and effective within a playful environment. Plass, Homer, and Kinzer (2015) argue, that GBL differs from gamification by design. While gamification only adds game design elements on top, GBL incorporates the learning subject into the game *"with the desire to prioritize game play"*. Real-world applications indicate a rising interest in GBL in educational settings. In 2009 *Quest to learn* has been established in New York (Salen, Torres, Wolozin, Rufo-Tepper, & Shapiro, 2010). *Quest to learn* is a public school with a curriculum built on game-based learning. This school for children between 6 and 12 follows an innovative approach aiming to change education through play. The students become more engaged through collaborative games, which allow them to fail and reiterate. The curriculum deliberately embraces a "learning by doing" mentality. The students acquire problem-solving skills, which can be useful for their future education and career.

Tech companies are certainly interested in technology affine future generations as employees, customers or developers for their devices and platforms. For instance, Apple offers the summer holiday Apple Camp[8] to engage children between 8 and 12 years with their technologies. In a playful environment the participating children learn how to create animations and games or how to program a robot to make autonomous decisions.

Digital game-based learning combines the benefits of game-based learning with technology-enhanced learning. Video games can create a playful environment with game elements and mechanics that wouldn't be possible without recent technology. According to Prensky (2003) *"Video games are not the enemy, but the best opportunity we have to engage our kids in real learning"*. Many diverse factors make games entertaining and engage the players to explore and play around. Some factors can be used to categorize games in genres. Game genres classify groups of games with similar gameplay elements and mechanics. Players often have a favorite genre, in which they have made good experiences. Based on these experiences, a player might be more or less engaged in a certain genre. The list of different game genres and subgenres is long: action, simulation, strategy, role-playing, and puzzle just to name a few. The genre is a core part of a game that arises from very early design decisions. Laporte, Zaman, and De Grooff (2013) have examined a small set of serious games and the relevance of game genres for learning success. They found that the linkage between the learning content and the genre provides a base for a successful learning process. If the genre of a serious game is not suitable for delivering the content, additional elements (e.g. textual information) have to be introduced, which might decline the playing experience.

Mobile Game-Based Learning

With the rise in popularity of mobile devices, digital game-based learning is not limited anymore to PCs and laptops. Quite the opposite, mobile devices offer a whole set of additional possibilities. Mobile games can be played anywhere and can use location data (GPS, network) as a game element. Built-in front & back cameras allow the integration of a real-world environment as well as face tracking or similar. Through Wifi, Bluetooth and NFC the devices can communicate with each other and offer seamless multiplayer experiences or retrieve additional information. Given an internet connection, servers can be used e.g. for location-independent multiplayer sessions, to load additional content dynamically or share created content. These possibilities can be used to enhance the learning experience and further motivate the player to explore and learn in new ways. Huang, Chang, and Wu (2017) analyzed how mobile game-based learning influences motivation and performance through a language learning game. They concluded that the students were satisfied and had fun. However the correlation between time played and learning progress was moderate. In a critical work Giannakas, Kambourakis, Papasalouros, and Gritzalis (2018) analyze the evolution of mobile game-based learning from 2004 to 2016 and discuss emerging difficulties and challenges. They conclude that *"traditional GBL undergoes a rapid shift to mobile*

[8]Apple Camp https://www.apple.com/today/camp/, accessed 2019-03-03

platforms with the aim not only to move learning outside the classroom and take advantage of the anywhere, and anytime experience, but also to transform radical the learning experience." When creating a learning game, it is important to consider all these factors during design and development.

2.3. Game Development

Creating a game with educational purposes requires an elaborated development process. The complexity of creating video games has increased over the years. *"Games are hard"* (Blow, 2004). Blow describes the situation in the game development industry in 2004. One of the difficulties he describes is the lack of development tools, which are designed specifically for game development. Fortunately, this situation has changed since 2004.

Recent frameworks and engines simplify the game development process by providing tools such as physics, 3D rendering, animation functionality, ready to use assets and much more. Most frameworks are able to build binaries for multiple platforms (cross-platform). This minimizes time spent on system-specific and redundant tasks. There is a variety of frameworks and engines with different advantages (see Table 2.1). The game platform itch.io[9] offers some insights on which engines are used for the published projects. Most projects on the platform use Unity. Other projects use different frameworks such as Gamemaker Studio, Unreal Engine, Godot, LibGDX, etc. A similar trend can be observed on the Steam platform[10] with Unreal Engine and Unity as leading development frameworks.

Unreal Engine offers stunning visualizations and a visual scripting system called Blueprints. The engine offers full source code access and is free to use, only commercial users pay a license fee. Unity follows a similar approach, where commercial products require a paid subscription. One of Unity's core concepts are customizable, reusable game objects called prefabs. Unity partnered with Vuforia Engine[11] to offer advanced AR capabilities within the framework. Vuforia features AR based on markers, images, 3D models and objects. The large community, the available content through the asset store and well documented functionality are a few of the upsides of Unity.

There are also open-source frameworks available such as Godot[12] and LibGDX[13]. They offer a lot of useful functionality, although e.g. AR support is still under development. Nevertheless, a growing audience for these frameworks shows the interest in open-source in the game development community.

Mobile games can also be developed exclusively for the respective system. The most used mobile operating systems Android and iOS offer frameworks to do so. Using this native approach allows efficient use of system resources. On the other hand, these

[9]itch.io https://itch.io/game-development/engines/most-projects, accessed 2019-13-05
[10]Steam https://store.steampowered.com/, accessed 2019-13-05
[11]Vuforia https://engine.vuforia.com, accessed 2019-13-07
[12]Godot https://godotengine.org/, accessed 2019-13-07
[13]LibGDX https://libgdx.badlogicgames.com/, accessed 2019-13-07

Table 2.1.: Comparison of Game Development Frameworks and Engines

	iOS	Android	Unity	Godot	Unreal	libGDX
Mobile Devices	✓	✓	✓	✓	✓	✓
3D Graphics	✓	✓	✓	✓	✓	✓
Augmented Reality	✓	✓	✓	work in progress	✓	with plugins
Cross-Platform	✗	✗	✓	✓	✓	✓
Asset Store	✗	✗	✓	✓	✓	✗
Open-Source	✗	✓	✗*	✓	✗*	✓

* Full source code access

frameworks are not designed for game development specifically. Developing a game natively can include a lot of system-specific work and can cause overhead when porting the game to another system.

The development of augmented reality applications is supported through a range of dedicated AR SDKs. Depending on the target platform and usage scenario they offer different advantages. Especially for mobile platforms it is important to use resources efficiently. This is why Apple and Google offer their own AR SDKs. Since iOS 11 Apple provides ARKit[14] for iOS developers. At the same time Google announced its own SDK ARCore[15]. Unlike ARKit, ARCore supports both Android and iOS. However, the before-mentioned Vuforia Engine supports a bigger range of devices, including older Android and Apple mobile devices.

Progress and changes in computer hardware and software enable innovative ideas to grow. In the last decade these advances empowered the creation of immersive interactive experiences in real and virtual environments. In the following section possible application areas are discussed.

2.4. Augmented Reality

AR is a field with vast potential in a world of advancing mobile technology. The technology *"allows the real time blending of the digital information processed by a computer with information coming from the real world by means of suitable computer interfaces. Augmented reality is comprehensive information technology that combines digital image processing, computer graphics, artificial intelligence, multimedia technology and other areas"* (Amin & Govilkar, 2015). Diverse AR projects have shown that the setup may contain various combinations of devices and representations, depending on the resources and the given environment. Early AR setups used specifically designed systems for very specific purposes. With wide-spread hardware such as beamers, cameras and immersive input devices like Microsoft Kinect (Vera, Gimeno, Coma, & Fernández, 2011) or the Nintendo Wii controller, setups became cheaper and more adaptable to different scenarios.

[14]ARKit https://developer.apple.com/arkit/, accessed 2018-04-17
[15]ARCore https://developers.google.com/ar/, accessed 2018-04-17

Nowadays, mobile devices contain high-resolution cameras and advanced hardware that allows fast data processing. *"The combination of enhanced processing and super dense storage capacity will enable a new generation of computational photography and augmented reality applications"* (Islam & Want, 2014). *Pervasive sensors in modern smartphones could help accurately measure our fine-scale movements such as sitting, eating or talking, which could be integrated into increasingly rich AR worlds* (Shea et al., 2017).

AR software uses the device camera to detect features for 3-dimensional positioning. These features can be connected to a preprocessed marker such as a QR code, a picture or a 3-dimensional model. Alternatively features can also be detected dynamically in the real environment. This so-called marker-less augmented reality requires more resources, as the features are not preprocessed and additional sensors are needed to locate the features correctly. Then the software renders 3D or other media objects such as videos or pictures onto the real-world position inside the screen. By moving the camera, one can view the projected content from different angles. Even if the markers disappear (out of sight), new technologies keep track of the content's positions, by using other surrounding features as anchors.

There are numerous different fields of application for AR. In an industrial or production environment, AR can be used for augmented maintenance or for training purposes. In 2014 the Swedish furniture company IKEA published an AR app called IKEA Place. With the app users can project IKEA furniture into their homes and see how the furniture fits in. So-called *virtual try-on* applications use face- and body-tracking e.g. to overlay different lipstick colors or cloths (Javornik, Rogers, Moutinho, & Freeman, 2016). Other applications enrich the natural environment with informational data such as names, distances or heights, such as the mountaineering app PeakFinder[16].

Digital games also use the possibilities of AR to connect to the real-world environment, hence extending the virtual playground. At present Pokemon GO (Niantic, 2016) is one of the most successful AR games with more than 65 million active users per month. In this game, the player has to move physically to a specific location in order to catch virtual creatures. These 3-dimensional creatures are placed in the real-world by using the device camera and display. Through its demand of movement, Pokemon GO is considered to improve health and fitness (LeBlanc & Chaput, 2017). Wizards Unite (Niantic, 2019) a more recent AR game by Niantic builds on previous success and follows the popularity of Pokemon GO.

Current projects like Google Glass[17] and Microsoft Hololens[18] follow a different approach. Instead of smartphones and tablets, head-worn devices, so called smart glasses project content in front of the eye. *"Augmented Reality Smart Glasses are defined as wearable Augmented Reality (AR) devices that are worn like regular glasses and merge virtual information with physical information in a user's view field"* (Rauschnabel, Brem, & Ro, 2015). Smart glasses can be used in various settings, for instance, in clinical (Mitrasinovic et al., 2015) or industrial applications (Niemöller et al., 2017).

[16]PeakFinder https://www.peakfinder.org/de/mobile/, accessed 2019-03-03
[17]Google Glass https://x.company/glass, accessed 2019-03-02
[18]Microsoft Hololens https://www.microsoft.com/de-AT/hololens, accessed 2019-03-02

AR technologies can enhance the user experience in educational settings as well. A widespread technology in classrooms is the smartboard or interactive whiteboard (IWB). *"The basic IWB system comprises a computer linked to a data projector and a large touch-sensitive wall-mounted electronic board which displays projected images ("objects") that can be manipulated directly by hand or with a stylus. The IWB allows direct interaction with text and images on the screen, as well as access to previously stored material and the Internet"* (Kershner, Mercer, Warwick, & Staarman, 2010). They conclude that *"IWBs can make some identifiable contributions to children's productive communication and thinking, while also providing both a tool and environment that encourages co-constructed knowledge building"*.

Several projects have explored the possibilities of mobile device based AR for educational purposes. Chen, Ho, and Lin (2015) study the usability of an AR game-based learning system about marine wildlife. The player has to complete tasks to advance and thereby learns about the marine wildlife and the food chain. Zarzuela, Pernas, Martinez, Ortega, and Rodriguez (2013) point out the advantages of using AR compared to VR. They created an AR interactive zoo to support learning for children and disabled people. AR offers the user additional information and freedom to explore. This makes AR interesting in an educational setup. In their literature review, Nincarean, Alia, Halim, and Rahman (2013) highlight the potential benefits of mobile AR for education. They advise, to consider learning theory in the design and development of AR applications for educational purposes. Akçayır and Akçayır (2017) provide an systematic overview on AR for education and found that the number of studies in this field increased recently. They conclude that some reviewed studies show conflicting results. Regardless most of the studies suggest that AR can be beneficial for learning, when challenges are overcome.

Virtual Reality

"The idea is simple, everything we do to educate with words and with pictures can be provided as virtual experience" (Bricken, 1990). Bricken outlines the following educational use cases of virtual reality (VR): Individualized instruction, intelligent training and what-if scenarios. Since then VR has changed a lot and is used in a variety of settings. Recent devices such as Oculus Rift[19] or HTC Vive[20] allow the user to immerse in a virtual environment. VR technology has reached a broad audience and meanwhile devices are designed for private entertainment. Online video platforms such as Youtube[21] offer 360° videos. Simulations and video games are constantly introducing new features to enrich the virtual experience. Diverse cross-platform frameworks (e.g. Unity3D) facilitate the implementation of VR applications for different devices.

Through advances in technology and cheaper hardware virtual reality is becoming interesting for educational settings. Thus there is an increased interest in recent research to investigate the educational use of VR. The games research group at Graz

[19]Oculus Rift https://www.oculus.com/rift, accessed 2019-07-29
[20]HTC Vive https://www.vive.com, accessed 2019-07-29
[21]Youtube https://www.youtube.com/, accessed 2019-07-31

University of Technology is continuously extending the virtual learning environment Maroon VR (Pirker, Lesjak, & Guetl, 2017). Maroon VR allows the user to explore and conduct physical experiments. The virtual environment and the visualization offer interaction that wouldn't be possible in a real experiment. Both AR and VR are promising approaches to increase engagement and facilitate understanding.

2.5. Summary

A growing interest in CT education indicates the relevance now and for future generations. In this chapter related background for this thesis is discussed. CT definitions in research vary. However, CT is being integrated into national curricula worldwide. For a proper integration of CT in educational settings, different learning preferences and appropriate instructional methods should be considered. Advancing technology can be used to visualize information, make the learning experience interactive and allow digital collaborative learning. With a growing number of mobile devices, particularly mobile learning gets more significance. Augmented reality (AR) as an emerging technology enables new interesting possibilities, particularly on modern mobile devices with increasing computation capabilities. Educational research strives to increase learner engagement and motivation through the use of game design elements. These concepts form the necessary background for designing and developing an AR mobile game-based approach to teach CT skills.

3. Related Work

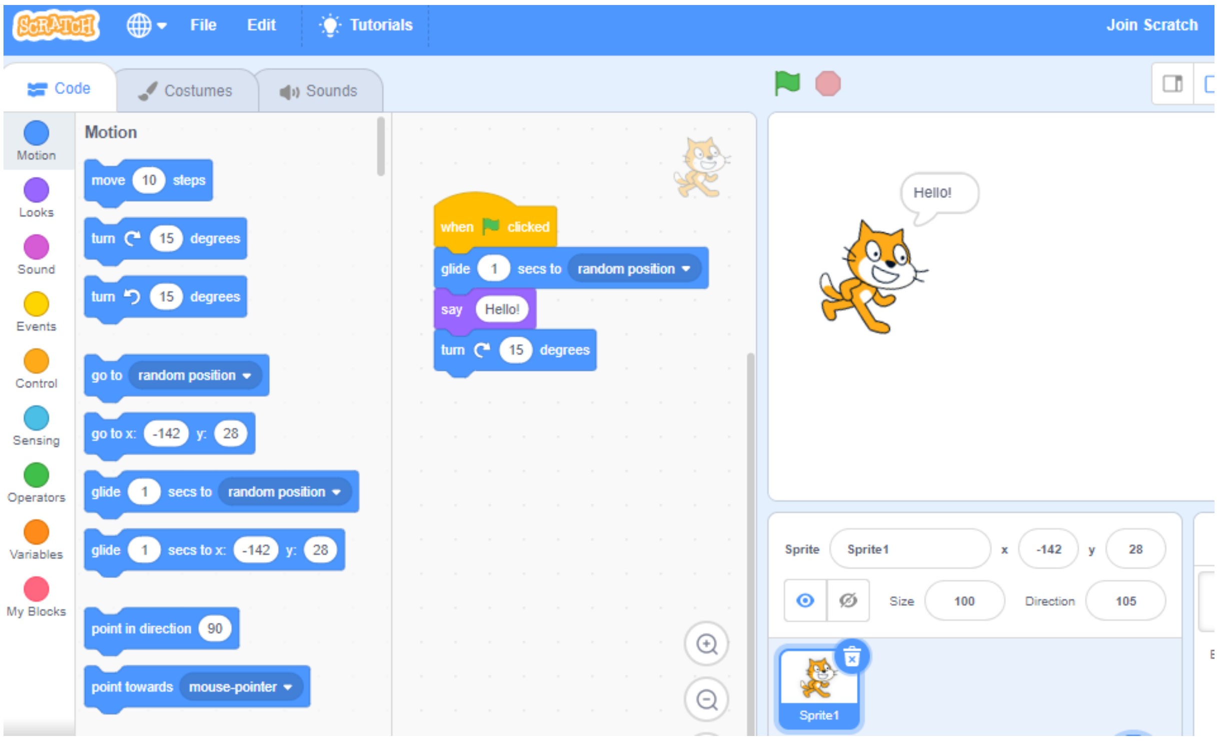
The rising interest in CT and initiatives such as *Hour of Code* (see Section 2.1) motivated the creation of more CT related projects. CT content is embedded in various games and research projects. For teaching CT there are many diverse interesting aspects including tangible interfaces, robotic frameworks, block-based programming, playful and explorable environments. The applications differ most in which specific skills should be learned and how. Considering the core idea, most CT learning applications can be grouped into two categories: Learn Through Code and Learn Through Play. While applications from the former category emphasize on the coding activity, the latter primarily embeds learning content in gameplay. This chapter covers related work in these two categories and other fields.

3.1. Learn Through Code

Projects in this category focus on the activity of coding. Either in a specific programming language or in some pseudo code with a simplified syntax. One popular approach to teach CT skills that has been explored and developed over the last years is block-based programming (BBP). The idea is to teach central programming concepts and give freedom to explore, rather than focusing on the syntax of a specific programming language. *"Visual programming languages use representation that is closer to human language"* (Lye & Koh, 2014). Block-based programming frameworks offer ready to use blocks to build a program by drag and drop. The blocks snap on other blocks, creating a Lego-like structure and encouraging a constructivist process. Rose (2016) studies the impact of the so-called bricolage programming with elementary school students. The visual aid in block-based programming is helpful to an intuitive understanding of complex structures such as loops or functions. In the following some widespread BBP projects and programming-learning tools are shown.

Scratch is an open-source project emphasizing on CT. It was developed by Mitchel Resnick with the MIT media lab in 2003. Scratch has been constantly improved since then by the Lifelong Kindergarten research group at MIT (see Figure 3.1). Scratch is *"[...] a programming environment that enables young people to create their own interactive stories, games, and simulations, and then share those creations in an online community with other young programmers from around the world"* (Brennan & Resnick, 2012). The Lifelong Kindergarten research group worked together with the Danish toy company Lego[1] to connect block-based programming with tangible Lego blocks and robotic functionality

[1]Lego Mindstorms https://www.lego.com/en-us/mindstorms, accessed 07-28-2019



```
when [flag] clicked
glide 1 secs to random position
say Hello!
turn 15 degrees
```

Figure 3.1.: Screenshot of the Scratch framework.

(Resnick et al., 2009). The robots are based on the Lego Mindstorms NXT computer and can be extended using a variety of Lego blocks. The Lego EV3 programming interface (see Figure 3.2) allows the user to reuse and modify programs created in EV3. These programs can be transfered onto Lego Mindstorm robots, which execute the programmed behavior.

Another widely used resource for BBP is Blockly. Blockly is an open-source library by Google to support the creation of block-based programming environments. MIT's Scratch team are developing a Blockly Fork called Scratch Blocks[2] where they combine textual and symbol-based BBP.

Inspired by Scratch a research group at Graz University of Technology has developed the app Pocketcode[3] to bring BBP onto mobile devices (Slany, 2012). The goal is to empower young students in self-controlled creation of animations, games and other programs (see Figure 3.3).

Online programming environments such as freecodecamp.org, codeacademy.com or repl.it provide a simple way to start learning a specific programming language. While learning through tutorials, the user can test and execute code directly in the browser, without any need of downloading, installing and configuring compilers, editors, etc. These sites often use gamification elements to keep users engaged and reward progress.

[2]Scratch Blocks https://github.com/LLK/scratch-blocks, accessed 03-19-2018
[3]Catrobat https://www.catrobat.org/#pocketcode, accessed 07-19-2019

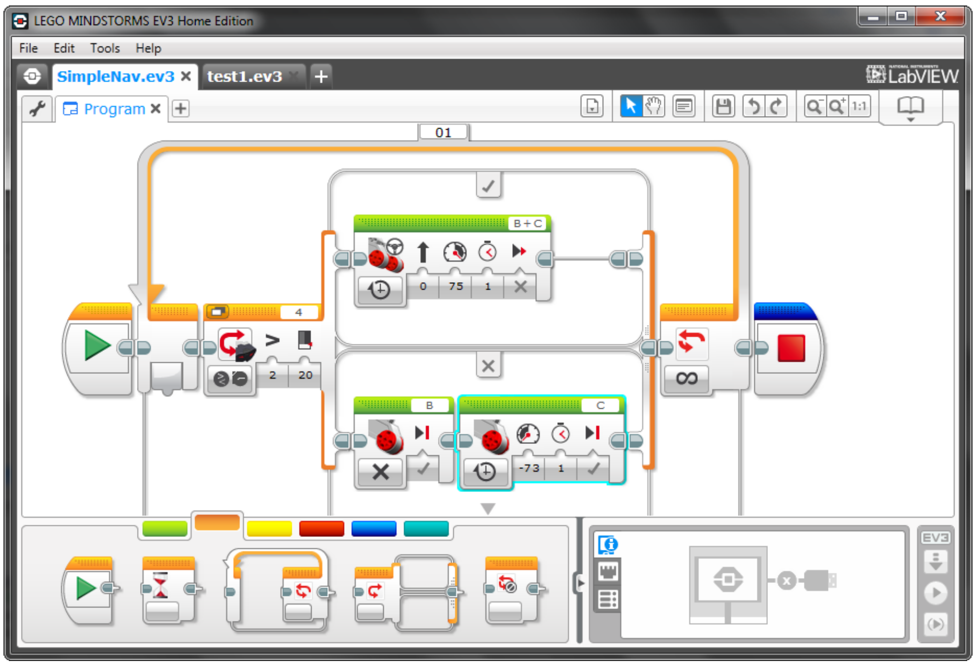


Figure 3.2.: Screenshot of the Lego Mindstorm EV3 block-based programming interface.

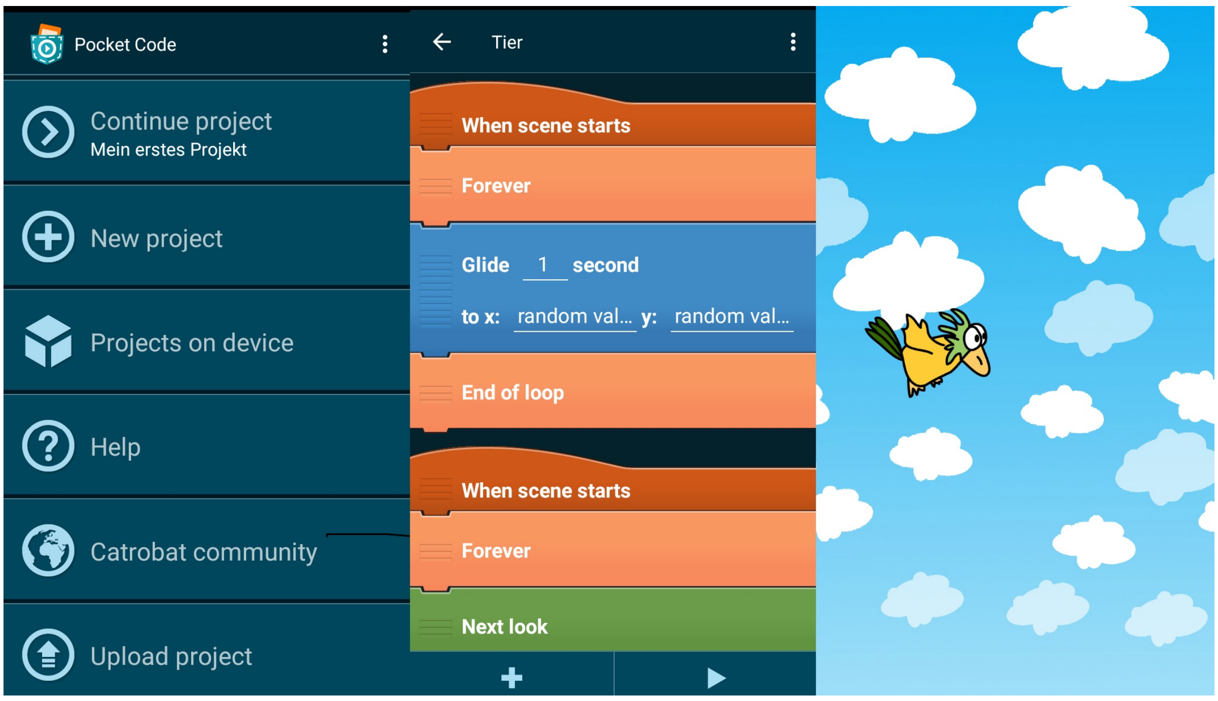


Figure 3.3.: Screenshot of mobile block-based programming in Pocketcode.

3.2. Learn Through Play

Before writing code, the learner has to understand the basic concepts of programming. Therefore *Learn Through Play* embeds these concepts in a playful environment. Instead of directly learning to program, the learner/player has to achieve clear goals by using and improving code elements. Through increasing difficulty throughout the gameplay and the introduction of new concepts, players practice and improve their CT skills. There are numerous games on different platforms that aim to teach CT. These games have diverse contents, goals, and visualizations. Block-based programming (as described in 3.1) is used as a control mechanic for most of the games.

Bauer, Butler, and Popović (2015) developed Dragon Architect, a block-based programming game to teach basic programming concepts such as variables, conditionals, and loops. According to their research, a Minecraft-like 3D environment (see Figure 3.4) engages young players to explore and experiment. Googles block-based programming tool *Blockly* is used as a programming interface for the player to program the actions of a 3D dragon character.

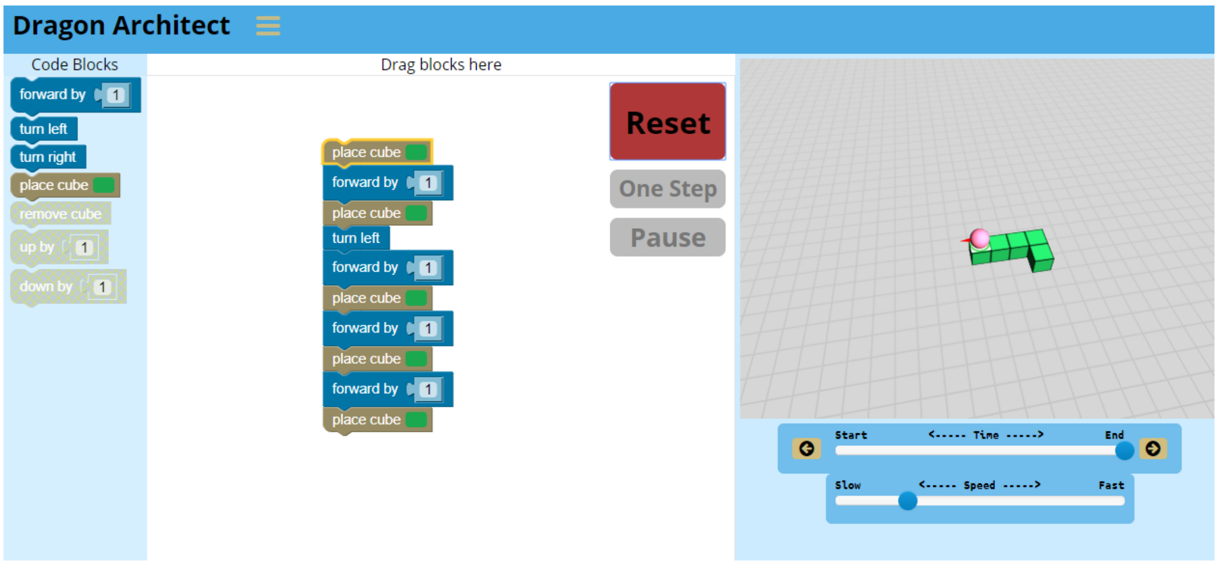


Figure 3.4.: Screenshot of a 3D construction through block-based programming in Dragon Architect (Bauer, Butler, & Popović, 2015).

Dragon Architect focuses on Divide & Conquer where the player either deconstructs a given structure (top-down) or creates one (bottom-up). The paper proposes the use of in-game metrics (e.g. used time) for evaluating the learning performance and outcomes. Additionally, on-paper assessments (pre-test and post-test) are considered to gain better insight on the learning outcomes. Kazimoglu, Kiernan, Bacon, and MacKinnon (2012) state that *"it is crucial for students to develop skills in CT before they are introduced to formal programming"*. Following a *learn through play* approach, they developed the educational game framework *Program your robot*. In the game the player programs a little robot to find a path to the teleporter in an isometric cartoonish environment. In the user interface the robot program is represented through a panel that can be filled with command blocks. The command blocks are separated in *action* (i.e. movement) and in *programming* (i.e. function and loop calls) commands to give a better understanding of the underlying programming concepts. When the player has dragged these commands into the program panel the interface offers three options: *Run* to execute the command blocks in the program panel, a *debug* mode which includes helpful textual feedback and *clear* to reset the program panel. A scoring system is applied to encourage the player to create cleaner solutions and repeatable patterns, which use fewer blocks. Feedback from a test evaluation with computer science students included the wish for more achievements and rewards to increase motivation.

Yaroslavski (2014) summarizes important programming practices as: planning, programming, testing and debugging. The game Lightbot (Yaroslavski, 2008) teaches the player CT skills by playing through these practices. The goal in each level is to program the robot to light up all blue tiles (see Figure 3.5). The player plans the actions of the robot by dragging command blocks from a toolbox into a program sequence. At execution, the robot processes these movement commands sequentially. Visual and auditive feedback provide additional debugging information. Levels can be restarted at any point e.g. to correct the code. In later levels procedures (i.e. functions) and loops (i.e. recursions) are introduced, such that the player learns to identify patterns and reuse block sequences. Meanwhile different versions of the game exist. The latest Lightbot version includes conditionals, is optimized for mobile platforms and includes user-friendly enhancements. The experience from Lightbot was then transferred to the platformer game Spritebox (Yaroslavski, 2017). Thus with a well-thought game design CT concepts and presumably others as well, can be included in various game genres. As an initial test of the CTF, Gouws et al. (2013) analyzed and evaluated the original version of Lightbot. With an overall score of 74% they describe Light-Bot as a *"useful game for practicing CT"* with strengths and weaknesses. One particular point to be improved is the confusion of the terms recursion and loop. The term loop is introduced, while in fact the player learns to implement recursions.

There are numerous games similar to Lightbot and Program Your Robot with different focuses, mechanics and graphical demands. These games differ mostly in the graphical representation and the user interface. Some use simplified interfaces, while others use Blockly, Scratch or a simple script language. The mobile STEM learning environment sCool (Kojic et al., 2018) provides students with a game environment and educators with a tool to analyze the students use of the system. In the game part sCool uses a mixed approach of Python code and predefined code blocks, which can be dragged into the code (see Figure 3.6). By programming a robot to find a path to a certain goal, the player gets to know basic scripting, debugging, the concept of variables, conditionals and loops. In the award-winning Box Island (Radiant Games, 2017) the player programs the movement of the box-shaped character through an exotic 3D scenery. The colorful world and lively animations engage especially young players to play and learn. The programming is done in a block-based programming (see Section 3.1) environment. Box Island, as well as Lightbot, support the hour of

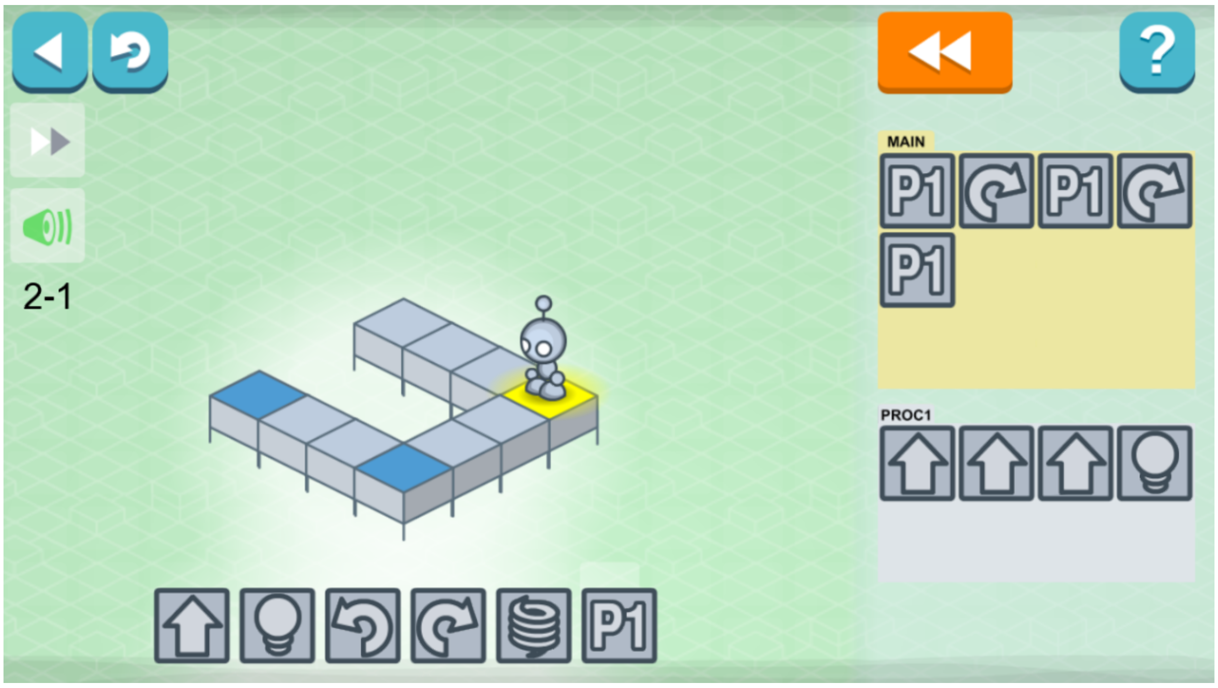


Figure 3.5.: Screenshot of the game Lightbot (Yaroslavski, 2014).

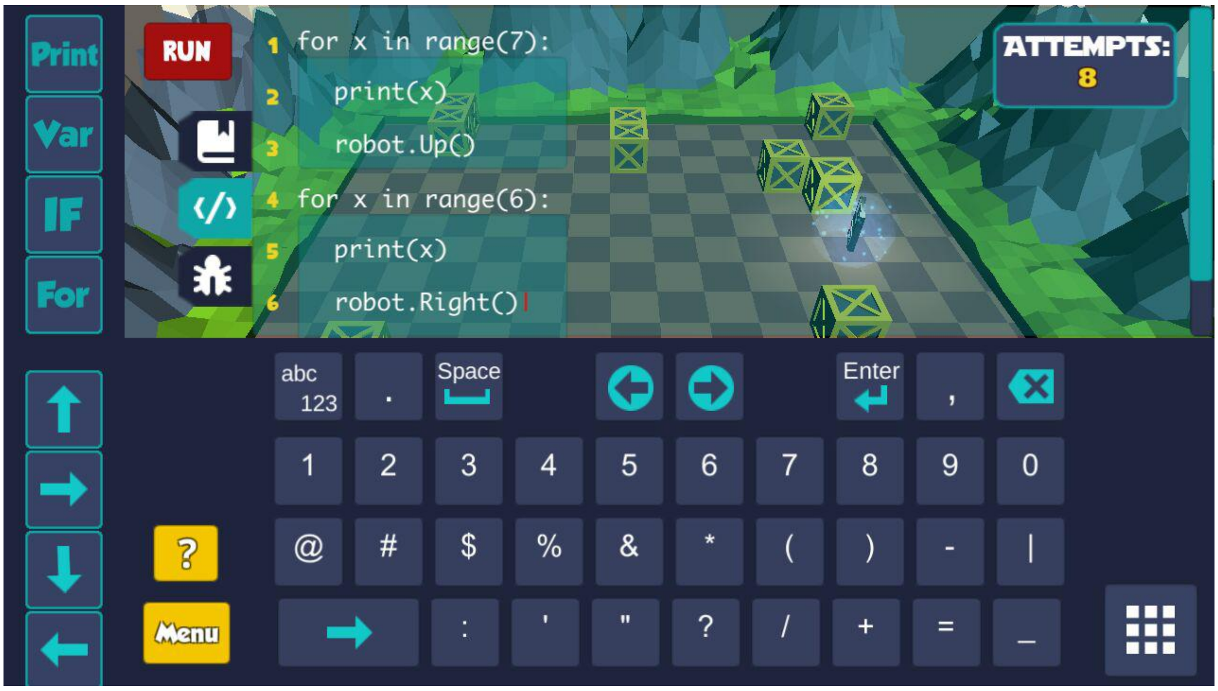

```
for x in range(7):
    print(x)
    robot.Up()
for x in range(6):
    print(x)
    robot.Right()
```

Figure 3.6.: The mobile STEM learning environment sCool.

code initiative, embedding a curriculum in their game which can be used by schools and teachers in their classes. In Cargo-Bot (Viana, 2012) the player has to program a crane to move boxes onto multiple stacks to reach a given end state (e.g. color sorted). This stack system changes the gameplay compared to the two-dimensional grid in Lightbot. The focus in Cargo-Bot lies on functional programming and recursion. Lee, Shan, Beth, and Lin (2014) evaluated the use of Cargo-Bot as a game-based approach to teach recursion in the classroom. Their findings demonstrate the effectiveness of this approach.

When these games include spatial challenges, perspective of view becomes an important factor. Coddy: World on Algorithm (Simply Projects, 2016) is another block based coding game that adds a 3D environment. In this environment the player can freely change the perspective to better understand and plan the movement actions. Similar to the other games, the program length is very limited. This way the levels are kept compact and fit the mobile use case and the learning happens in bit-sized portions.

Rose (2016) implemented two Lightbot-similar games, one with simple movement blocks and the second with Scratch-like blocks. The study examines the influence of the programming interface on the learning behavior. Rose concludes that the Scratch-like version supports learners with a bricolage approach. Advances in mobile device technology allow researchers to consider augmented reality in a mobile learning scenario. Goyal, Vijay, Monga, and Kalita (2016) created the paper-based mobile AR CT learning environment *Code Bits*. Tangible paper code blocks and the mobile AR application create an immersive mixed reality environment in which the player can try and explore. The aim is playful learning of functions, loops, and recursions.

Vahldick, Mendes, and Marcelino (2014) give a comprehensive overview of programming learning games. They observe a trend towards mobile devices and conclude, that most games are either too simple for the long term or too difficult for beginners.

3.3. Learn Through Augmented Reality

With advancing mobile technology mobile learning and augmented reality become interesting for various application fields. The following section shows different concepts with interesting approaches for this thesis.

Tomi and Rambli (2013) developed an *"interactive mobile augmented reality magical playbook"* for preschool children learning numbers. They designed a physical book and a corresponding mobile AR app. The book is designed to be readable also without the app. Markers for the app were embedded into the book, such that they are invisible to the reader. The embedded markers proved to be convenient. The special design for young children includes a tangible user interface, which means the reader can interact with the augmented 3D character. In an observational study they found that particularly the interaction offers an engaging learning experience.

DiedricAR (de Ravé, Jiménez-Hornero, Ariza-Villaverde, & Taguas-Ruiz, 2016) is a marker-based, mobile AR system for descriptive geometry learning. The markers are embedded in printed exercise workbooks. Students could use their own mobile devices to augment the content in the workbook. de Ravé et al. found that the application increases the students' positive feelings and has a positive impact on their spatial ability. They also compared the performance on different devices to point out the importance of a proper user experience. Other research also suggests that AR can support learning complex three dimensional concepts (Dünser, Walker, Horner, & Bentall, 2012; Sommerauer & Müller, 2014).

Another project that emphasizes on interaction with a tangible user interface is the AR learning environment HELIOS (Fleck, Hachet, & Bastien, 2015). This environment aims to support astronomy learning for primary school children. They designed a low-cost setup with a webcam, a laptop and printed markers. Compared to a mobile device setup this approach focuses entirely on the spatial comprehension through the tangible user interface i.e. by manipulating the printed markers. The same kind of interaction is presumably difficult while holding a mobile device in one hand. In an empirical evaluation they found that the mix of virtual and real objects and the tangible interaction support spatial abilities development and knowledge construction. These studies show that marker-based AR can be used beneficially in very different ways. Depending on the context a mobile device setting might be more flexible and intuitive for the user.

3.4. Summary

CT has become an important topic in education and there are diverse ways of transporting different CT contents. Research has shown effectiveness of tangible robot programming (Sapounidis & Demetriadis, 2013) as well as educational CT games on mobile devices (Lee et al., 2014). Overall many CT related educational applications share the concept of block-based programming (see Section 3.1). Some applications use Scratch or a similar system to encourage bricolage programming (as shown by Rose, 2016) while others use individual context related blocks. This way the barrier of understanding the syntax of a programming language is lowered. Concepts such as conditionals, functions and loops can be learned quicker and in an intuitive way. In order to achieve good qualitative results, Lye and Koh (2014) suggest a think aloud protocol and capturing the on-screen activity.

Related work shows a focus on teaching functions and loops (as shown in Table 3.1). Loops and recursions are mixed in Lightbot and Cargobot, although they still provide an understanding of recurring tasks. The goal in most games is to navigate a character through a grid by programming its movement behavior. Differently, in Cargobot a crane is programmed to sort boxes vertically on a number of stacks. The games differ mostly in the visualization of the content. While some use 2D to simplify the visual content others use 2,5D (isometric) or 3D to include spatial challenges into the game. From the mentioned games only Coddy allows a full 360 degree view on the given challenge. The complete understanding of the given environment and a clear goal are important factors to the success of the player. AR supports visual and spatial

Table 3.1.: Comparison of CT learning games

Game Title	Programming Interface		Content			Graphical Interface		
	Scratch (like)	individual Blocks	Condition	Function	Loops	2D	2,5D	3D
sCool		✓**	✓	✓	✓			✓
Dragon Architect	✓		✓	✓	✓			✓
Program your robot		✓	✓	✓	✓		✓	
Lightbot		✓		✓	✓*		✓	
Box Island	✓		✓	✓	✓			✓
Cargo Bot		✓		✓	✓*	✓		
Coddy		✓	✓	✓	✓			✓

* Recursions instead of loops
** Python code

understanding and can further offer new ways of interaction. This interaction combines the advantages of tangible interfaces and engaging augmented visualizations. Not much research has yet been conducted on CT learning in a mobile AR setting. In this thesis a prototype of a mobile AR educational game on CT is designed, implemented and evaluated. It should give insights, which parts contribute to successful CT learning.

4. Design

Literature research and related work show growing potential in teaching computational thinking skills through mobile game-based learning. The emphasis is on mobile. Mobile games can address a variety of learning preferences and can motivate different learners within a familiar environment on their own devices. AR on mobile devices offers new ways to design and create applications. Based on the conclusions from the previous chapters, the mobile AR game prototype *ARobot* is designed and implemented. This chapter describes the design process of *ARobot*. At first, the motivation and the target group are described. Then functional and non-functional requirements for the application are defined. From these requirements, a conceptual design is created as the base of the game. At the end of the chapter, design decisions are made which are important for the implementation and the final prototype.

4.1. Motivation

There are various approaches to engage young students with computational thinking. However, augmented reality on mobile devices has become applicable only recently. Hence not much research on CT education has yet been conducted with mobile AR technology. The game prototype *ARobot* shall be designed, implemented and evaluated in an exploratory study. This shall yield potential benefits of mobile AR combined with game-based learning in education within the scope of computational thinking. In the game the player shall engage with the following concepts: Analyzing of a given problem, planning a solution, sequential processing, debugging, pattern recognition, functions and loops. The programming of a robot character's movement to reach a clear goal shall increase engagement and facilitate learning. The choice of mobile devices as platform and the use of augmented reality shall increase interest and thereby motivation.

4.2. Target User Group

The primary target group for the game are young students between 10 and 14 years without much prior programming experience. The targeted generation is suggested to be very familiar with mobile technology for playing, communicating, learning or gathering information. The game should however still be interesting to play for young adults and people who already have programming experience.

4.3. Tools

First, the platform on which the game should be played needs to be considered. A desktop setting can offer resources for high-quality graphics and a high frame rate. Mobile devices, on the other hand, offer a flexible, low-cost alternative with lots of different built-in sensors and features. For the chosen target group, mobile devices are convenient because they are very familiar with it. Among mobile device operating systems, Android[1] is widespread. It is continually updated, extended with new features and open-source. Since 2018 Android versions include the augmented reality module ARCore[2]. Hence recent Android devices offer increased AR capabilities. Many different hardware producers offer mobile devices built for Android. This makes it simple to compare software applications on devices with different hardware specifications. Nevertheless, the possibility should be available to port the game on other platforms as well. A cross-platform game development framework eases the implementation according to these expectations. Available frameworks (see 2.3) offer the needed functionality to facilitate the development of a mobile AR game.

4.4. Requirements

In a first step, requirements are defined to ensure functionality and quality of the game. Functional requirements establish a usable environment while non-functional requirements assure, that specified quality measures are met. These requirements provide a base for the next design steps.

4.4.1. Functional Requirements

"A functional requirement describes an action that the product must take if it is to be useful to its operator–they arise from the work that your stakeholders need to do"(Robertson & Robertson, 2012, p. 10). In the case of a mobile learning game, the stakeholders are players. The game should provide an environment where the players can explore and thereby analyze, process and learn. The game ARobot shall be played by a single player on an Android mobile device. The tasks for the player include getting to know the user interface, playing the game and solving challenges. This leads to the following functional requirements.

Requirements for the In-Game Navigation and Customization

1. The start screen shall offer the possibility to customize the robot character
2. The start screen shall offer the possibility to enter a player nickname
3. The start screen shall lead the user to the level menu

[1]Android https://www.android.com/, accessed 08-02-2019
[2]ARCore https://developers.google.com/ar/, accessed 02-08-2019

4. The user shall be able to choose and start a level through the level menu
5. The user shall be able to change the volume at any time

Requirements for the User Interface

1. Inside each level, the user shall be able to add command blocks from the toolbox into empty slots in the program panel (by dragging and by clicking)
2. The user shall be able to delete command blocks from the program panel
3. The user shall be able to replace command blocks in the program panel
4. The user shall be able to start the execution and thereby the animation of the command block sequence
5. The user shall be able to fast forward the execution visualization
6. The user shall be able to reset the robot character to its starting position without deleting the program panel at any time
7. The user shall be able to return to the menu at any time
8. The user interface shall be on a distinct layer from the level contents

Requirements for Augmented Reality Interaction

1. An image with many features shall be preprocessed to be used as a well detectable marker
2. The system shall be able to detect the specified marker
3. The system shall place level contents based on the location of the marker.
4. The system shall scale the displayed level contents depending on the distance to the marker
5. The user shall be able to freely choose the game perspective by moving the camera around the marker.

4.4.2. Non Functional Requirements

The quality of a game is defined by how the players receive it. Hence providing a good experience is prioritized when designing a game. *"Non-functional Requirements are properties, or qualities, that the product must have if it is to be acceptable to its owner and operator"*(Robertson & Robertson, 2012, p. 10). The following requirements define how the game should work and set quality guidelines for the development.

Usability & Interface To address the mobile use case, the user interface shall be simplistic, well-arranged, with commonly used symbols and the expected feedback.

Reliability The game shall not crash or perform unexpected behaviour upon user input in 90% of the cases.

Performance The game shall start in less than 10 seconds. During the gameplay a minimum of 30 frames per seconds (FPS) shall ensure a proper game experience.

Coding Standard The project source code shall follow the *.NET Foundation Coding Guidelines* for consistent readability.

Platform The game shall run on any mobile device running Android 7 (or newer) with a backside camera.

Localization The game shall be entirely in German, but shall include the possibility to add other languages.

4.5. Conceptual Design

Based on the requirements, an initial design is created (see Figure 4.1). This section discusses the considerations that influenced the outcome. In the game the player programs a movement sequence of a character to reach goals in short levels with a clear goal. A sequence of simplified movement blocks shall help the player to comprehend the sequential execution of code. By planing and creating the movement block sequence, the player analyzes the problem scenario. By executing this sequence and observing the movement of the character, the player learns if the plan was successful or not. In the latter case, the player can delete and change the program and re-execute. Related work shows that a robot-like character engages young players and creates a logical connection to Computer Science. In addition, customization (e.g. choice between male and female robot) increases the immersion (Teng, 2010).

4.5.1. Game Story

A story contributes to the player's immersion in the game (Ryan, 2009). The story shall introduce an environment in which the game takes place. Implicitly the story explains the goal of the game.

> A robot is sent on a mission to another galaxy. The spaceship crashes on Mars. The robot needs to collect lost parts of his rocket to be able to continue his mission. You (the player) are observing the robot through a satellite drone (your smartphone). Help the robot by sending remote movement codes, which the robot can execute. After completing all levels, the robot reaches his spaceship and can continue his mission.

4.5.2. User Interface (UI)

Mobile devices require a particular UI design. Usually, there are few hardware buttons and the touchscreen is the most used input. Current devices allow multi-touch and gestures such as dragging and pinching (zoom in and out with two fingers). At the same time, the touchscreen is usually the only display. Hence a well-fitting UI design is crucial, to not interfere with the display capabilities and at the same time place important UI elements such that they are comfortably reachable. Besides, the screen must be able to scale to different screen sizes. On the other hand, it is important to consider, how the player will interact with the UI (see Figure 4.2).

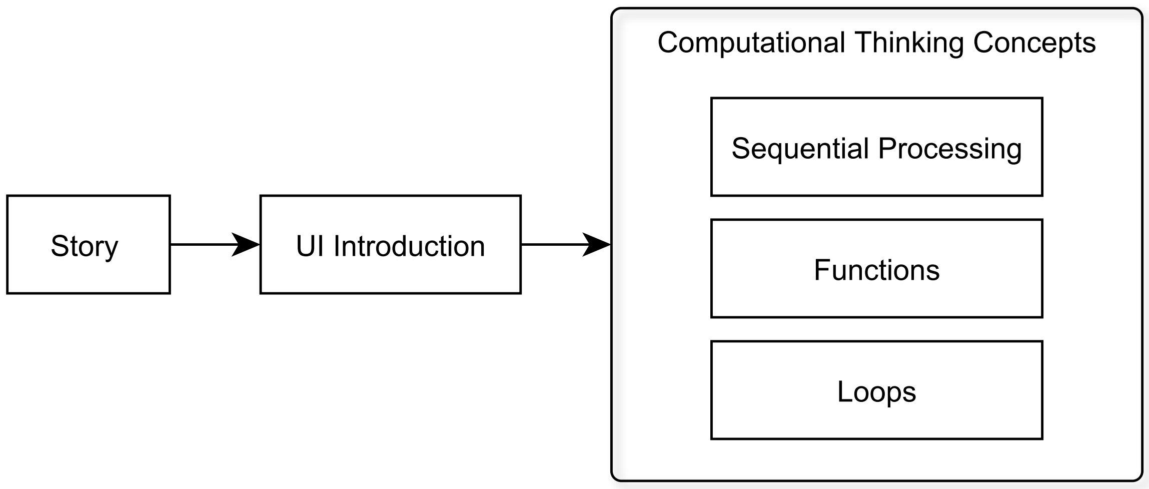


Figure 4.1.: Conceptual play flow in the game.

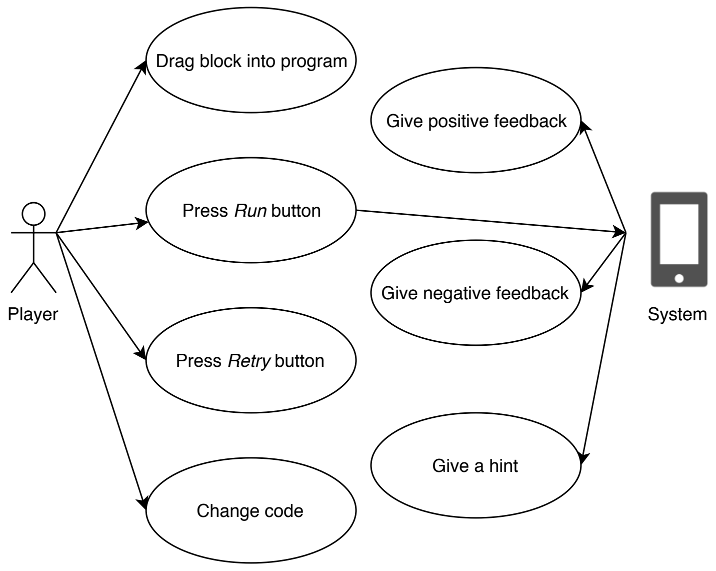


Figure 4.2.: User interaction use cases in ARobot.

4.5.3. CT Concepts

The player is confronted with a challenge. To overcome this challenge, the player needs to analyze the situation. With a limited set of tools, the player can plan a sequence of steps to reach the goal. On execution, through the character animation, the player automatically reviews his solution and is able to debug if necessary. When these concepts are understood well enough to solve the first challenges, the function tool is introduced. With this tool, movement patterns can be reused. This shall assist the player to recognize patterns and find efficient solutions. Building on the experience with function calls, the loop tool is introduced. In a similar way as functions, the loop tool shall strengthen the idea of recurring patterns. Both tools are used in combination with the sequential "main code". This approach inspires the decomposition of a bigger problem into smaller ones (repeated or not). Finally, the number of program slots is limited in such a way that the player has to plan efficiently in order to reach the goal.

4.5.4. Level Design

Each level consists of a scene with the robot character and a clear goal. Obstacles create spatial challenges, which define the difficulty and a possible solution set of the level. Well-thought level design is very important for good comprehension. The learning curve should neither be too steep nor too shallow. The beginning levels should be introductory and help to understand the gameplay and the task. Then levels have to become more challenging in order to keep the player engaged. This could then lead to larger or even open-end levels. However, literature research and related work showed that larger levels and longer code do not necessarily lead to better comprehension. On the contrary, if a level is too large and complex, that could be frustrating and decrease motivation or lead to drop-outs. To keep higher levels challenging and interesting, new game mechanics may be introduced. This can increase the complexity without increasing the necessary number of commands. Additionally, short levels support familiar handling as in other mobile games.

4.6. Design Decisions

The considerations and ideas in the conceptual design lead to several open questions. The answers to these questions provide a clear idea, how the implementation should behave and look like.

4.6.1. User Interface

The screen is divided into two layers (see Figure 4.3). The first layer is placed on the tracked AR marker and is rendered dynamically when the marker or the camera is moved. It displays the world in which the robot moves around. On this layer, there is no direct interaction or input from the player. The second layer is on top of the first one

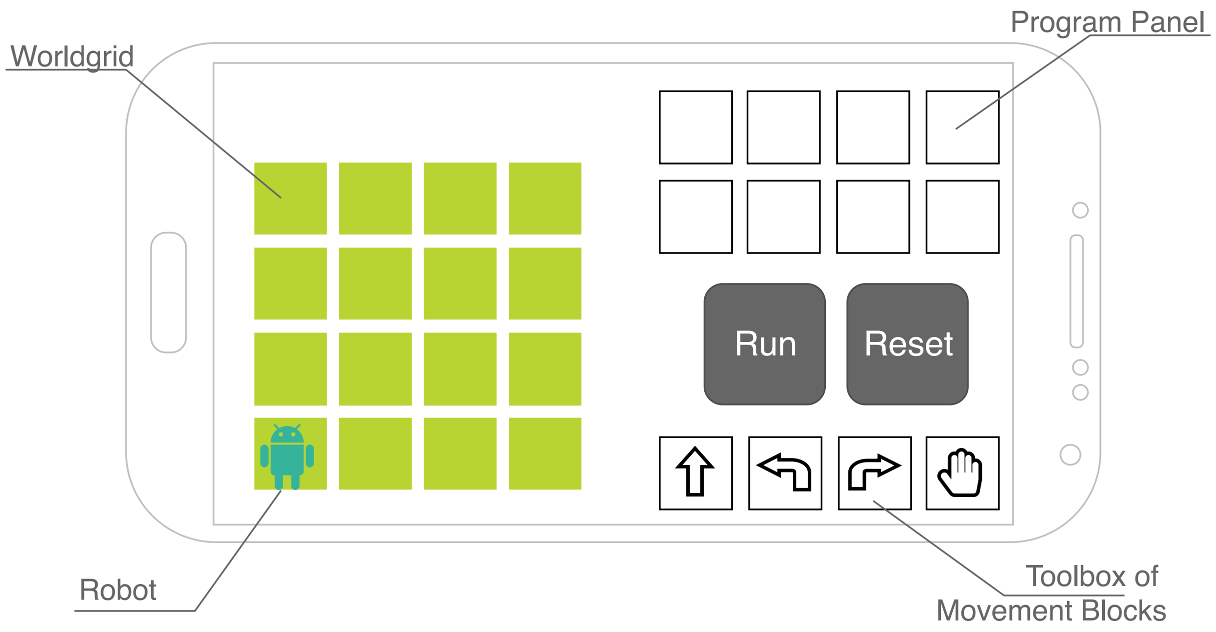


Figure 4.3.: A first sketch of the game's user interface.

and static. This layer is the user interface. It receives the user input and manipulates robot movements in the first layer. This indirect way of controlling the robot engages the player to reflect on the given problem scenario and plan a path before execution. The use of tangible elements (except the marker) is neglected because the interaction appears to be difficult while holding the mobile device in one hand.

Related work shows that block-based programming provides a user-friendly learning environment without much syntax overhead. In ARobot, a limited set of functionality blocks assists faster learning of mechanics and thereby supports understanding overall. Similar to an inventory in various role-playing games, the program panel consists of slots that can be filled. A *Toolbox* panel contains the movement command blocks, which can be added to the *Program* panel. In the beginning, there are four blocks: Go forward, turn left, turn right and pick up. In later levels, the toolbox is extended and an additional panel is added for function calls and for invoking a loop. To simplify the understanding of loops, the number of iterations can be set directly by the player. This enables the player to intuitively engage with the concept of loops without explicitly learning variables or conditions beforehand. The *Run* button executes the movement commands in the program panel (if possible) in the sequence order. The *Retry* button restarts the level (according to the functional requirements). This option is essential for trial and error during the gameplay.

Marker-based AR is chosen for free perspective inside levels. Compared to markerless AR, markers facilitate tracking and locating in a real environment. Thus increasing stability and using fewer resources. This improves usability, especially on older or low-cost devices with little resources. In addition, markers can be embedded in educational print materials such as books or exercise sheets. Therefore a QR code was preprocessed as a starting point for the instantiation of the game elements. QR codes have many features and thereby strengthens the stability of the tracking.

4.6.2. Levels

The first question that arises for designing levels is if they should be predefined or dynamically created. The levels shall not be too easy nor too difficult, especially not in the beginning. To ensure a concrete comparability and the wanted rise in difficulty, nine predefined levels are designed. However, dynamically generated levels could be very interesting, as long as they are solvable and adapting to the player skill. The size of the grid on which the character can move can also be variable. Related work proposes that the size should be relative to the size of the program. Especially in the mobile case both are quite limited through screen size and practical usability of the UI. For the predefined levels, a grid size of 4x4 is chosen. This size keeps the levels compact and still provides enough place for increasingly difficult challenges. Related work partially includes three-dimensional spatial challenges, which increase the complexity and difficulty of the game. This might be too difficult for beginners and interfere with the introduced concepts. Which is why the introductory challenges are designed on the plane only. In further levels, higher complexity would be welcome to keep the players engaged.

4.7. Summary

In this chapter the design process of the mobile augmented reality game ARobot is discussed. The purpose of embedding computational thinking concepts in the game involves several challenges. After specifying the core idea and the target group, requirements are defined. These requirements lead to a conceptual design, that is refined through further design decisions. The game will be developed with a game development framework to facilitate the development process. The user interface will be intuitive through simple buttons and game specific blocks. Individual movement command blocks facilitate the planning and comprehension during the game. Levels will consist of 4x4 shaped grids, on which the Robot has to be navigated to a specific goal. The outcome of the design phase builds the foundation for implementing the game prototype.

5. Development

The previous chapter gives a clear idea of how the result should look like. The desired outcome is a user-friendly mobile augmented reality game that engages the player with computational thinking concepts. This chapter describes the development environment and the implementation details.

5.1. Development Environment

Game development with all its components (physics, visualization, animation etc.) comes with a certain complexity. This complexity contains recurring patterns and similarities that are used by many game developers. Game development frameworks and engines help to overcome this complexity and facilitate the development process. For the implementation of the prototype alongside this thesis, the reviewed frameworks (see Section 2.3) and the specified requirements were compared. After this comparison, the two frameworks with the most fulfilled requirements were considered: Unity and Unreal. The Unreal engine offers quick and simple creation of stunning visuals. However, high-end graphics weren't in the focus of the prototype. Augmented reality on an average mobile device requires to keep the graphic rendering overhead low. Unreal's visual scripting system Blueprints[1] maps complex programming concepts in a very intuitive way. Again this feature is not specifically beneficial for the development of our prototype. We decided that Unity is the best suitable framework for the prototype development. Unity's prefab system is very efficient for the reuse and modification of recurring game objects. In the prototype, the level environments shall fit to the game story the levels are designed to be very similar. Therefore the simple reuse of game objects was an important factor. Further the integration of the AR framework Vuforia[2] in Unity supported this decision. It's worth mentioning that with every new version, new features are included in the frameworks, thus influencing the choice in the future.

5.1.1. Unity

Unity is currently one of the most used cross-platform game development frameworks. It is developed and maintained by Unity Technologies[3]. A large user community

[1]Blueprints https://docs.unrealengine.com/en-US/Engine/Blueprints/index.html, accessed 03-08-2019
[2]Vuforia http://vuforia.com/, accessed 02-08-2019
[3]Unity Technologies https://unity.com, accessed 08-07-2019

contributes through feature requests, bug reports and most importantly through the Unity asset store. The asset store allows developers and designers to share or sell their creations, so that others can reuse them. Unity offers an extensive and well-maintained documentation as well as many tutorials to support developers. Unity is used for the development of the game prototype in this thesis. The decision to use Unity is based on a review of currently available and eligible game development frameworks (see 2.3). Following some of the key features are highlighted.

- Unity's cross-platform approach allows creating builds for most current systems. In particular, Unity supports builds for the most common mobile operating systems Android and iOS.
- The unity asset store is accessible through the web or directly in the editor. It offers a large amount of free and premium content, from simple 3D models to complete functional showcase worlds.
- The unity editor provides predefined 2D and 3D settings to ease the work with respective content (e.g. optimize the import and use of models). Unity offers real-time 3D rendering and different rendering pipelines to optimize graphics for different hardware.
- Since version 2017.2 the Unity editor includes the augmented reality engine Vuforia.
- Unity offers a toolkit to simplify the creation of user interface elements.
- Unity uses so-called prefabs for the reuse of game objects. Further the prefab system allows dynamically triggered initialization of predefined game objects.
- In Unity's terminology a game is structured into scenes. This system allows a simple organization of levels and menus.
- Unity provides an entirely built-in physics engine. The components facilitate processes such as collision detection, applying force to an object and gravitation effects.

5.1.2. Vuforia

Vuforia is an engine for creating augmented reality applications. The integration in the Unity editor (since version 2017.2) supports a seamless use of AR components in Unity projects. Vuforia offers state of the art mobile augmented reality based on different tracking techniques. For the prototype development alongside this thesis, Vuforias marker-based tracking feature was chosen. This feature allows the detection and tracking of pre-processed pictures. In the pre-processing step features are extracted from a picture and converted into an image target. The stability of the tracking depends on the number of features. Therefore a QR code with many features was used for the prototype implementation. In Unity a camera object with a Vuforia Behaviour script component (further called Vuforia camera) changes the scene background to the camera view. When the corresponding marker to an image target is detected, the objects connected to the image target (in Unity) are displayed on the marker in the real world.

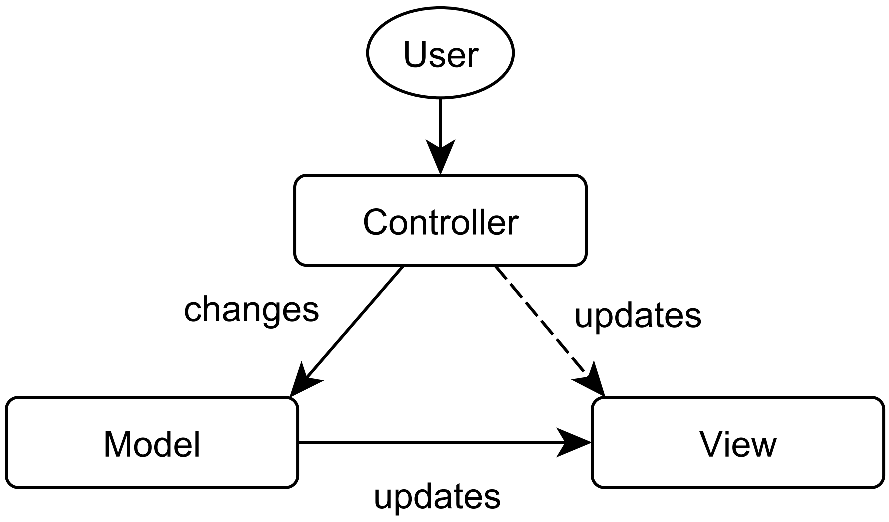


Figure 5.1.: The Model-View-Controller design pattern.

5.2. Implementation Details

The outcome of the design phase shows a clear distinction between the user interface and the actual game content. For this reason, the model view controller (MVC) software design pattern was chosen. MVC separates the user input (controller), the data logic (model) and the data representation (view). MVC offers interfaces between its components, such that the controller changes the model and the model updates the view (see Figure 5.1). The user sees the update and influences the model through the controller.

In the context of design for mobile devices, MVC has to be adapted because the screen acts as controller and view at the same time. In Unity, it is a non-trivial approach to apply the MVC pattern, because most dynamic game objects use Unity's base script MonoBehaviour. This script takes care of initialization, physics, events, logic, rendering and more. However, for this prototype model, view and controller functionality are separated as much as possible. The model plays the central logic role and triggers logic components in other objects, This approach provides a proper distinction so that the player can focus on analyzing, planning and debugging.

5.2.1. Model

The game manager (GM) is the representation of the model in the game. It connects the user interface elements with the game logic. Upon user input (i.e. push button) the GM updates its background data and (if necessary) the view. For instance, on execution the GM retrieves the sequence command blocks from the program panel and triggers the movement of the robot character (see Listing 5.1). The GM is also responsible for triggers, such as finishing a level and sound effects. The GM is implemented as singleton, so it exists in every level exactly once.

Listing 5.1: The code execution function in the game manager triggers the robot movement.

```
public void RunSequence()
{
 List<string> programSequence = RetrieveSequence();
 if (programSequence != null)
 {
  SetRunButton(false);
  IsInExecutionMode = true;
  _robotMovement.StartMoveSequence(programSequence);
 }
}
```

Persistent Data

For simple data (i.e. int, float, string) Unity offers so-called PlayerPrefs (see Listing 5.2). as a way of persistence between scenes. In the developed prototype PlayerPrefs are used to store the player name, active scene index and volume settings.

Listing 5.2: Usage Example of PlayerPrefs

```
PlayerPrefs.SetString(key, value);
PlayerPrefs.GetString(key);
```

To implement the MVC pattern the objects representing the model need to be available scene-independent. A game object can be made persistent by adding a call to the method DontDestroyOnLoad to the initialization method (see Listing 5.3).

Listing 5.3: Persistent Game Object

```
void Start(){DontDestroyOnLoad(gameObject);}
```

When a scene is closed, all game objects in that scene are cleared. The method *DontDestroyOnLoad* prevents that and the respective game object continues to exist in the next scene. This functionality is used by two persistent model objects. The level manager acts as a communication bridge between the current game state, the scene manager and the PlayerPrefs. It supports switching from the current level to the next level as well as any level through the level menu. Further, it can be used to restrict access to levels conditionally. The sound manager stores and changes the sound settings in the PlayerPrefs. Additionally, it can randomize a range of sound effects as well as the pitch of the sound effects for a more realistic sound environment.

5.2.2. View

The components in the view object are responsible for visualizing the level contents and animations. The view includes a 3D and an AR mode which can be switched

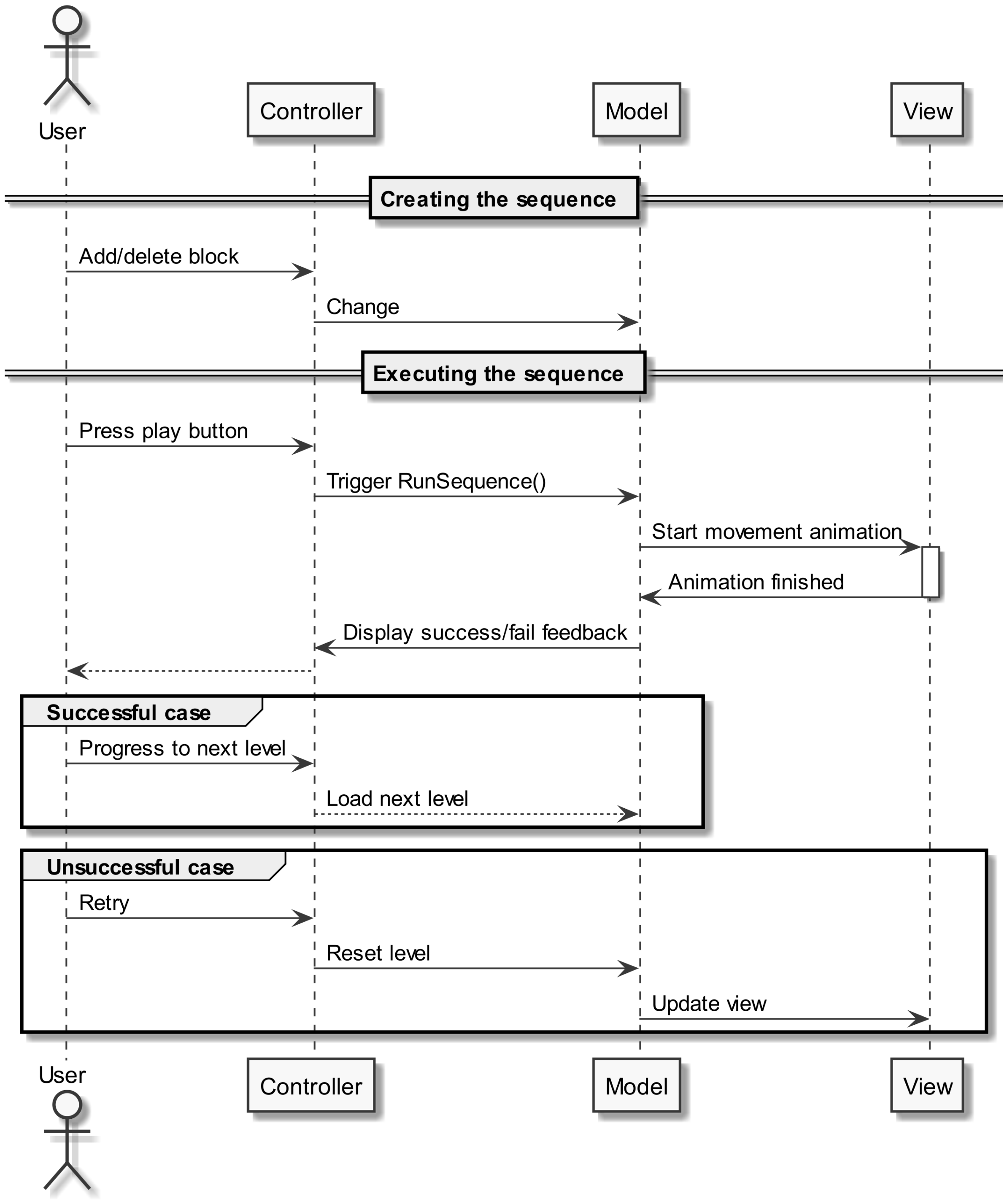


Figure 5.2.: An example interaction sequence between the user and the MVC components.

easily. The AR view consists of the scene lighting, the Vuforia AR camera and the image target. The image target contains the boardmanager, which is loaded when the device camera detects the marker for the image target. The boardmanager instantiates a specified level prefab and handles the positioning of level objects. The level prefabs are defined for each level scene separately.

5.2.3. Controller

The user input in the developed prototype is designed for block-based programming. This approach requires a set of available program blocks, a program space, a way to execute the code and feedback upon execution.

The movement command blocks are objects with the following tags and corresponding symbols: Forward, Left, Right and PickUp. A panel on the bottom right of the screen contains these blocks (see Figure 5.3). From here, they can be added to the program by dragging or clicking. The program space is implemented as a panel, which contains empty slots. A slot detects if a dragged command block is dropped on it. If the slot already contains a command block, the dragged block replaces the old one. If the slot is empty, the block is added to the first free slot in the program. Single command blocks can be removed from the program by clicking or by dragging them onto the trash can next to the program panel. In later levels, an additional program panel is added for the introduction of functions and loops. A button with a play symbol triggers the execution of the command blocks in the program panel (see Figure 5.2). On execution, this button is replaced by a retry button. Depending on the success of the execution, the user interface displays a retry or a level finished overlay. The execution also triggers the animation of the character movement, which illustrates the visible feedback of the programmed movement.

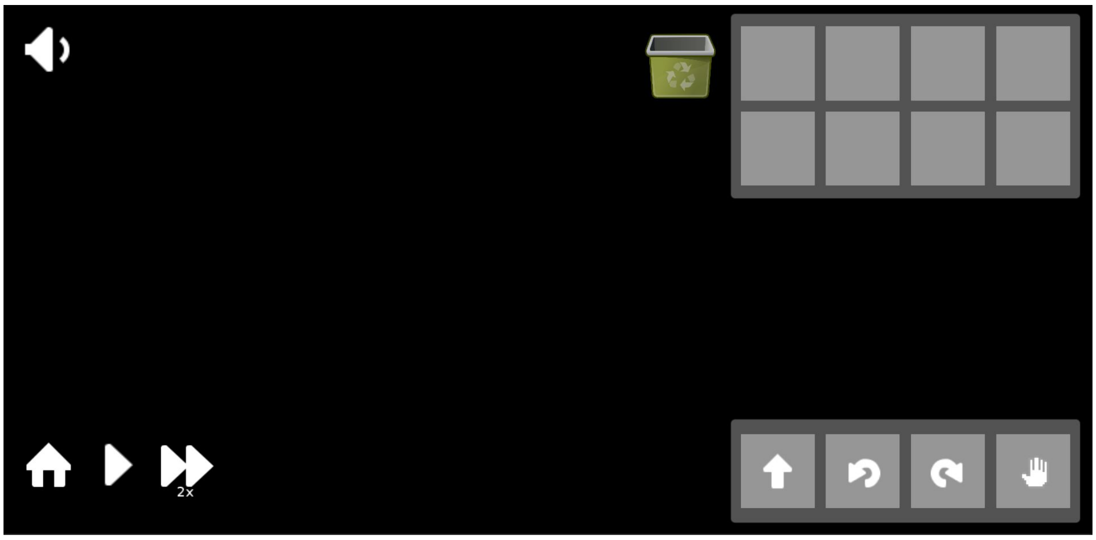


Figure 5.3.: The user interface consisting of the program panel, movement block toolbox and control buttons.

5.3. Scenes

The game is structured in a typical way for mobile games to increase usability. The start screen quickly introduces the player to the game environment. A martian universe background sets the relation to the game story. The robot character floats in the front and welcomes the player. At the start screen, the player can choose a color and a name for the robot. The start screen leads to the level menu. The level menu functions as an overview and entry point for all levels. The levels are structured into basics, functions and loops corresponding to the included challenges. Completing the final loop level leads to the final screen.

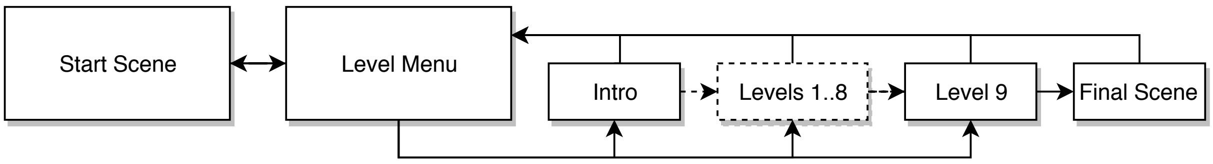


Figure 5.4.: A schematic overview of scenes and the transitions in between.

5.3.1. Level Elements and Animations

The used level elements shall represent the story to increase the player's immersion. Therefore all models and textures were chosen accordingly to the science fiction story taking place on planet Mars. Reddish stony textures are chosen to create a martian environment. Big red-brownish stones function as obstacles inside the level scenes.

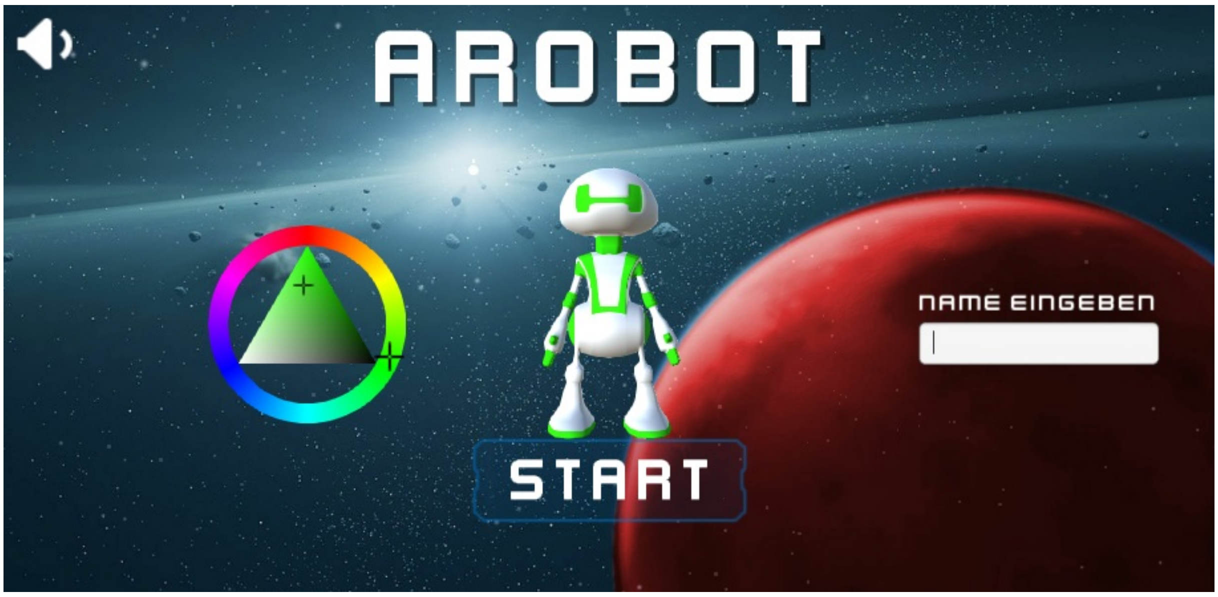


Figure 5.5.: The ARobot start screen including customization.

Robot

The robot is the main character of the game. The character will move on the level grid and pick up the target items. Hence two animations are needed; walk and pickup. The walk animation is triggered upon execution (when the player presses the run button). The animation is only visible if the robot is able to move as programmed. Similarly, the pickup animation is triggered only if the robot has already moved onto the target items position. Colliders facilitate the detection of this condition. Eventually, a wave animation is used to welcome the player on the start screen. More animations can be added to the animator controller (see Figure 5.7).

Alien

The alien character (see Figure 5.6) functions as an alternative to the static stone obstacles. It is animated and can move along a given path. The function as a moving obstacle was neglected after the first tests because it was not necessary for the level concepts. However, the idle Alien obstacle fits well into the Martian setting.

5.3.2. Levels

When the Vuforia camera in a scene detects the specified marker, the view object loads the level prefab defined in the boardmanager. Therefore the designed levels are implemented as prefabs with obstacles, positioning (and orientation) information of the robot and the target item.

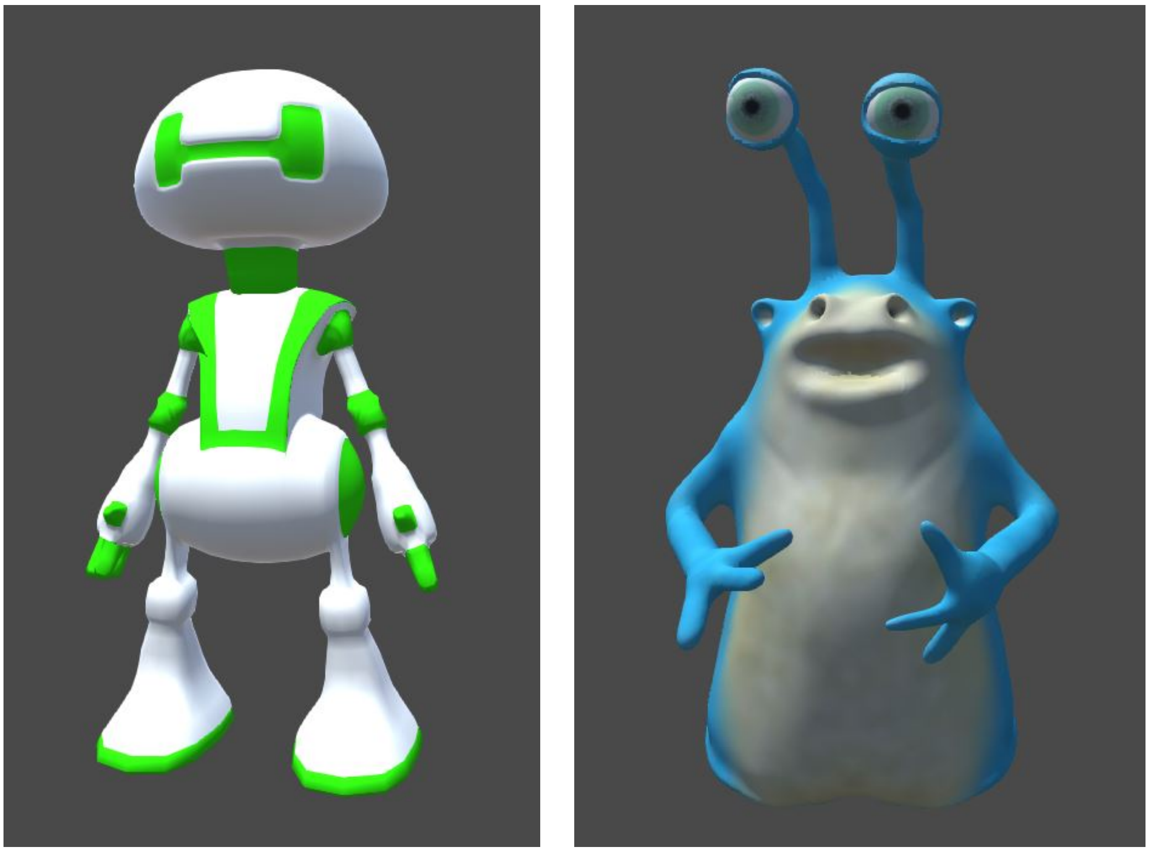


Figure 5.6.: The robot and and the alien character.

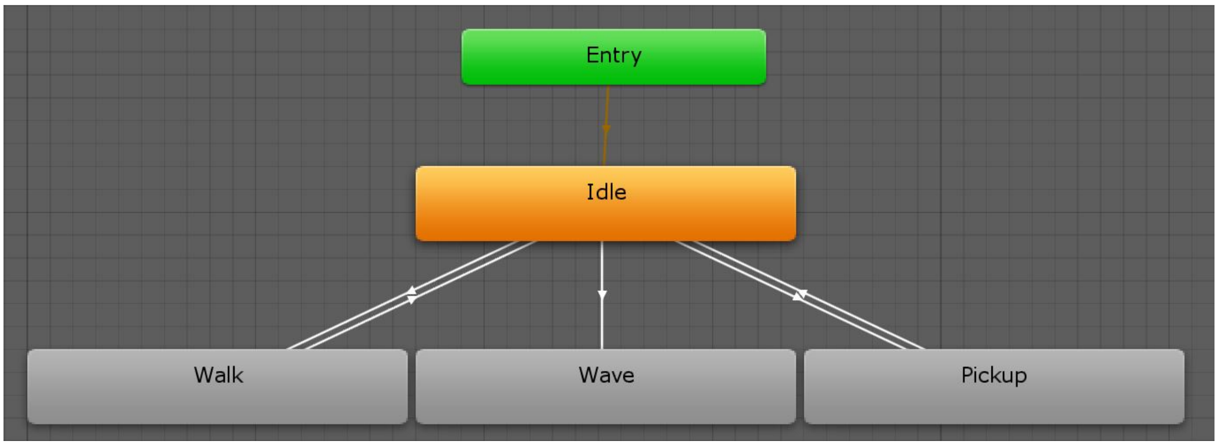


Figure 5.7.: The robot animation controller, which is responsible for triggering the animation transitions between the states.

Basic Levels

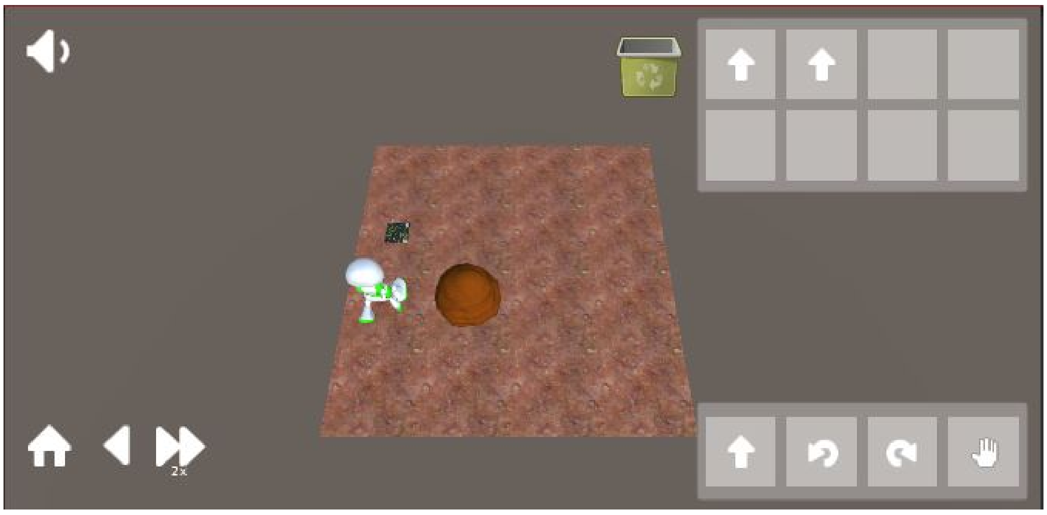
The introduction level consists only of a transparent QR marker, which shows to robot character when tracking the printed marker. This way, the player comprehends quickly how the marker-based tracking works. The following basic levels introduce the player to the user interface and game mechanics. Only a 3x3 fraction of the 4x4 grid is used to reduce the complexity. Here the player begins to engage with analyzing the given level scenarios, planning a solution, sequential processing and debugging.

Function Levels

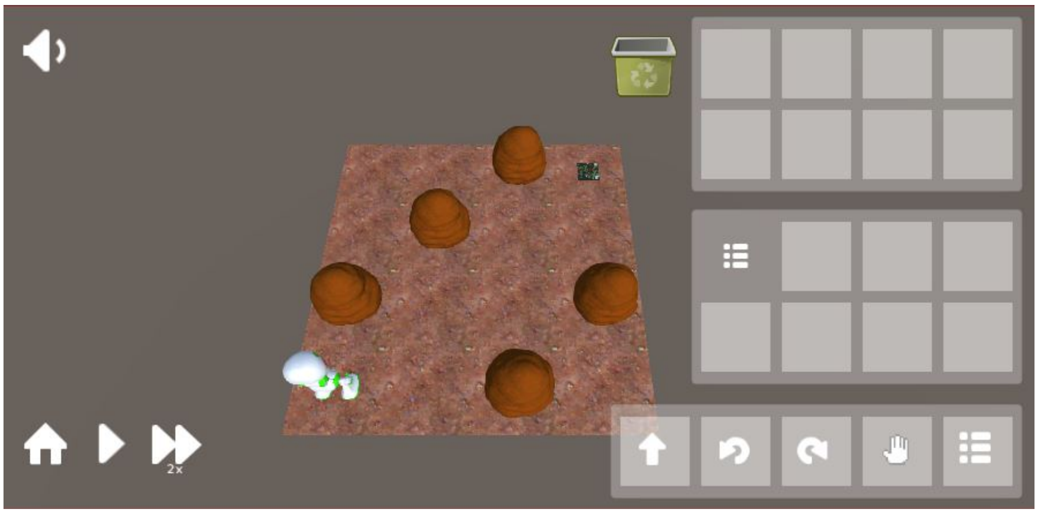
After the basic levels, the function concept is introduced. A new function block is added to the toolbox together with a second program panel. The function block can be dragged from the toolbox into the program panel. On execution, the code blocks from the distinct function panel (see Figure 5.8b) are executed at the location of the function block in the program panel. In the function levels (see Figure 5.9b), the full 4x4 grid is used. Thus the paths between robot and goal are longer. The increased complexity demands the player to plan and efficiently use the limited slots. All function levels require the function panel to be used. Optimal solutions for these levels use recurring patterns through the function tool.

Loop Levels

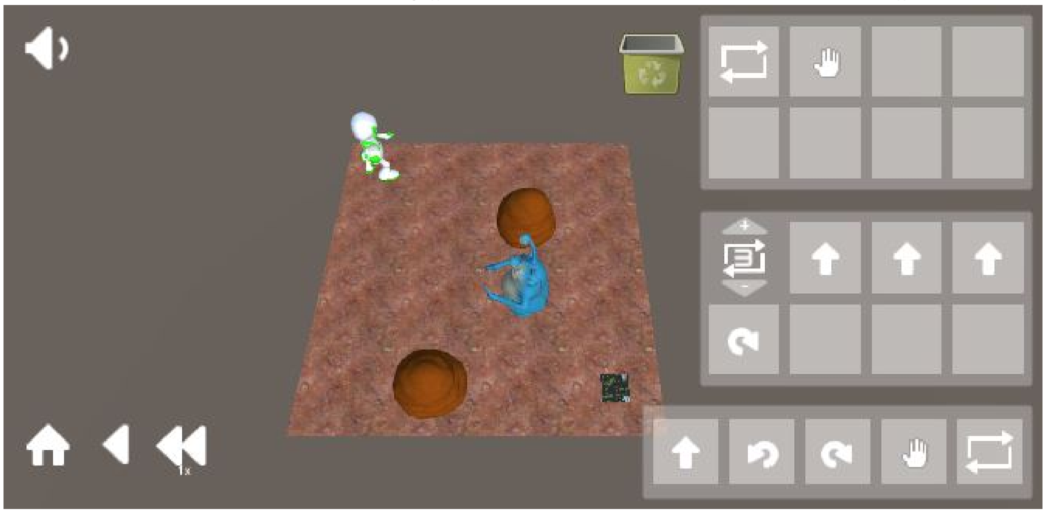
When the three function levels are completed, the loop tool is introduced. The loop tool is based on a simple number of iterations to increase intuitive understanding. The player can increase or decrease the number of iterations by pressing the + or − button (see Figure 5.8c). In the first loop level, the player can experiment and get familiar with the new tool. The following level contains a simple recurring movement pattern for the solution path. In the third and final level, the challenge is the same as in the third function level. This shall strengthen the comprehension of recurring patterns. At the same time, the player has already seen this challenge before. Supposedly this helps to solve the level while learning the concept of loops.



(a) The first basic level introduces the game environment.

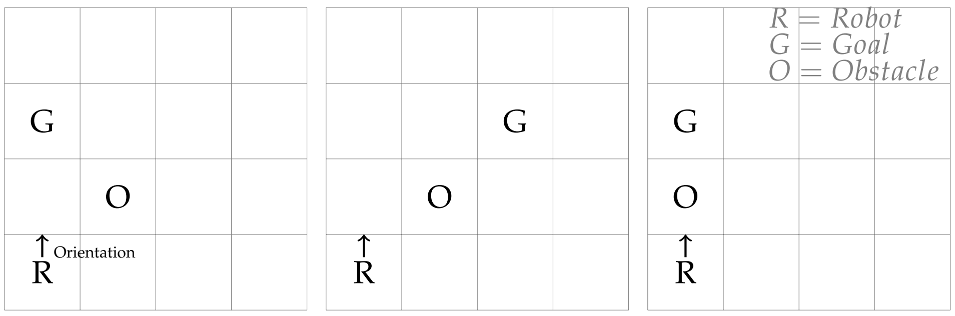


(b) A level with a recurring pattern and the additional function panel.

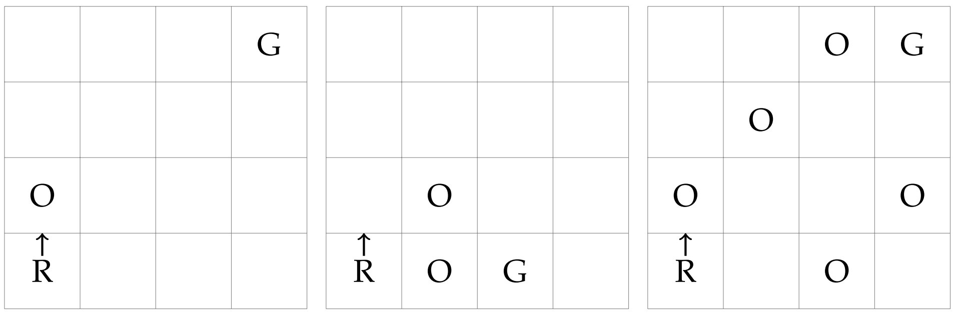


(c) The loop functionality allows the player to reuse movement sequences a specified number of times.

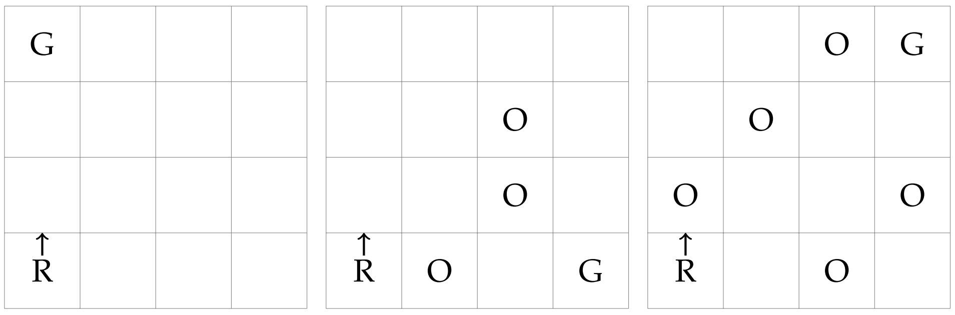
Figure 5.8.: User's perspective of three levels from different stages.



(a) Basic levels



(b) Function levels



(c) Loop levels

Figure 5.9.: Level schemes of the nine predefined levels.

5.4. Summary

The outcome of the implementation alongside this thesis is the marker-based augmented reality game prototype ARobot. The implementation is based on related work analysis and the preceding design. First, a suitable development environment was established by using the game development framework Unity and the augmented reality engine Vuforia. Then the core structure, mechanics and further the individual levels were implemented.

The game contains customization options, a tutorial and nine predefined levels. These levels lead the player through the following concepts: Analysis of a given problem, planning a solution, sequential processing, debugging, pattern recognition, functions and loops. With the function tool and the loop tool building recursions is possible. The player may use recursions to finish the levels. However, recursions were not particularly of interest for the concept of this prototype. Hence the number of recursive calls was limited to avoid inconsistencies. The evaluation in the next chapter provides deeper insights into how the prototype is perceived by test users.

6. Evaluation

The development of the game prototype alongside this thesis was motivated through the increasing importance of computational thinking in education. Therefore we wanted to evaluate how well the prototype is accepted by the target audience in an educational setting. To ensure a quality baseline, the prototype was tested preliminarily with a small group. The feedback was implemented into the next prototype version, which was then tested by a group of young students in a school project. A comparison group of university students should give insights on the possibly different perception of the game. This chapter describes the process of the evaluation and discusses the results.

6.1. Expert Evaluation

In the first iteration of the evaluation, a small group of users from the target group examined the stability, usability, designed use case scenarios and defined new use cases of the prototype. The results of this preliminary evaluation should be used to improve and provide a stable prototype with excellent usability. Additionally, the procedure should confirm what is useful and give a time estimate for the final evaluation. During the whole test, questions would be answered to ensure a smooth test flow (see Figure 6.1). The questionnaires and the interviews were conducted in German to lower the communication barrier for the young test users. The feedback from this evaluation shall improve the procedure of the final evaluation.

6.1.1. Procedure

The expert evaluation is structured in a pre-test questionnaire, the play-test, a post-test questionnaire and a concluding interview. The test is conducted with each test user individually. This way, it is possible to focus on the individual's experience and needs. First, the test user answers a questionnaire related to her affinity for games, smartphones, augmented reality and computer science. Then the project and the prototype are introduced and the test device is handed out. The test user is then instructed to start a screen recording app and afterwards the prototype. Then the game story, the goal and the user interface are explained. During the play-test, the test user is observed and arising questions are answered. The post-test questionnaire consists of nine open questions about the experience with the prototype and the game engagement questionnaire (Brockmyer et al., 2009). After the questionnaire, an interview with the test user shall provide further insights on the users perspective and experience.

Figure 6.1.: Expert evaluation of the ARobot prototype.

6.1.2. Test Users

Two female students (12 and 13 years old) from local schools volunteered to take part in the prototype evaluation. Through the pre-test questionnaire, the test users provide a self-evaluation on their previous experience. According to the answers, the test users have little experience with augmented reality. This might be explainable through the absence of the term augmented reality in popular applications, which use AR. One of the test users has actually tried Google Glass and Microsoft Hololense before. However, both test users have a lot of experience with video games. They describe their experience with computer science and programming as above average.

6.1.3. Test Device

Current mobile devices provide a wide range of diverse hardware, screen sizes, camera resolutions etc. In order to test AR applications on a mobile device, it is recommended to use newer devices with improved hardware and the latest operating systems. Recent versions of Android and iOS offer their own AR modules (ARCore and ARKit) within the operating system. The device used for the expert evaluation is a Google Pixel Smartphone running Android 7.1 Nougat. Overheating is a known issue for AR applications on smartphones. The hardware structure in smartphones is made as compact as possible. This causes heat when parts are used extensively. The camera tracking, as well as the 3D rendering on the GPU, use many resources. To avoid a failure of the test device and to improve performance, the number of trackers, models and polygons have to be kept low.

6.1.4. Preliminary Results

The procedure per test user took approximately 45 minutes, which was longer than expected. The observation combined with the questionnaires, the screen capturing and the interview provided valuable feedback and information, which is implemented for the final evaluation.

Questionnaires

The pre-test questionnaire provided only little meaningful information and can be reduced to a minimum. Through the post-test questionnaire, the test users reflected upon the play-test. The answers provided feedback on what was well accepted or understood and what could be improved. The game engagement questionnaire however did not yield much useful information. To avoid overhead and comply with time limitations, the game engagement questionnaire is omitted in the final evaluation.

Interview

Single interviews after the test procedure provided a better understanding of the participant's perspective and expectations. This feedback is used in the next iteration of the prototype development and should lead to improved usability. The test users were very interested in the game and thought it was engaging and motivating. In the beginning, they had difficulties with the user interface. Some button functionality was not clear, because the idea of the game itself was not clear yet. The support during the play-test proved to be very helpful in these situations.

Translated answers to the question "What could be improved?":

1. Some user interface elements could be more intuitive.
   For example, there could be a paper bin to delete blocks.
2. A simple sound on/off button would be nice.
3. A wider variety of level elements would make the game more interesting.
4. After finishing a level, one should be rewarded with some visual effects and learning content.
5. It would be helpful if the retry button did not delete the program.

Screen Capture Analysis

The analysis of the screen recordings revealed some minor bugs, which were fixed in the next iteration. During the play-test, one of the test users accidentally discovered the possibility of creating recursions with the function or loop functionality. Recursions were not designed as learning content in the prototype. However, instead of blocking recursions, a specified depth of recursions was supported in the next version of the prototype to avoid infinite recursions. In some situations, the test users were stuck or took longer to plan. It would be interesting to know what the user thinks in these situations.

6.2. Final Evaluation

Building on the outcome of the expert evaluation, we designed and conducted an evaluation of the prototype with two different groups. The gathered feedback from the expert evaluation was implemented in the prototype to improve its usability. The procedure was optimized to fit the limitations and keep the participants engaged. Through the results of this evaluation, we want to get a deeper understanding of the following questions:

Q1 Is our mobile augmented reality game suitable to engage students in classroom with computational thinking concepts?
Q2 Does marker-based AR enhance the mobile learning experience?
Q3 How do lessons learned in-game transfer to an on-paper task?

6.2.1. Participants

In chapter 4 we declared young students between 10 and 14 years age to be the main target group. The majority has already experience with smartphones and mobile games. In addition, they are curious to try out new technologies like augmented reality. We tested the app with volunteers from the designed target group (group A) and with university students (group B). We assume that participants from both groups will have a similar experience regarding the usability of the prototype. However, we assume that participants with programming experience will finish the levels faster and with fewer attempts. Each participant is assigned to a pseudonym, which is used to connect results, feedback and analysis. Before the test, all participants are asked about previous experience with games and with programming. Additionally, they have to fill a letter of agreement for the research use of anonymized data.

Group A - Young Students

Eleven students, six girls and five boys from a local school volunteered to take part in the evaluation. The average age in the group was 14,55 with a standard deviation of 0,782. For the evaluation, male and female students separately tested the prototype. Firstly the evaluation was set up for a maximum group size of six. Secondly, the participants were allowed to interact and help each other. Their teacher suggested that a separation would contribute to a better learning environment.

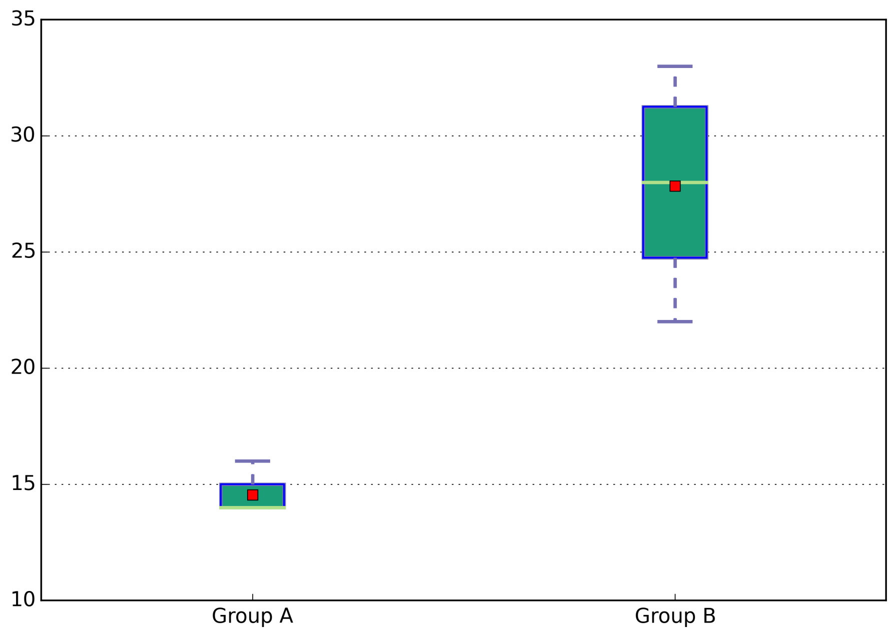


Figure 6.2.: Age distribution of group A and group B.

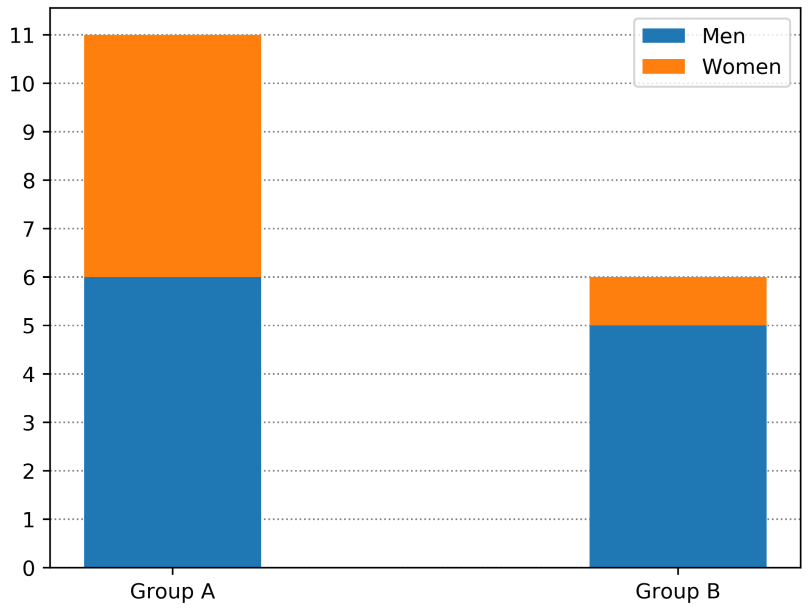


Figure 6.3.: Gender distribution of group A and group B (in years).

Group B - Graduate and Undergraduate Students

One female and five male students volunteered to take part in the evaluation. All participants in this group are students at Graz University of Technology. The average age in this group was 27,83 years with a standard deviation of 3,98. The participants study architecture, computer engineering and software development. Some participants have strong prior knowledge of programming concepts. Five of the participants stated that they have experience with computer games.

6.2.2. Methodology

In preparation of the evaluation, a proper environment and test setup have to be established. The procedure during the evaluation is designed to cover our interests, gather as much information as possible and fit the limitations. The participants' activity is captured on-screen and through a think-aloud protocol. These recordings are analyzed retrospectively. Following the play-testing, the participant answers a usability questionnaire and an adapted question from Bebras (Dagienė, 2006). The evaluation is concluded with a short discussion to reflect on the experience.

Environment

For the camera tracking of the AR marker, it is important to have a well-lit environment. The participants shall be able to focus on the prototype and the evaluation procedure. Therefore a calm room was an additional requirement. During the evaluation, participants were observed and questions were answered. The evaluation of group A took place at the school of the volunteering participants. The school provided a classroom and one school lesson (50 minutes) per group for the evaluation. The participants of group B tested the prototype one by one in a quiet office environment at Graz University of Technology.

Test Device and Setup

For the evaluation of group A, a set of Samsung A6 smartphones was provided from the Institute of Software Technology. For group B, one device with similar hardware (Motorola Moto E4) and one high-end device (Samsung Galaxy S9) was used. This way, the functionality on different devices is verified. Two options for handling the prototype installation on the test devices have been considered: The Google Play Store Beta-test offers a simple roll-out functionality given the Google accounts of the test users. This option is reasonable for a large number of devices, which are not necessarily physically accessible. Additionally, updates can be installed in the same way as for regular apps. This Beta-test option can be useful for long-term tests, including updates or to reach a broader audience. At first, we considered that participants use their own mobile devices. In this case, the Play Store would facilitate the installation process. However, especially in a classroom setting the use of a personal mobile device could cause distraction. On the other hand, given access to the test devices, the app can be

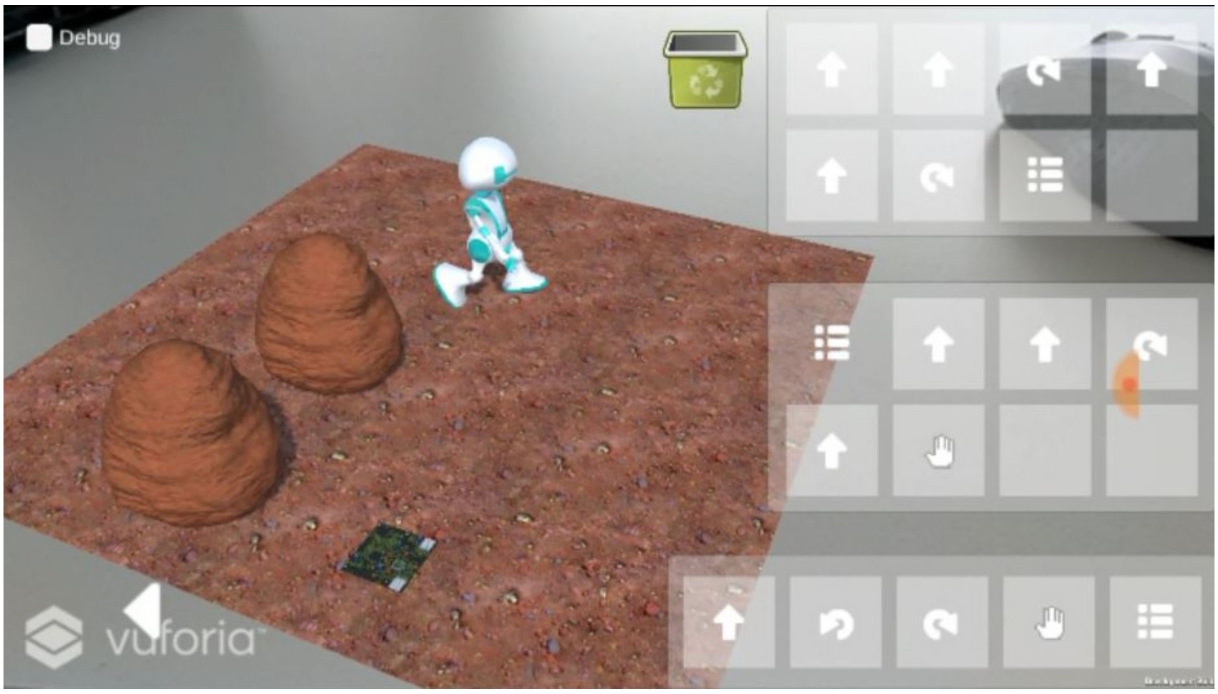


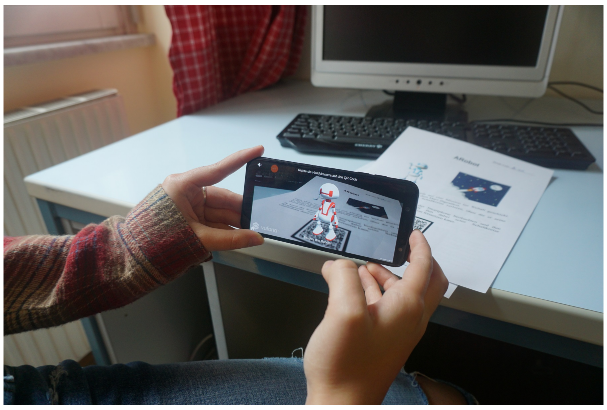
Figure 6.4.: In-Game view during the play-test.

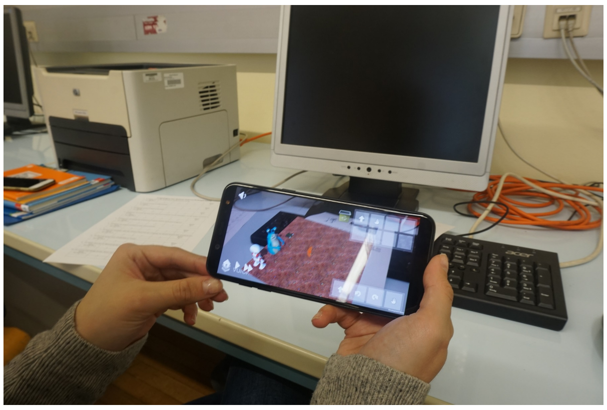
installed directly on the devices using Android's developer option. The latter process is fast and simple neglecting the play store publishing policy and the necessity of Google accounts. Only the standard and required apps are installed on the test devices, thus avoiding any unnecessary distractions. For the evaluation, the on-device installation was chosen, given the small number of test devices.

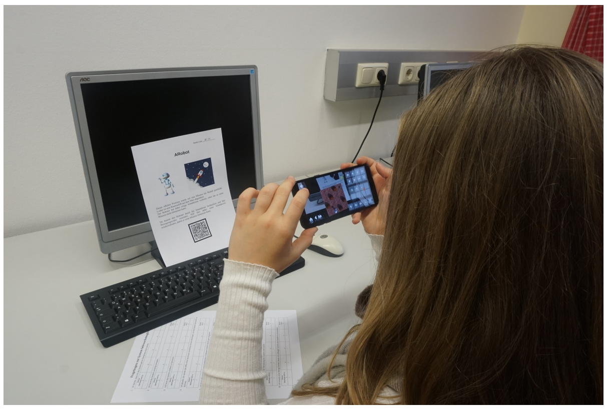
Procedure

The evaluation is structured in an introduction, the prototype test, a post-test questionnaire and a group discussion. As in the expert evaluation, the whole procedure was conducted in German.

1. The instructor introduces himself and the project.
2. The procedure of the evaluation is explained.
3. The participant receives an exercise sheet (see Appendix A).
4. The participant receives a test device.
5. A recorder app is started to record screen and voice
6. The prototype app is started, the play-test begins and arising questions are answered.
7. After the play-test, the participant fills the SUS questionnaire.
8. The participant answers an adapted multiple-choice question from the Austrian Bebras competition *"Biber der Informatik"* (see Appendix B)
9. The evaluation is concluded by a guided discussion (see Appendix C).


(a) Experimenting with the QR marker.


(b) Exploring a level.


(c) The posture during the play-test.

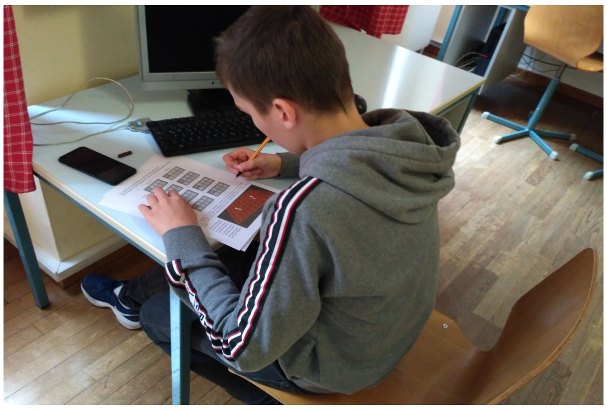

(d) Answering the on-paper task.

Figure 6.5.: Evaluation in a school classroom with group A.

Think Aloud and Screen Capture

Assessment of computational thinking is not trivial. Think-aloud protocol and on-screen recording of participants activity are suggested for examining computational practices (Lye & Koh, 2014). At the beginning of the play-testing, the participants are asked to speak out loud what they think while playing. Additionally, the screen is recorded as long as the game is played. An Android recorder app on the test device records screen and voice at the same time. This way, spoken thoughts can be easily related to the corresponding in-game activity.

System Usability Scale (SUS)

Good usability is necessary to ensure a good overall play experience. Especially for mobile games lack of usability can cause early drop-out and missing the actual game content. We wanted to ensure good usability as a starting point for discussing the results. To measure the usability of the prototype, we used the system usability scale (SUS) questionnaire[1]. It is a well-established usability questionnaire, which consists of ten question items on a Likert scale. The question items alternate between positive (odd) and negative (even) statements. The SUS score is calculated by adding up the zero-based (score minus one) positive item scores with the inverse of the negative item score (five minus score). This sum is normalized to a range between 0 and 100 (see equation 6.1).
Let q_n be the score of the n-th questionnaire item.

$$score = 2.5 * \sum_{i=1}^{5}[(q_{2i-1} - 1) + (5 - q_{2i})] \tag{6.1}$$

A higher score generally indicates better usability. There are various ways to interpret the SUS score (Sauro, 2018). The adjective rating scale (Bangor, Kortum, & Miller, 2009) allows a well-defined distinction, especially for higher scores. The adjectives range from *worst imaginable* to *best imaginable* and can be directly related to SUS scores. We use the adjective rating scale for the interpretation of SUS scores to distinguish between *ok*, *good*, *excellent* and *best imaginable*. A German version of SUS (see Appendix D) was used to avoid language barriers.

Adapted Question from Bebras

Bebras (see Section 2.1.2) challenges are short computational thinking tasks. As a printed task, we adapted one Bebras challenge, which targets the same CT concepts as our game prototype (see Appendix B). In the task, the participant has to choose a combination of movement signals for two distinct robots to meet in the same position. We expect the participants to understand the task after having played the

[1]German version of the system usability scale retrieved from https://experience.sap.com/skillup/system-usability-scale-jetzt-auch-auf-deutsch/

prototype. However, the answer modality is multiple choice instead of trial and error. The participants have no intermediate feedback nor an opportunity to correct their solution.

Informal Open Questions for Discussion

In the last part of the evaluation, the participants reflect on the experience and give feedback through a guided discussion. The students are asked the following questions (translated from German). While group A discusses these questions in the group, participants from group B answer them in a single interview. The discussion is recorded and analyzed afterwards.

- What did you expect from the game?
- How was your experience with the game?
- How would you describe the duration/difficulty of the levels?
- Would you create levels and share them with a level editor?
- What would you add?
- Did you think the game was motivating?
- How did you like the interaction with the camera and display?
- If you would play similar AR games on your mobile, where and when would you play them? (e.g. on the bus, at home)
- Do you think you learned something in the game? If yes, what?
- Would you like to add or comment anything else?

6.2.3. Findings and Discussion

The functionality of the prototype was well tested before the evaluation. Only one device caused unexpected failure and had to be exchanged for another test device. During the evaluation, we noted how differently the participants would approach the prototype. From the setup of the printed exercise sheet, the perception of the game, to their strategies of finding a solution. The SUS questionnaire results confirmed our usability requirement. The adapted Bebras challenge yielded insight into the differences between challenges in-game and on paper. In the concluding discussion, the participants reflected on their experience with the prototype. The think-aloud and screen capture analysis delivered rich information on each player's process of playing through the prototype.

System Usability Scale

The SUS questionnaire was well accepted by all participants. The ten items of the questionnaire were answered by everyone in less than five minutes. The mean SUS score of group A was 82, with a standard deviation of 9.7. This score is considered *excellent* on the adjective rating scale (Bangor et al., 2009). The individual scores range from 70 to 92.5 (see Figure 6.6).

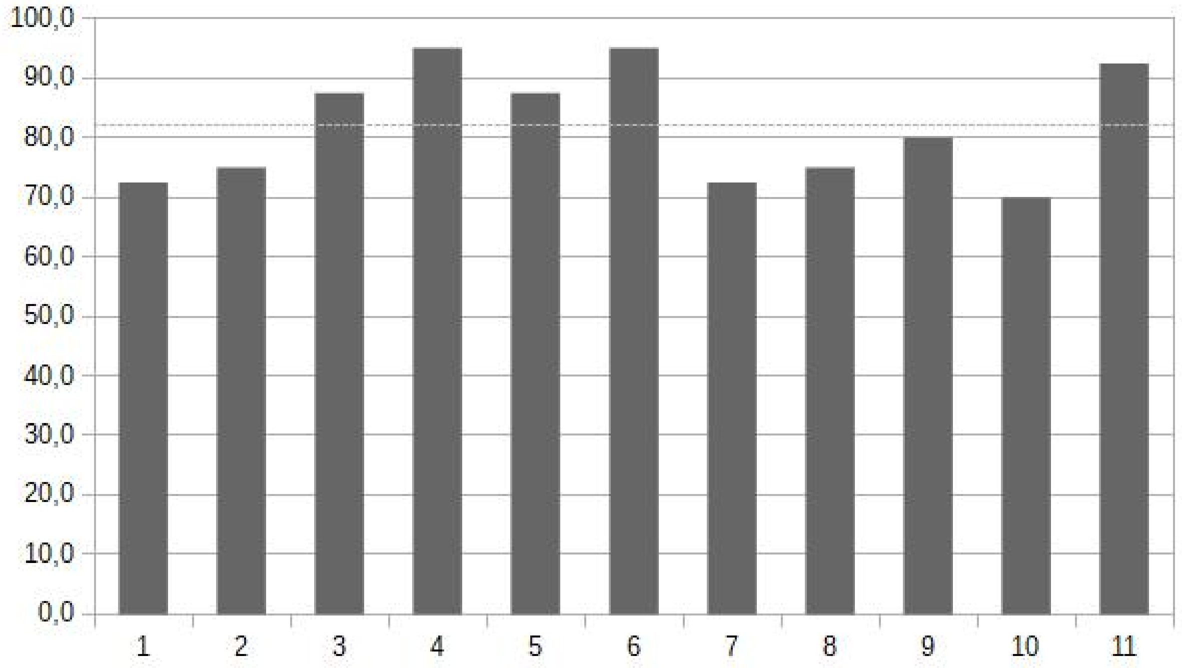


Figure 6.6.: The SUS scores of group A with an average score of 82 (standard deviation of 9.7).

The average SUS score of group B was 82.5, with a standard deviation of 9.2. This score correlates with the results from group A. The individual scores range from 65 to 90 (see Figure 6.7). As expected, participants of both groups had a similar experience regarding usability. Average scores from both groups indicate *excellent* usability with room for improvement. The lowest scores are considered between *ok* and *good* while the highest scores are considered *best imaginable*.

Adapted Question from Bebras

The multiple-choice question (see Appendix B) was supposed to transfer knowledge from the game to an on-paper task. The context was slightly different from the game, since two characters move at the same time according to two different signals. The apprehension of the new context took longer than expected. However, most participants seemed to have understood the task. Still only about half of the participants picked the correct answer option D. In group A it is visible that most of the participants in each gender group chose the same option (see Table 6.1). The participants were allowed to interact and help each other, which might have caused the monotonous results. The reason for the incorrect answers might be the missing feedback through trial and error. In the game, the participants could see the outcome of their solution and fix mistakes. A translation of this task into the game environment could lead to a higher rate of correct answers. The relation between in-game skills and paper tasks is complex and requires further research.

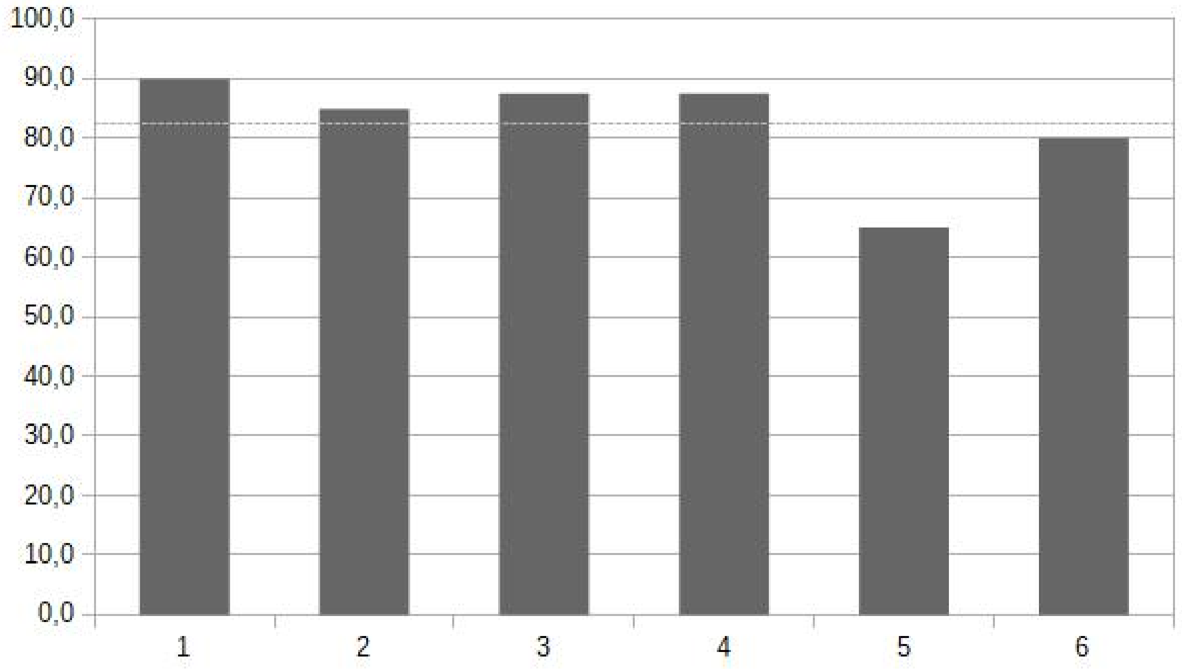


Figure 6.7.: The SUS scores of group B with an average score of 82.5 (standard deviation of 9.2).

Table 6.1.: Answers from all participants to the adapted question from Bebras.

Answer	A_f	A_m	B
A	-	-	-
B	1	-	2
C	5	-	-
D	-	5	3
no answer	-	-	1

Discussion

The participants expected a casual mobile learning game and were positively surprised by the marker-based augmented reality. They enjoyed the animated characters. The size and difficulty of the levels were well accepted;

"Not too long, not too short."

Some participants expected more levels and higher difficulty in the end. All participants could imagine creating their own levels, given a level editor. Most participants thought the game was motivating. The camera-marker interaction was mostly perceived very well;

"That was cool!" - "That was very good!"

When asked about the learning experience, some participants mentioned issues with distance and coordination. For them, it was difficult to estimate the number of steps in the planning phase. A highlighted visible grid could easily solve this problem. On the other hand, participants noted that they learned;

"A new way of thinking and strategic planning." - "Structured thinking."

The function panel was partially misinterpreted as a simple extension of the program panel. The loop component, on the other hand, was well understood to reduce the number of used blocks. A participant described that he has learned;

"To use fewer commands."

Some participants noted that they would have enjoyed rewards for finishing a level. Overall the participants had a positive experience and were very engaged in the game.

Screen Capture and Think Aloud Analysis

All participants used the customization option to change the robot's appearance. As expected, most participants quickly clicked through the textual introduction level. The first use of the QR marker detection, which shows the robot character caused a "wow" effect for many participants. Interestingly most participants had a very static top-down posture during the play-test. The marker was placed on the table and the device was positioned straight above the marker. This should be taken into consideration in future work with marker-based augmented reality. Perhaps a model-based approach would encourage the user more to move around the marker. In the first level, the participants began to explore the level elements and the user interface. After a few trials, the basics were mostly understood. The following levels needed fewer retries until the function tool was introduced. The thinking aloud revealed that many participants were confused by the newly introduced function part of the user interface. They used the function panel as an extension of the main program instead of reusing movement patterns (see Figure 6.4). Other participants experimented and learned how to use the

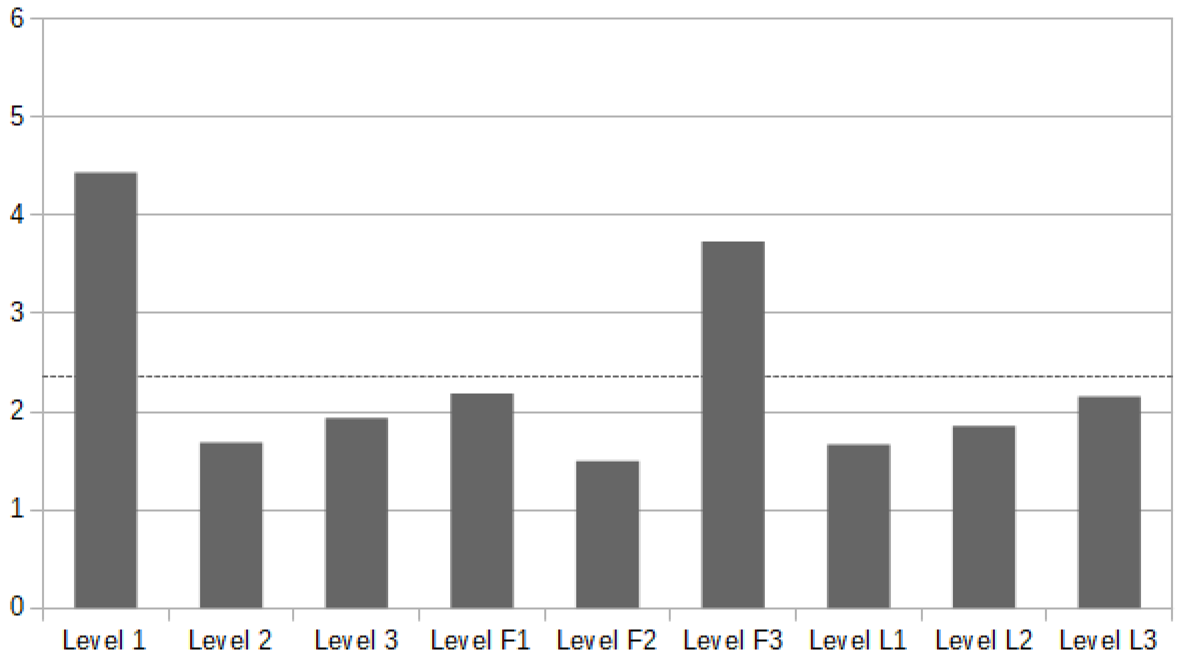


Figure 6.8.: Average attempts per level over all participants.

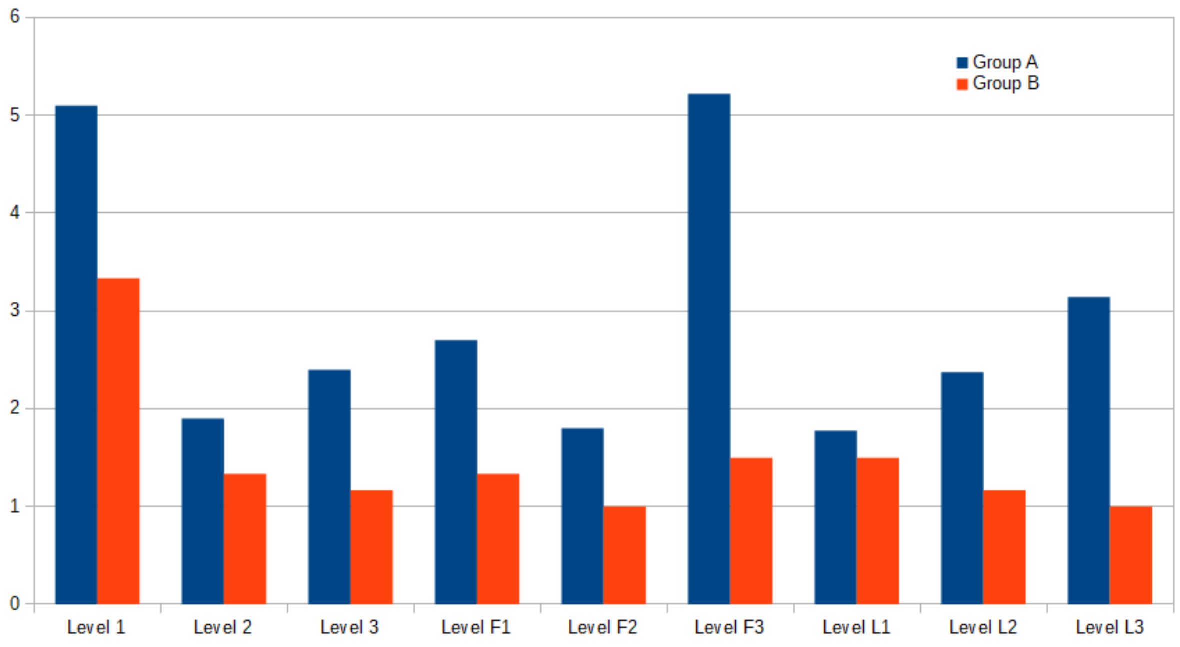


Figure 6.9.: Average attempts per level in group A and group B.

new tool quickly. Experimenting in the introductory levels helped to use the acquired knowledge in the following levels. This development can be seen in the number of retries of the individual levels (see Figure 6.8).

Surprisingly in the group evaluation, the participants copied not only solutions but also mistakes from each other. The paper bin, which was introduced after the expert evaluations, was very well accepted. Interestingly a few participants used it to delete the whole program when improving their code. All but three participants finished the nine levels in the provided time (20 minutes). The final screen, on which the robot gets back to his spaceship, was perceived very positive by most participants.

Limitations

The evaluation was designed for a small number of participants. The setup and procedure would need to be restructured for more participants. The expert evaluation showed that time is clearly a limitation. Firstly the evaluation per group was designed to fit in one lesson (50 minutes). Secondly, to keep the students engaged, it was important to keep the non-play time to a minimum. Exhaustive pre-test questionnaires could have decreased the overall motivation. However, the small number of participants allowed us to use think-aloud and on-screen recording. In group B the volunteering students had different schedules and the room was only available at certain times. Hence participants from group B had to test the prototype individually.

6.2.4. Summary

We designed and conducted an evaluation of the mobile augmented reality computational thinking game prototype ARobot. The expert evaluation helped to improve the prototype and the procedure for the final evaluation. Eleven students from a local school and six students from Graz University of Technology volunteered to take part. After the actual play-test, the participants evaluated the usability of the game. The overall usability score can be considered *excellent*.

After the play-test, the participants received an on-paper task, which was adapted from the Bebras challenge. About half of the participants answered correctly. The causality for the incorrect answers requires further investigation. However, the adapted Bebras challenge or similar tasks could be interesting for quantitative analysis in future work. A concluding discussion reflected upon the experience during the play-test.

The analysis of screen and think-aloud recordings showed that a guided introduction and someone to answer questions was helpful. The textual introduction was only skimmed by most participants. The support during the play-test ensured that mechanics and the user interface were properly understood. Participants from group A and group B approached the game in a similar way. However, the number of attempts per level (see Figure 6.9) supports our assumption that participants with programming knowledge (group B) solve levels with fewer attempts. The following chapter concludes the results from this evaluation and discusses how the gained insights can be used in future work.

7. Lessons Learned

The process of this thesis went through multiple stages. Each of these stages revealed some insights. This chapter reviews what worked out well and what can be improved.

7.1. Literature

Research in education provided various learning style models. These models are highly controversial. We prefer to speak of learning preferences instead of categorizing learners by learning styles. Publications on learning styles must be read critically. However, parts of the learning style models can definitely support a deeper understanding of the learner's perspective.

Literature about computational thinking in education is controversial too. Definitions of CT vary a lot, mostly depending on regional consensus. Hence the work building on these definitions has to be seen from different perspectives. Further, literature before Wing (2006) is very different from more recent work. Before 2006 CT referred mostly to skills gained from programming experience (Denning, 2017). Wing promoted CT skills as a useful tool for everyone and not only programmers. However, through the variety of possible application fields, there is a lot of interesting approaches and publications.

Augmented reality and mobile learning are rapidly evolving. Finding and analyzing literature that is not outdated is crucial. Besides, it is very difficult to find comparable literature, since most research settings and foci are fairly different. On the other hand, current mobile devices offer low-cost augmented reality with many possible application scenarios. This development generates new research possibilities and thus increasing up-to-date literature.

7.2. Design and Development

Design and development went through a few iterations, each one influencing the next. In an earlier iteration, the use of multiple markers was tested. It would be very interesting to interact in real-time with the AR content. However, the dynamic use of multiple markers with mobile AR is very difficult to handle for the user, as one hand is already holding the mobile device. The development with Unity and Vuforia worked very well. Both are well maintained and documented, including many community-driven resources. Unity's play-test environment inside the editor supported quick and simple feature testing. However, for testing in AR mode in a realistic scenario, the prototype build had to be uploaded on a mobile device.

Intermediate testing on various devices revealed minor issues, which could have gone unnoticed otherwise. For instance, not all devices have a standardized screen aspect ratio. When user interface elements scale automatically, they can be cut off or they are too small. The first case could make the user interface unusable. In the second case an empty margin fills the gap to the display edge. To avoid both cases, the user interface must scale according to the device's screen properties. The game content was not affected, as the content scales depending on the distance between camera and marker.

Another big issue was the overheating on some test devices. The simultaneous use of camera, CPU and graphic rendering resources produces a high load on the hardware. This transforms into heat, which at some point can make it uncomfortable to hold the device or can even crash the operating system. Optimizing the code and minimizing the work for the CPU and GPU creates less heat (and extends battery life). In the prototype, the number of polygons was reduced to tackle the issue.

The expert evaluation with users from the target group proved very helpful for the improvement of the user interface. The implemented feedback provided an enhanced user experience in the final evaluation. For instance, the paper bin, that was introduced after the feedback of the expert evaluation. It was very well accepted and used by everyone intuitively to adapt their solutions.

7.3. User Experience in Evaluation

The results from the system usability scale revealed that overall, the game's usability was perceived very well. However, there is still room for improvement. Some participants in the evaluation had issues to estimate block distances on the textured ground plane of the levels. The correct distances are essential for a correct solution. This issue did not occur in the expert evaluation. A highlighted visible grid could easily help to identify distances correctly and plan the correct amount of steps. Such a minor design change can imply a strong impact on the user's perception and further on motivation. Interestingly, minor animations that did not influence the game state were very well received.

Similar to experiences in related work, participants in the evaluation would have liked rewards in the game. Presumably, the participants from the target group are used to reward systems from other mobile games and applications. Therefore they expect to be rewarded and the absence of rewards might influence their experience negatively. Additionally, rewards could deal with sufficient but not optimal level solutions. Players could be encouraged to replay the level, to improve for a better score or a reward.

The timing in the expert evaluation gave a rough estimation of how much time is needed for the questionnaires and the play-test. A tight schedule with buffers was crucial for a smooth procedure. However, communication during the group evaluation was sometimes a bit challenging. Some questions during the play-test were relevant for all participants but had to be answered several times. The analysis of the screen recordings provided some insight into this issue. Players progressed through the game at different paces. While questions were answered, some players were too engaged in their own game situations. Because of that, they were not able to listen. An on-demand info tool could help (e.g. textual explanation or visual hints through long-press on a symbol). Furthermore, the group evaluation provided some insight into the group dynamics. While a single-player game was evaluated, the peers supported each other and exchanged their experiences during the play-test.

8. Conclusion and Future Work

Computational thinking is a broad field with lots of different understandings and definitions. Anyhow, it influences education globally for the near future. The literature research revealed an increase of different CT related projects in the last years. That shows that CT is getting more attention lately. However, there are many open questions.

8.1. Conclusion

The evaluation of our prototype was conducted to answer the research questions (see Chapter 6.2). The findings suggest that a mobile game is suitable when embedded carefully into a classroom setting;

- ◇ The game itself must match the learning content.
- ◇ The environment must be set up accordingly.
- ◇ The use of classroom devices is recommended to avoid distractions.

All participants engaged with the incorporated CT concepts in the prototype. However, the misinterpretation of the function tool needs to be addressed. Only an extended evaluation could deliver better insights on the profound understanding.

The analysis of the screen recording and the discussion were used to answer question **Q2**. The results showed that marker-based augmented reality in the prototype was very well accepted and deserves more attention. The attempt to answer question **Q3** yielded interesting insights and more questions. Approximately half of the participants chose the correct answer to the adapted question from the CT competition Bebras. The trial-and-error method in the game was helpful for all participants. Are the missing feedback and correction cycles the main reason for the incorrect answers on the paper task? On the other hand, it would be interesting to transfer this kind of assessment into the game. How would the users perform in-game with the possibility to iteratively improve their solution? How would this reflect on their motivation to keep learning?

Overall mobile learning and augmented reality are promising options to enhance traditional education.

8.2. Future Work

Building on the insights of this thesis, it would be interesting to analyze motivation and learning progress in the long term with a bigger audience. Therefore the evaluation setup and the prototype could be improved and adapted.

8.2.1. Evaluation

Overall it would be interesting to conduct further research with a bigger audience and more data. Then we could analyze differences and correlations of gender, age, ethnical background, game preferences and much more. The game could be distributed through the Google Play Store Beta option. Unity Analytics[1] could be easily integrated to extract useful data. Various different metrics, such as ″time spent in level″ and ″number of code changes″ could be analyzed efficiently. That data could provide quantitative results and insights with different demographics and devices.

Another interesting aspect to look at is the effect of peers in game-based learning. How are the progress and the engagement influenced, when a single-player game is played in a group such as in a classroom setting? Can an option to share in-game experiences online have similar effects?

8.2.2. Prototype

The game needs to offer more challenging levels and more features to keep players engaged over a longer time. Most participants liked the idea of creating levels on their own. This constructive approach could be exciting to deepen understanding of the learned concepts. Further, dynamically generated levels could be interesting as well. They must be solvable and adapt to the player's skill. Additional features such as bigger maps, moving level objects or spatial challenges could be introduced to keep the game challenging. The game could also include more CT concepts such as recursions. Rewards could further motivate players and keep them engaged. Further marker-less AR could make the game accessible without the printed marker. Finally, the game could offer different play modes to address various genres. This could help to engage players, who prefer the respective genres.

[1]Unity Analytics https://docs.unity3d.com/Manual/UnityAnalytics.html, accessed 2019-11-19

Appendix

Appendix A.

ARobot Exercise Sheet

ARobot-Name:________________
Alter: ____
Spielerfahrung(1 keine 5 viel): ____
Programmiererfahrung(1 keine 5 viel): ____

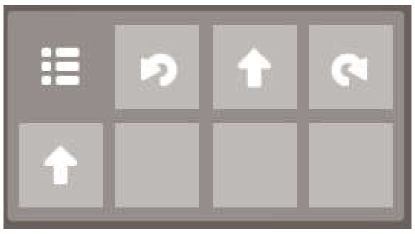
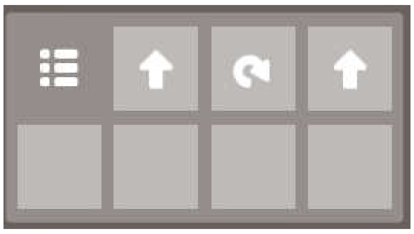
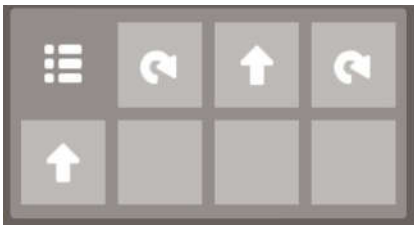
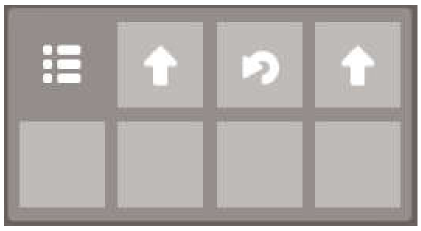
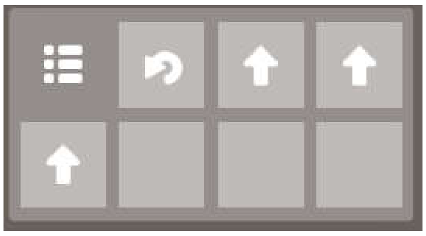
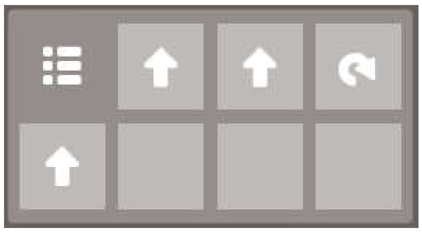
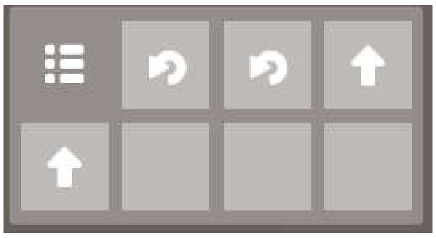
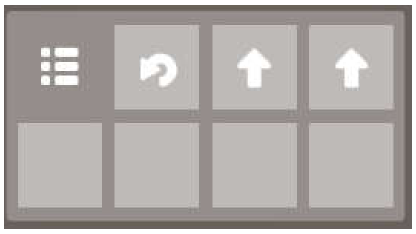
ARobot

Dieser ARobot Prototyp wurde auf eine Mission ins Weltall geschickt. Leider ist sein Raumschiff auf dem Mars abgestürzt.
Der Roboter hat dabei einige Schaltkreise verloren, ohne die er seine Mission nicht fortsetzen kann.

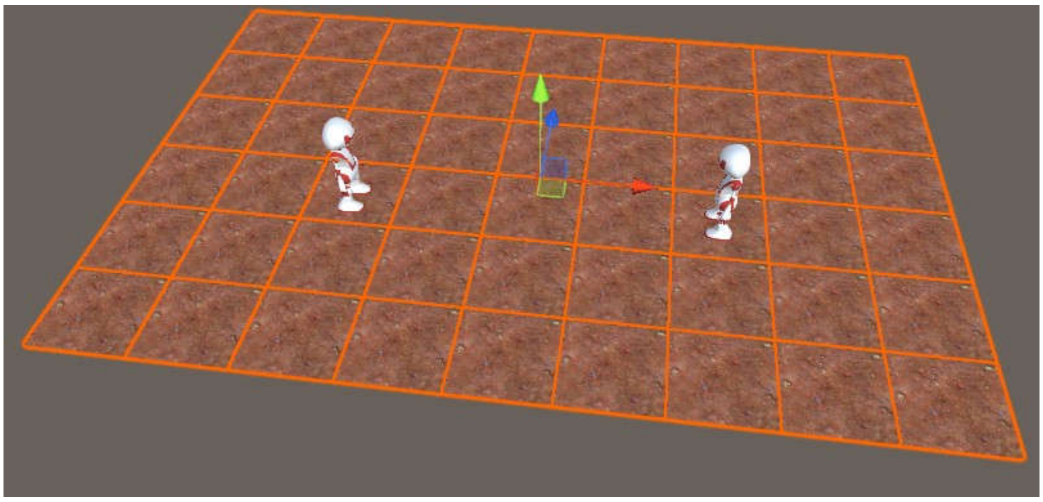
Du kannst den Roboter durch das Smartphone beobachten und ihm Bewegungscodes zusenden. Hilf ihm dabei seine Schaltkreise wiederzufinden, damit er seine Mission fortsetzen kann.
Besuche dazu die Levels 1-3 (Grundlagen), 4-6 (Funktionen) und 7-9 (Schleifen).

Appendix B.
Adapted Bebras Question

Es wurde ein zweiter ARobot auf den Mars geschickt.
Beide reagieren auf Funksignale, jedoch auf unterschiedliche Weise.

	Roboter 1	Roboter 2
Signal 1		
Signal 2		
Signal 3		
Signal 4		

Zum Beispiel bei Signal 1: Roboter 1 bewegt sich um 1 Feld nach oben und nach rechts. Roboter 2 hingegen bewegt sich ein Feld nach links und dann nach oben.
Welche Signale müssen gesendet werden, damit beide Roboter sich auf einem Feld treffen?



a) Zuerst Signal 4 und dann Signal 2

b) Zuerst Signal 1 und dann Signal 2

c) Zuerst Signal 2 und dann Signal 2

d) Zuerst Signal 3 und dann Signal 2

Appendix C.

Discussion

Diskussionsfragen ARobot

Einleitung

- Die Diskussion wird aufgenommen
- Es gibt keine falschen oder richtigen Antworten, alle Meinungen und Erfahrungen sind interessant
- Zur Verbesserung der App ist es wirklich wichtig, dass wir alle positiven und negativen Eindrücke von euch hören.
- Fragen von Diskussionsleitung, dann Gespräch untereinander
- Zu Beginn kurz etwas sagen (erfundener Name) damit wir Stimmen in der Aufnahme zuordnen können.

Zuerst möchte ich auf die App eingehen.
- Wie waren eure Erwartungen?
- Wie waren dann eure Erfahrungen?

- Wie würdet ihr die Länge/Schwierigkeit der Levels beschreiben?

- Könntet ihr euch vorstellen mit einem Leveleditor selbst Levels zu erstellen und diese zu teilen?
- Was würdet ihr noch zusätzlich einbauen?

- Fandet ihr das Spiel motivierend?

- Wie hat euch die Interaktion mit der Umgebung mit Kamera und Display gefallen?

- Wenn ihr ähnliche (AR) Spiele auf eurem Handy spielen würdet, wo/wann würdet ihr die spielen? (zB auf dem Weg zur Schule, zu Hause...)

- Glaubt ihr durch das Spiel etwas gelernt zu haben? Falls ja, was?

Abschließend wollte ich euch noch fragen, ob es noch etwas gibt was ihr gerne noch ergänzen würdet. Gibt es noch etwas was aus eurer Sicht was noch wichtig wäre, was wir beachten sollten, was man noch besser machen könnte oder generell etwas was ihr gerne noch einbringen möchtet?

Vielen Dank!

Appendix D.

System Usability Scale

German translation[1]

[1]Retrieved from https://experience.sap.com/skillup/system-usability-scale-jetzt-auch-auf-deutsch/

Fragebogen zur System-Gebrauchstauglichkeit

1. Ich denke, dass ich das System gerne häufig benutzen würde.

Stimme überhaupt nicht zu 1	2	3	4	Stimme voll zu 5
○	○	○	○	○

2. Ich fand das System unnötig komplex.

Stimme überhaupt nicht zu 1	2	3	4	Stimme voll zu 5
○	○	○	○	○

3. Ich fand das System einfach zu benutzen.

Stimme überhaupt nicht zu 1	2	3	4	Stimme voll zu 5
○	○	○	○	○

4. Ich glaube, ich würde die Hilfe einer technisch versierten Person benötigen, um das System benutzen zu können.

Stimme überhaupt nicht zu 1	2	3	4	Stimme voll zu 5
○	○	○	○	○

5. Ich fand, die verschiedenen Funktionen in diesem System waren gut integriert.

Stimme überhaupt nicht zu 1	2	3	4	Stimme voll zu 5
○	○	○	○	○

6. Ich denke, das System enthielt zu viele Inkonsistenzen.

Stimme überhaupt nicht zu 1	2	3	4	Stimme voll zu 5
○	○	○	○	○

7. Ich kann mir vorstellen, dass die meisten Menschen den Umgang mit diesem System sehr schnell lernen.

Stimme überhaupt nicht zu 1	2	3	4	Stimme voll zu 5
○	○	○	○	○

8. Ich fand das System sehr umständlich zu nutzen.

Stimme überhaupt nicht zu 1	2	3	4	Stimme voll zu 5
○	○	○	○	○

9. Ich fühlte mich bei der Benutzung des Systems sehr sicher.

Stimme überhaupt nicht zu 1	2	3	4	Stimme voll zu 5
○	○	○	○	○

10. Ich musste eine Menge lernen, bevor ich anfangen konnte das System zu verwenden.

Stimme überhaupt nicht zu 1	2	3	4	Stimme voll zu 5
○	○	○	○	○

Ludography

Niantic. (2016). *Pokemon GO*. Android, iOS. The Pokémon Company, Nintendo. Retrieved from https://pokemongolive.com/

Niantic. (2019). *Wizards Unite*. Android, iOS. Niantic. Retrieved from https://www.harrypotterwizardsunite.com

Radiant Games. (2017). *Box island*. Android, iOS. Accessed 2019-12-03. Radiant Games. Retrieved from http://boxisland.io/

Simply Projects. (2016). *Coddy: World on Algorithm*. Android. Accessed 2018-04-01. Simply Projects. Retrieved from https://play.google.com/store/apps/details?id=com.SimplyProjects.Coddy

Viana, R. (2012). *Cargo-Bot*. iOS. Accessed 2019-12-03. Two lives left. Retrieved from https://twolivesleft.com/CargoBot/

Yaroslavski, D. (2008). *Lightbot*. Code.org. Retrieved from https://lightbot.com/

Yaroslavski, D. (2017). *Spritebox*. Android, iOS. SpriteBox LLC. Retrieved from http://spritebox.com/hour.html

Bibliography

Akçayır, M. & Akçayır, G. (2017). Advantages and challenges associated with augmented reality for education: A systematic review of the literature. *Educational Research Review*, *20*, 1–11.

Amin, D. & Govilkar, S. (2015). Comparative study of augmented reality sdks. *International Journal on Computational Science & Applications*, *5*(1), 11–26.

Bangor, A., Kortum, P., & Miller, J. (2009). Determining what individual sus scores mean: Adding an adjective rating scale. *Journal of usability studies*, *4*(3), 114–123.

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.

Bauer, A., Butler, E., & Popović, Z. (2015). Approaches for teaching computational thinking strategies in an educational game: A position paper. In *Blocks and beyond workshop (blocks and beyond), 2015 ieee* (pp. 121–123). IEEE.

Berge, Z. L. & Muilenburg, L. (2013). Seamless learning: An international perspective on next-generation technology-enhanced learning. In *Handbook of mobile learning* (pp. 133–146). Routledge.

Blow, J. (2004). Game development: Harder than you think. *Queue*, *1*(10), 28.

Brennan, K. & Resnick, M. (2012). New frameworks for studying and assessing the development of computational thinking. In *Proceedings of the 2012 annual meeting of the american educational research association, vancouver, canada* (pp. 1–25).

Bricken, W. (1990). Learning in virtual reality.

Brockmyer, J. H., Fox, C. M., Curtiss, K. A., McBroom, E., Burkhart, K. M., & Pidruzny, J. N. (2009). The development of the game engagement questionnaire: A measure of engagement in video game-playing. *Journal of Experimental Social Psychology*, *45*(4), 624–634.

Chen, C. H., Ho, C.-H., & Lin, J.-B. (2015). The development of an augmented reality game-based learning environment. *Procedia-Social and Behavioral Sciences*, *174*, 216–220.

Coffield, F., Moseley, D., Hall, E., & Ecclestone, K. (2004). *Learning styles and pedagogy in post-16 learning: A systematic and critical review.*

Coon, D. & Mitterer, J. (2004). *Introduction to psychology*. Thomson Learning.

Csizmadia, A., Curzon, P., Dorling, M., Humphreys, S., Ng, T., Selby, C., & Woollard, J. (2015). Computational thinking-a guide for teachers.

Curzon, P., McOwan, P. W., Plant, N., & Meagher, L. R. (2014). Introducing teachers to computational thinking using unplugged storytelling. In *Proceedings of the 9th workshop in primary and secondary computing education* (pp. 89–92). ACM.

Dagienė, V. (2006). Competition in information technology–learning in an atrractive way.

de Ravé, E. G., Jiménez-Hornero, F. J., Ariza-Villaverde, A. B., & Taguas-Ruiz, J. (2016). Diedricar: A mobile augmented reality system designed for the ubiquitous descriptive geometry learning. *Multimedia Tools and Applications*, *75*(16), 9641–9663.

Denning, P. J. (2017). Remaining trouble spots with computational thinking. *Communications of the ACM*, *60*(6), 33–39.

Deterding, S., Dixon, D., Khaled, R., & Nacke, L. (2011). From game design elements to gamefulness: Defining gamification. In *Proceedings of the 15th international academic mindtrek conference: Envisioning future media environments, mindtrek* (pp. 9–15).

Dijkstra, E. W. (1979). My hopes of computing science (ewd709). In *Proceedings of the 4th international conference on software engineering* (pp. 442–448). IEEE Press.

Domínguez, A., Saenz-De-Navarrete, J., De-Marcos, L., FernáNdez-Sanz, L., PagéS, C., & MartíNez-HerráIz, J. J. (2013). Gamifying learning experiences: Practical implications and outcomes. *Computers & Education*, *63*, 380–392.

Dror, I. E. (2008). Technology enhanced learning: The good, the bad, and the ugly. *Pragmatics & Cognition*, *16*(2), 215–223.

Dünser, A., Walker, L., Horner, H., & Bentall, D. (2012). Creating interactive physics education books with augmented reality. In *Proceedings of the 24th australian computer-human interaction conference* (pp. 107–114). ACM.

eMarketer. (2015). Number of smartphone users worldwide from 2014 to 2020 (in billions). Retrieved from https://www.statista.com/statistics/330695/number-of-smartphone-users-worldwide/

Felder, R. M., Silverman, L. K. et al. (1988). Learning and teaching styles in engineering education. *Engineering education*, *78*(7), 674–681.

Finardi, K. R., Leao, R. G., & Amorim, G. B. (2016). Mobile assisted language learning: Affordances and limitations of duolingo. *Education and Linguistics Research*, *2*(2), 48.

Fleck, S., Hachet, M., & Bastien, J. (2015). Marker-based augmented reality: Instructional-design to improve children interactions with astronomical concepts. In *Proceedings of the 14th international conference on interaction design and children* (pp. 21–28). ACM.

Fleming, N. (2011). The vark modalities. *Online: http://vark-learn.com/introduction-to-vark/the-vark-modalities/.(accessed 15 April, 2015)*.

García-Peñalvo, F. J. & Cruz-Benito, J. (2016). Computational thinking in pre-university education. In *Proceedings of the fourth international conference on technological ecosystems for enhancing multiculturality* (pp. 13–17). ACM.

Giannakas, F., Kambourakis, G., Papasalouros, A., & Gritzalis, S. (2018). A critical review of 13 years of mobile game-based learning. *Educational Technology Research and Development*, *66*(2), 341–384.

Gouws, L. A., Bradshaw, K., & Wentworth, P. (2013). Computational thinking in educational activities: An evaluation of the educational game light-bot. In *Proceedings* *of the 18th acm conference on innovation and technology in computer science education* (pp. 10–15). ACM.

Goyal, S., Vijay, R. S., Monga, C., & Kalita, P. (2016). Code bits: An inexpensive tangible computational thinking toolkit for k-12 curriculum. In *Proceedings of the tei'16: Tenth international conference on tangible, embedded, and embodied interaction* (pp. 441–447). ACM.

Hanus, M. D. & Fox, J. (2015). Assessing the effects of gamification in the classroom: A longitudinal study on intrinsic motivation, social comparison, satisfaction, effort, and academic performance. *Computers & Education*, *80*, 152–161.

Hew, K. F. & Cheung, W. S. (2014). Students' and instructors' use of massive open online courses (moocs): Motivations and challenges. *Educational research review*, *12*, 45–58.

Huang, Y.-L., Chang, D.-F., & Wu, B. (2017). Mobile game-based learning with a mobile app: Motivational effects and learning performance. *Journal of Advanced Computational Intelligence and Intelligent Informatics*, *21*(6), 963–970.

Hwang, G.-J., Lai, C.-L., & Wang, S.-Y. (2015). Seamless flipped learning: A mobile technology-enhanced flipped classroom with effective learning strategies. *Journal of Computers in Education*, *2*(4), 449–473.

International Society for Technology in Education. (2016). Ct competencies. Retrieved from https://www.iste.org/standards/computational-thinking

Islam, N. & Want, R. (2014). Smartphones: Past, present, and future. *IEEE Pervasive Computing*, *13*(4), 89–92.

Javornik, A., Rogers, Y., Moutinho, A. M., & Freeman, R. (2016). Revealing the shopper experience of using a"magic mirror"augmented reality make-up application. In *Conference on designing interactive systems* (Vol. 2016, pp. 871–882). Association for Computing Machinery (ACM).

Juul, J. (2003). The game, the player, the world: Looking for a heart of gameness. In *Level up: Digital games research conference proceedings, edited by marinka copier and joost raessens* (pp. 30–45).

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

Kershner, R., Mercer, N., Warwick, P., & Staarman, J. K. (2010). Can the interactive whiteboard support young children's collaborative communication and thinking in classroom science activities? *International Journal of Computer-Supported Collaborative Learning*, *5*(4), 359–383.

Kirkwood, A. & Price, L. (2014). Technology-enhanced learning and teaching in higher education: What is 'enhanced'and how do we know? a critical literature review. *Learning, media and technology*, *39*(1), 6–36.

Kirschner, P. A. (2017). Stop propagating the learning styles myth. *Computers & Education*, *106*, 166–171.

Kirschner, P. A. & van Merriënboer, J. J. (2013). Do learners really know best? urban legends in education. *Educational psychologist*, *48*(3), 169–183.

Kojic, A., Kojic, M., Pirker, J., Gütl, C., Mentzelopoulos, M., & Economou, D. (2018). Scool - a mobile flexible learning environment. In *Ilrn 2018 montana: Workshop, long and short paper, and poster proceedings from the fourth immersive learning research network conference* (pp. 72–84).

Kolb, D. (1984). *Experiential learning: Experience as the source of learning and development*.

Koster, R. (2013). Theory of fun for game design:"o'reilly media. Inc.

Laporte, L., Zaman, B., & De Grooff, D. (2013). Exploring the value of genres in serious games.

LeBlanc, A. G. & Chaput, J.-P. (2017). Pokémon go: A game changer for the physical inactivity crisis? *Preventive medicine*, *101*, 235–237.

Lee, E., Shan, V., Beth, B., & Lin, C. (2014). A structured approach to teaching recursion using cargo-bot. In *Proceedings of the tenth annual conference on international computing education research* (pp. 59–66). ACM.

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.

Marcelino, M. J., Pessoa, T., Vieira, C., Salvador, T., & Mendes, A. J. (2018). Learning computational thinking and scratch at distance. *Computers in Human Behavior*, *80*, 470–477.

McLeod, S. A. (2017). Kolb - learning styles. Retrieved from https://www.simplypsychology.org/learning-kolb.html

Mitrasinovic, S., Camacho, E., Trivedi, N., Logan, J., Campbell, C., Zilinyi, R., ... Martineau, D., et al. (2015). Clinical and surgical applications of smart glasses. *Technology and Health Care*, *23*(4), 381–401.

Negroponte, N., Bender, W., Battro, A., & Cavallo, D. (2006). One laptop per child. In *Keynote address at national educational computing conference in san diego, ca. retrieved april* (Vol. 5, p. 2007).

Niemöller, C., Zobel, B., Berkemeier, L., Metzger, D., Werning, S., Adelmeyer, T., ... Thomas, O. (2017). Sind Smart Glasses die Zukunft der Digitalisierung von Arbeitsprozessen? Explorative Fallstudien zukünftiger Einsatzszenarien in der Logistik.

Nincarean, D., Alia, M. B., Halim, N. D. A., & Rahman, M. H. A. (2013). Mobile augmented reality: The potential for education. *Procedia-social and behavioral sciences*, *103*, 657–664.

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

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

Papert, S. (1999). What is logo? who needs it. *Logo philosophy and implementation*.

Partovi, H. & Sahami, M. (2013). The hour of code is coming! *SIGCSE Bull. 45*(4), 5–5. doi:10.1145/2553042.2553045

Piccione, P. A. (1980). *In search of the meaning of senet*. Archaeological Institute of America.

Pirker, J., Lesjak, I., & Guetl, C. (2017). Maroon vr: A room-scale physics laboratory experience. In *2017 ieee 17th international conference on advanced learning technologies (icalt)* (pp. 482–484). IEEE.

Pirker, J., Riffnaller-Schiefer, M., & Gütl, C. (2014). Motivational active learning: Engaging university students in computer science education. In *Proceedings of the 2014 conference on innovation & technology in computer science education* (pp. 297–302). ACM.

Plass, J. L., Homer, B. D., & Kinzer, C. K. (2015). Foundations of game-based learning. *Educational Psychologist*, *50*(4), 258–283.

Prensky, M. (2003). Digital game-based learning. *Computers in Entertainment (CIE)*, *1*(1), 21–21.

Rauschnabel, P. A., Brem, A., & Ro, Y. (2015). Augmented reality smart glasses: Definition, conceptual insights, and managerial importance. *Unpublished Working Paper, The University of Michigan-Dearborn, College of Business*.

Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., ... Silverman, B., et al. (2009). Scratch: Programming for all. *Communications of the ACM*, *52*(11), 60–67.

Riener, C. & Willingham, D. (2010). The myth of learning styles. *Change: The magazine of higher learning*, *42*(5), 32–35.

Robertson, S. & Robertson, J. (2012). *Mastering the requirements process: Getting requirements right*. Addison-wesley.

Rose, S. (2016). Bricolage programming and problem solving ability in young children: An exploratory study. In *European conference on games based learning* (p. 914). Academic Conferences International Limited.

Ryan, M.-L. (2009). From narrative games to playable stories: Toward a poetics of interactive narrative. *Storyworlds: A Journal of Narrative Studies*, *1*, 43–59.

Salen, K., Torres, R., Wolozin, L., Rufo-Tepper, R., & Shapiro, A. (2010). *Quest to learn: Developing the school for digital kids*. The MIT Press.

Sapounidis, T. & Demetriadis, S. (2013). Tangible versus graphical user interfaces for robot programming: Exploring cross-age children's preferences. *Personal and ubiquitous computing*, *17*(8), 1775–1786.

Sauro, J. (2018, September 19). Retrieved from https://measuringu.com/interpret-sus-score/

Seaborn, K. & Fels, D. I. (2015). Gamification in theory and action: A survey. *International Journal of human-computer studies*, *74*, 14–31.

Shea, R., Fu, D., Sun, A., Cai, C., Ma, X., Fan, X., ... Liu, J. (2017). Location-based augmented reality with pervasive smartphone sensors: Inside and beyond pokemon go! *IEEE Access*, *5*, 9619–9631.

Slany, W. (2012). A mobile visual programming system for android smartphones and tablets. In *Visual languages and human-centric computing (vl/hcc), 2012 ieee symposium on* (pp. 265–266). IEEE.

Sommerauer, P. & Müller, O. (2014). Augmented reality in informal learning environments: A field experiment in a mathematics exhibition. *Computers & Education*, *79*, 59–68.

Teng, C.-I. (2010). Customization, immersion satisfaction, and online gamer loyalty. *Computers in Human Behavior*, *26*(6), 1547–1554.

Tomi, A. B. & Rambli, D. R. A. (2013). An interactive mobile augmented reality magical playbook: Learning number with the thirsty crow. *Procedia computer science*, *25*, 123–130.

Vahldick, A., Mendes, A. J., & Marcelino, M. J. (2014). A review of games designed to improve introductory computer programming competencies. In *2014 ieee frontiers in education conference (fie) proceedings* (pp. 1–7). IEEE.

Vallance, M. (2017). Advancing computational thinking and knowledge development in a 3d virtual simulation.

Vera, L., Gimeno, J., Coma, I., & Fernández, M. (2011). Augmented mirror: Interactive augmented reality system based on kinect. In *Ifip conference on human-computer interaction* (pp. 483–486). Springer.

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

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

Yaroslavski, D. (2014). How does lightbot teach programming. *Retrieved January*, *29*, 2016.

Zarzuela, M. M., Pernas, F. J. D., Martinez, L. B., Ortega, D. G., & Rodriguez, M. A. (2013). Mobile serious game using augmented reality for supporting children's learning about animals. *Procedia computer science*, *25*, 375–381.