UNIVERSITY OF CALGARY

Towards Block-Based Programming Tools in Mixed Reality

by

Tarishi Upadhyay

A THESIS

SUBMITTED TO THE FACULTY OF GRADUATE STUDIES

IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE

DEGREE OF MASTER OF SCIENCE

GRADUATE PROGRAM IN COMPUTER SCIENCE

CALGARY, ALBERTA

MAY, 2020

ABSTRACT

Block-based programming languages have been used successfully to assist in overcoming the initial challenges for cultivating computational thinking for novices and non-programmers. Block-programming environments support project based learning that includes building and deploying program with the physical hardware to build creativity, computational thinking, and collaboration skills. However, traditional 2D block-based programming environments are often restrictive when it comes to practical and interactive learning, i.e. building projects and validating programs experimentally. This research focuses on enhancing the learning experience for novices using virtual 3D objects and mobile-based Mixed Reality. It aims to develop a new approach for learning block-based programming concepts, which offers real-time interaction with virtual 3D objects for practical implementation of programs and improved learning performance for novices. We designed and implemented a prototype that combines block-based programming with mobile MR. The prototype was designed in collaboration with teachers from STEM Learning Lab, who provided pragmatic insights on current practices and challenges with block-based programming. We evaluated the prototype using a comparative study with non-programmers and gathered feedback from teachers. The results indicate that the prototype and approach have the potential to improve the interactive experience and the learning performance of novices and non-programmers.

ACKNOWLEDGEMENTS

First and foremost, I would like to thank my supervisor, Dr. Frank Maurer, for your enthusiasm, advice, support, and patience. You have been a great supervisor to me, and your optimistic and warm nature motivates me.

I would also like to thank Dr. Lora Ohelberg and Dr. Nelson Wong for being a part of my supervisory committee. I am very thankful for your insightful recommendations and discussions.

I would like to acknowledge my colleagues and ASE Lab members for their support and encouragement.

I am grateful to Dr. Teddy Seyed for all your support and guidance. I am especially thankful to you for making me believe in myself when many times my self-confidence faltered. Thank you for being such a great mentor.

I want to thank my family. Thanks to my elder sister, Tanima Upadhyay, and my brother-in-law, Anshul Bhargav, for always being supportive. I'd also like to thank my parents and my in-laws for encouraging me to pursue this program and supporting me throughout my time here. Thanks for supporting me throughout my life, even when I did not want or ask for your support.

I thank my husband, Abhishek Sharma, for overcoming difficulties and support my study and research away. None of this would have come to fulfillment without your guidance.

CONTENTS

LIST OF TABLES

LIST OF FIGURES

ACRONYMS

STEM Science, Technology, Engineering and Mathematics

STEAM Science, Technology, Engineering, Arts and Mathematics

BBP Block-based programming

BBPE Block-based programming environment

BBPL Block-based programming language

VPL Visual programming language

MR Mixed reality

AR Augmented reality

VR Virtual reality

IDE Integrated Development Environments Vocational Educational Training

1

INTRODUCTION

With the increasing number of global initiatives to broaden and encourage science, technology, engineering, arts and mathematics (STEAM) careers, there has been a renewed interest in introducing and integrating programming concepts into educational curriculum and classrooms. Furthermore, the rise of automation and robotics in all sectors of production and management has created an immediate need for improving the quality and effectiveness of training in computer science and technology for both youth and adults alike [34]. A key challenge for many novice learners is to learn programming concepts in an engaging and intuitive manner [93]. The most frequently used (and general) approach to solve this challenge involves using visual and block-based programming languages, which are designed to abstract specific tasks (e.g. setting the value of a variable) from a traditional programming language (e.g. C++ or JavaScript) [93].

Prior research has shown that numerous factors contribute to making block-based programming easy for novice programmers, by including the natural language description of blocks, and the drag-and-drop interaction, but it has also been identified that block-based programming environments also have a perceived lack of interactivity [93] and are less powerful [97]. Block-based programming environments support a constructionist approach [67] that encourages the development of personal projects and artifacts like video games, animated

stories, or simulations of simple real-world phenomena [67]. Block-based programming languages such as LOGO[1] supports the idea of "creative creators" by Papert and his philosophy of "learning-by-making" where students learn by creating the programs and then building the project using the real-world objects [72].

Improvements in immersive technologies such as mixed reality (MR) have opened the door to creating more dynamic platforms for teaching coding. Research suggests mobile AR/MR may have potential in enhancing beginners' learning experience for programming, "especially for tasks that are more interactive and benefit from visual feedback" [26]. Combining block-based programming and mixed reality can provide more resources in the form of virtual objects resulting in more creativity and innovation. By manipulating virtual objects in real-time, learners can observe how the logic works, formulate hypotheses and validate them through experiments. It allows learners to connect real-life sensory experience with the digital environment [22].

1.1 BLOCK-BASED PROGRAMMING

One of the most commonly used Visual Programming Language (VPL) approaches today is *block-based programming*. Figure 1 shows lego-like blocks that are assembled in a sequence to create a program that can solve a specific task or function. These blocks are puzzle-like, and their shape enforces syntactically correct programs for users [10, 98]. The primary purpose of this type of programming language is not merely to arrange program chunks. Instead, its purpose is to let novices acquire knowledge related to the logical organization of an algorithm so that they can solve a certain task [54].

[1] https://www.media.mit.edu/projects/logo-blocks/overview/

Figure 1: Example of puzzle-like blocks to create syntactically correct programs[2]

Block-based programming environments (BBPEs) such as Scratch [80], MakeCode [62], Blockly [35] are widely used amongst educators and the wider STEAM education communities, and have shown tremendous value in lowering the barrier of entry for programming [28]. Block-based programming (BBP) is a recent addition to the extensive history of programming languages and environments that have been designed for novice learners. The first block-programming language was developed explicitly for children in 1995, but because of its simplicity and activities supported, its audience expanded beyond children and educators [97]. Learners with different backgrounds were encouraged to explore and create public, shareable artifacts, often in the form of artwork, games, and interactive stories [97].

[2]https://developers.googleblog.com/2019/01/scratch-30s-new-programming-blocks.html
[3]https://s4scoding.com/mit-app-inventor-2-introduction-to-android-app-development/visual-programming-language-blocks/

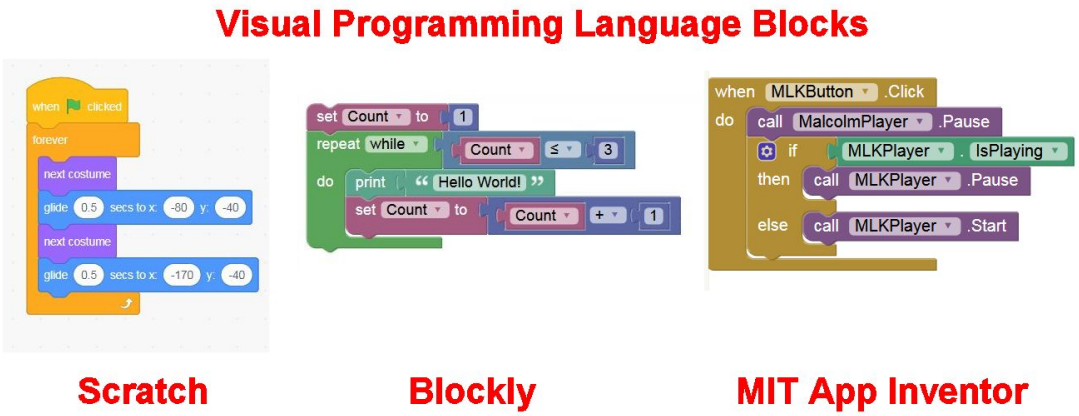


Figure 2: Example of block-based programming tools, such as Scratch, Blockly and MIT App Inventor[3]

One of the key aspects of block-based programming is its readability, where code is organized in blocks, helping readability [55]. This is because, for simple commands, blocks can forgo the punctuation that text code uses to denote structure and use descriptive words instead. For example, Figure 3 explains that a simple call in Python requires reading delimiters and knowing argument order, whereas the equivalent block in a language like Scratch, reads naturally using everyday words to explain its behaviour [11].

Previous research has even shown that people who have no experience with application development can better understand the basic concepts of programming through block-based programming and can even begin creating algorithms using its approach [54]. It has also been shown that learning a block-based language can later improve the learning of a traditional text-based language (e.g. Python or JavaScript) [11].

[4]https://www.kidsakoder.no/faq-category/kodetimen-5/

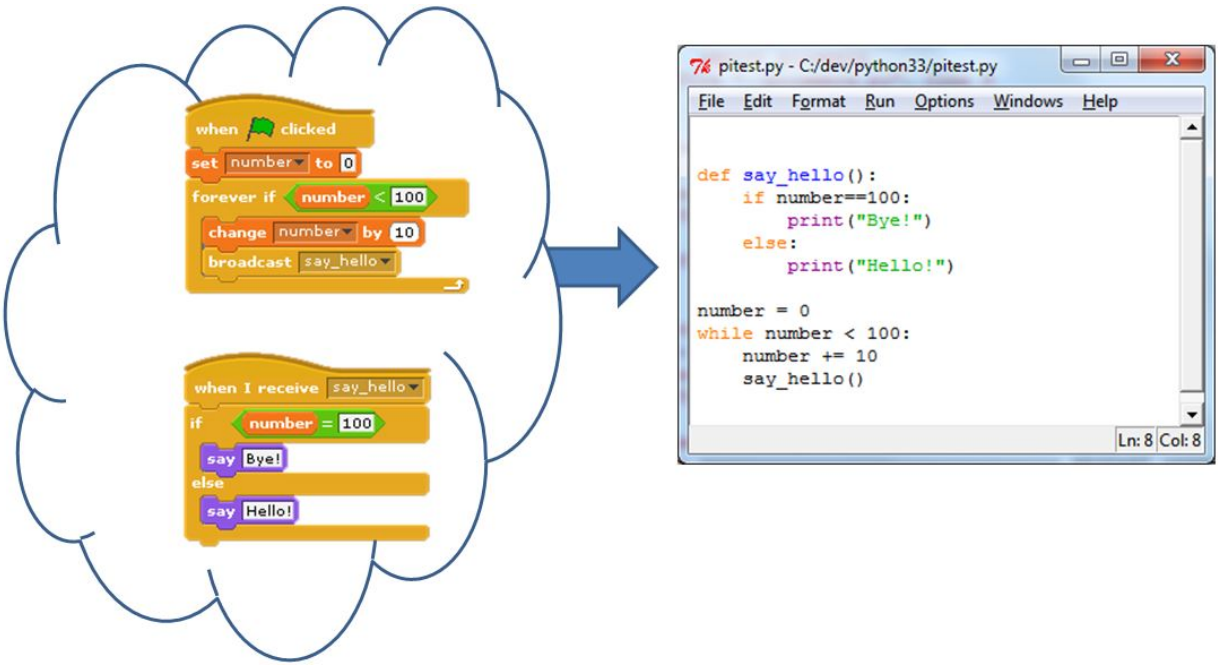


Figure 3: Scratch blocks compared with Python code[4]

From the perspective of educators today, learning to program with block-based programming helps with creativity, innovation, and the development of computational thinking, thus setting them up for future career opportunities [104]. For this reason, many introductory Computer Science courses and classrooms use blocks before the text-based approach. "In Harvard's CS50, students move from Scratch to C; Berkeley's CS10 progresses from Snap! to Python; Project Lead The Way's Computer Science Principles (CSP) course uses both Scratch and App Inventor before moving on to Python, and Code.org's CSP App Lab course moves from Droplet blocks to JavaScript" [11].

Research has also indicated that the lightweight style of work in block-based programming environments (BBPEs) stimulates and motivates students, allowing them to concentrate mainly on the algorithm and script components of a project [34]. This considerably develops their logical and algorithmic thinking and frees them from the complexity associated with syntax errors and other areas that may be perceived as difficult (e.g. compilation).

1.2 MOBILE LEARNING

Mobile-based learning has been defined as: 'learning across multiple contexts, through social and content interactions, using personal electronic devices' [23]. It is concerned with learners' mobility to learn outside a classroom or at different locations. It requires the motivation to learn wherever the opportunity appears – from books, electronic resources, places and people [47]. Narrowing learning content to make it accessible on mobile devices might be the most intuitive approach for mobile learning. Such applications can have their benefits, in particular, to reach distant and mobile learners.

Mobile devices become more pervasive today, they are getting more affordable [24]. Many people from various backgrounds, including students and teachers from educational institutions, are using it. It is a form of media technology that is used to practice learning activities through the use of mobile devices. Currently, many institutions in the educational sector, either formal or informal, are adopting mobile learning. In this regard, students are the most potential users of mobile learning. There are several benefits that mobile learning can offer to students. Firstly, it can provide an enjoyable environment in learning so that students are motivated to improve their learning performance. Moreover, it can provide the students with a flexible time in their learning activities. Lastly, students have a choice to study and gain practical understanding either in the class, outside class, or a combination of the two [37].

There are three dimensions of mobile learning adoption, i.e. "mobility of technology, mobility of learners, and mobility of learning" [30]. First, the mobility of technology implies that the devices used to deliver learning content should have the ability to connect to the Internet in order to access the information anywhere and anytime by students. Second, the mobility of learners means that the learning applications should support the students' mobility as

Figure 4: Students learning with Tivitz mobile app[5]

well as the freedom to access the learning material based on their needs. Third, the mobility of learning means that the learning content should also be modified to be more suitable for mobile devices [37].

Research indicates that the full potential of mobile learning will not be realized until we stop producing learning apps or mobile websites that simply repackage classroom materials to be read or played with on a smaller screen [75]. There is plenty of learning content available to access by mobile devices. It can be in the form of video, audio, text, and images that offer students various methods of learning. Mobile learning could improve the effectiveness and accessibility of learning activities in the future [84]. Mobile AR/MR platform may augment learners' interaction experience by focusing on practical learning and providing solutions for abstract concepts that are difficult to understand in a 2D interface, without the restrictions of fixed locations [25].

[5]https://www.tivitz.com/teach-with-tivitz-in-the-classroom/

1.3 MOBILE LEARNING WITH AUGMENTED/MIXED REALITY

Augmented reality (AR) can be defined as a system or visualization technique that fulfills three main features: (1) a combination of real and virtual worlds; (2) real-time interaction; (3) accurate 3D registration of virtual and real objects [7]. Researchers have suggested that AR/MR technology, combined with education, provides learners with a more immersive and engaging learning environment without decreasing the authenticity of the real world [56].

The integration of AR/MR and education has already shown positive educational effects which have resulted in a higher level of motivation, learning performance, engagement and collaboration among students [8]. Mathematics [49], science [21], geography [83], history [79],and astronomy [32] are a few subjects where studies have been conducted to realize the significance of AR/MR in teaching abstract concepts [78]. Figure 5 highlights examples of augmented/mixed reality in the field of education.

AR has also been used to augment various mediums, such as books, games, training modules, or object modelling. In classrooms, an engaging experience can be created using AR, where students can be made to visualize concepts which are otherwise difficult to imagine (e.g. the solar system or the human body). Positive education effects, such as improved performance and increased motivation by students, have been reported in various studies, ultimately indicating the significance that AR/MR might hold soon [78].

With the interactive and engaging characteristics of AR/MR, combined with the ubiquity and portability of the modern smart-phone, the number of mobile AR/MR apps is rapidly increasing, allowing children and adults to explore AR/MR and provide new interactive learning experiences across numerous domains and application scenarios. This technology, in

[6]https://zealar.com.au/magic-happens-when-augmented-reality-meets-education-sector/ https://www.rmit.edu.au/industry/develop-your-workforce/tailored-workforce-solutions/c4de/articles/not-a-gimmick-ar-use-in-business

Figure 5: Examples of Augmented/Mixed Reality in education[6]

combination with mobile devices and wireless connectivity, enables learning anywhere and at any time [25].

[8] indicates that AR has been mostly applied in higher education settings and compulsory levels of education for motivating students. Target user groups such as early childhood education and Vocational Educational Training (VET) are potential groups for exploring the uses of AR/MR. [8] has also shown that AR is effective for: a better learning performance, engagement and positive learning attitudes for people within all age groups.

1.4 MOTIVATION

BBPEs are frequently used to introduce novice learners to concepts in programming, electronics and IoT [28, 53]. BBPEs and their languages (BBPLs) have not only engaged a diverse range of people of all ages, especially youth, but they have also opened computer programming and other areas to a broader global audience in fields such as gaming [63], App development [102], IoT [9], machine learning [46].

However, one key challenge with existing BBPEs (regardless of the application domain) is that traditional methods of interaction with blocks in a 2D, mouse-based environment restricts learning and prevents a learner from being immersed in the learning environment [93]. For example, a novice learner may not get an interactive learning experience, which allows them to learn and interact with learning material (e.g. microbit, cardboards, cables etc.) in a more natural way [26]. In several classrooms today, electronics (e.g. the BBC Microbit [61] or the Adafruit Circuit Playground Express [1]) are often combined with BBPEs to teach different computing concepts, and these challenges are further magnified in scenarios when physical electronics are used alongside BBPEs. For example, depending on the socio-economic background of a classroom or its students, costs and accessibility can be challenging

when working with electronics [85, 89]. This frames the application domain for this thesis, where I focus on block-programming environments as an educational tool.

Ultimately, there has been a substantial rise in the use of BBPEs and BBPLs in several of the previously mentioned fields, but little work has been done in examining how learners, and their teachers, currently perceive of block-based programming environments and what they see as the benefit of the approach relative to the more traditional text-based alternatives [97]. As prior work has already shown some challenges, as previously mentioned, combining mobile learning and mixed reality (MR) — which already has demonstrated value in numerous educational contexts [3, 42] — is a potential solution for creators of block-based programming tools. This combination can potentially overcome interaction challenges, provide a more immersive learning experience [95] and a novel way that allows discovery and learning with the virtual objects and digital information augmented onto 3D objects in MR, as well as address the resourcing issues for electronics that are frequently used in classrooms.

In this thesis, I focus on the following interactions for the MR prototype :- (1) the interaction between the student and the learning content (with the learning content augmented as a layer superimposed on virtual objects), and (2) interaction between the students and learning aids (with the presentation of virtual objects and an interface to build the projects and run the program on it). It enables learners to identify solutions to problems through digital data and virtual objects in MR [22].

1.5 RESEARCH QUESTIONS

The first step in providing creators of block-based programming tools in mixed reality with design considerations for educational environments, is to understand the existing research space. This provides the first research question for this thesis:

1. *What is the current state of research in block-based programming environments?*

Understanding existing block-programming environments will help researchers and practitioners learn about various block-based programming approaches for different domains, and audiences with different goals.

Given the focus on education for practitioners, it is also important to understand the current trends, practices and challenges with block-based programming tools in classrooms. This leads to the following question:

2. *What are the challenges with existing block-programming environment from an educators' perspective?*

After answering the first and second question, a set of design considerations should be defined that can be applied for implementation of the potential solution (combining block-based programming and mixed reality) that enhances block-programming learning experience and solves the existing challenges, which leads to the following question:

3. *What are the key design considerations for tools that combine block-based programming and mixed reality?*

Building upon the previous question, it is also important to understand the impact this combination can have in learning basic programming concepts, which leads to the following question:

4. *What is the impact of MR-based mobile app on the performance of a non-programmer while learning programming concepts?*

5. *What challenges and opportunities does a system that combines block-based programming and mixed reality face?*

These research questions are answered by findings from observational studies, multiple semi-structured interviews, a critique of an implemented system and a comparative study, all of which are discussed in the remainder of this thesis.

1.6 THESIS CONTRIBUTIONS

The contributions of the work discussed throughout this thesis for mixed reality and block-based programming in education, are as follows:

1. A comprehensive summary of the research space of block-based programming environments and an exploration of the challenges that novices face with existing block-programming applications and what are the potential solution for these challenges, such as augmented/mixed reality.

2. A set of studies — an observation study and semi-structured interviews — providing a set of design considerations that can be applied when implementing a system that combines block-programming and mixed reality, to enhance learning experiences for novices, and an implementation that utilizes these design considerations.

3. A comparative study - that helps to understand the effectiveness of an MR-based mobile app compared to a non-MR based mobile app for learning block-based programming and a preliminary design critique and discussion of combining mixed reality and block-based programming. This could serve as a starting point for educators to select a BBPE to teach programming.

1.7 RESEARCH GOALS

This thesis has two primary research goals. The first is to provide an overview of the existing research on block-based programming and mixed reality. This overview provides a structuring of the current research space by elucidating each space and highlighting the necessary areas of research. The second goal is to provide creators of block-based programming tools in mixed reality, a set of design considerations from the perspective of educators who may use their tools in classrooms. This goal is addressed with two qualitative studies, as well as an example of an application that was critiqued by teachers. A comparative study illustrates the effectiveness of the MR-based mobile app in comparison to a non-MR based mobile app to learn and teach block-programming. The study elucidates the impact of MR-based mobile applications on the performance of non-programmers for learning programming.

1.8 THESIS STRUCTURE

This introductory chapter provided a concise background of the research for this thesis. Additionally, the research topic and questions, as well as goals were discussed, followed by an outline of the thesis contributions. The remaining chapters for this thesis are organized as follows:

Chapter Two: An Overview of Block-Based Programming - provides an overview of related research into programming challenges for novices, followed by a summary of block-based programming and how it solves some of these challenges, and lastly, how immersive environments have been used for learning in classroom settings, and the research gap this thesis addresses.

Chapter Three: Designing an Mixed Reality, Block-Based Tool for Learning Programming – describes a set of studies that lays the groundwork in block-based programming and education by interviewing teachers and generating design guidelines. Also discussed is an implementation that utilizes these guidelines.

Chapter Four: Evaluating Block-Based Programming Tool in MR - describes a comparative study that evaluates our approach by comparing the test results from the group of non-programmers using the MR-based mobile app against those using the non-MR based mobile app to learn programming. It also discussed the challenges and opportunities of the tool by gathering feedback from teachers, to guide future work in building mixed reality and block-based programming tools. The chapter concludes with the limitations of the study described in this thesis.

Chapter Five: Conclusion and Future Work – discusses the conclusion of the research work in this thesis, its contributions, as well as future research that can be built upon the research presented in this thesis.

2

RELATED WORK

Block-based languages are used to gradually introduce programming concepts to novices, helping in overcoming the initial challenges encountered for cultivating computational thinking. Visual programming languages (VPLs) and their environments are designed to reduce the complexity of programming for inexperienced users and novice programmers [19]. One of the most popular approaches within VPLs is *block-based programming* that allows the design of programming algorithms (program logic) using drag and drop of program chunks, called blocks [54]. Block-based programming environments (BBPEs), have been shown to help prevent most syntactic errors [13] and are generally considered significantly easier to use than text-based alternatives [97, 99].

The integration of immersive environments — such as virtual reality (VR) and mixed reality (MR) – with visual programming is a relatively new research area. Recently, Khalid et al. explored the impact of using a 3D visual programming tool on student's performance and attitude [4]. The findings of their study indicated that their 3D programming tool had a positive impact on students attitude towards programming, and it was effective in improving student performance and outlook towards computer programming. In addition to being used in the computer science and educational contexts, there has also been a sharp increase in the number of different domains that have adopted the block-based programming paradigm to

lower the entry barrier for programming. Some of these domains include tools for modelling and simulation [12, 94], mobile app development [39, 86, 100, 102], artistic tools such as [16, 88], and even game-based learning environments [64, 96]. Additional work is also being done to make it easier to develop an application or task-specific block-based programming languages. The diversity and modality of block-based programming tools highlight the importance of uncovering its current challenges and use-cases for block-based programming. For example, one common challenge is centred around the presentation of blocks in a block-based programming environment. Prior research has shown that students often struggle in the search for blocks, with category switching making up an average of 16% of their total programming actions [77]. Little research has examined approaches in dealing with this challenge in the scope of education or other domains.

In this section, we discuss the existing research into visual and block-based programming languages, how block-programming languages work, how Mobile AR/MR is currently being used for learning, followed by how block-based programming languages have been combined with AR/MR to-date.

2.1 Visual and Block-based Programming Languages

Programming concepts presented to novices and students are often an abstract method that seems uninspiring [4]. The use of complex logic in teaching programming concepts makes it difficult for them to learn programming and pushes them to lose interest and confidence in learning programming [4].

Textual programming environments (Figure 6) — which until recently have been the primary mode of programming — have paved the way to visual programming languages.

They have evolved into environments that allow programming based on the use of readily available program blocks, and in some cases, the real source code remains hidden [54].

```c
/*
 * This line basically imports the "stdio" header file, part of
 * the standard library. It provides input and output functionality
 * to the program.
 */
#include <stdio.h>

/*
 * Function (method) declaration. This outputs "Hello, world\n" to
 * standard output when invoked.
 */
void sayHello(void) {
    // printf() in C outputs the specified text (with optional
    // formatting options) when invoked.
    printf("Hello, world!\n");
}

/*
 * This is a "main function". The compiled program will run the code
 * defined here.
 */
int main(void)
{
    // Invoke the sayHello() function.
    sayHello();
    return 0;
}
```

Figure 6: Example of textual programming environment[7]

Today, computer science educators are aware of students' low tendency towards programming and have explored numerous tools, techniques and methods to motivate students to learn programming and to make computing appealing and accessible to more students [4]. Governments and education ministries in many countries have also shown renewed interest in

[7]https://en.wikipedia.org/wiki/Programming_language

integrating programming into the curriculum of primary education. This curriculum not only introduces computer science terms to children, but also the basic concepts of programming and the creation of computer programs [82]. One of the most popular methods to introduce programming concepts is through block-based programming languages and is currently used by millions of people of all ages and backgrounds, all around the world [57].

2.1.1 *History*

Traditional programming with textual languages was difficult for kids and novices to process. Therefore, the MIT media lab presented the concept of block-based programming and implemented the first block-programming language, Logo Blocks, in 1995 [59].

The concept was to create an interface to represent the logic and complex programming concepts, so children could learn to program faster by merely dragging and dropping puzzle blocks. Using this new method, the MIT Media Lab developed the first version of Scratch in 2003 for teaching and learning computer science concepts. Block-programming languages like Scratch are explicitly designed to help kids, and other non-technical novices learn how to write a computer program. This simple idea removed the necessity to learn the syntax of a programming language. Today, Scratch inspires many of the current block-based programming languages, and these languages help to teach children basic programming concepts with fundamental elements required in traditional programming and develop their interest in programming. Furthermore, this made it accessible for novices with no formal coding background to learn the basic concepts of programming.

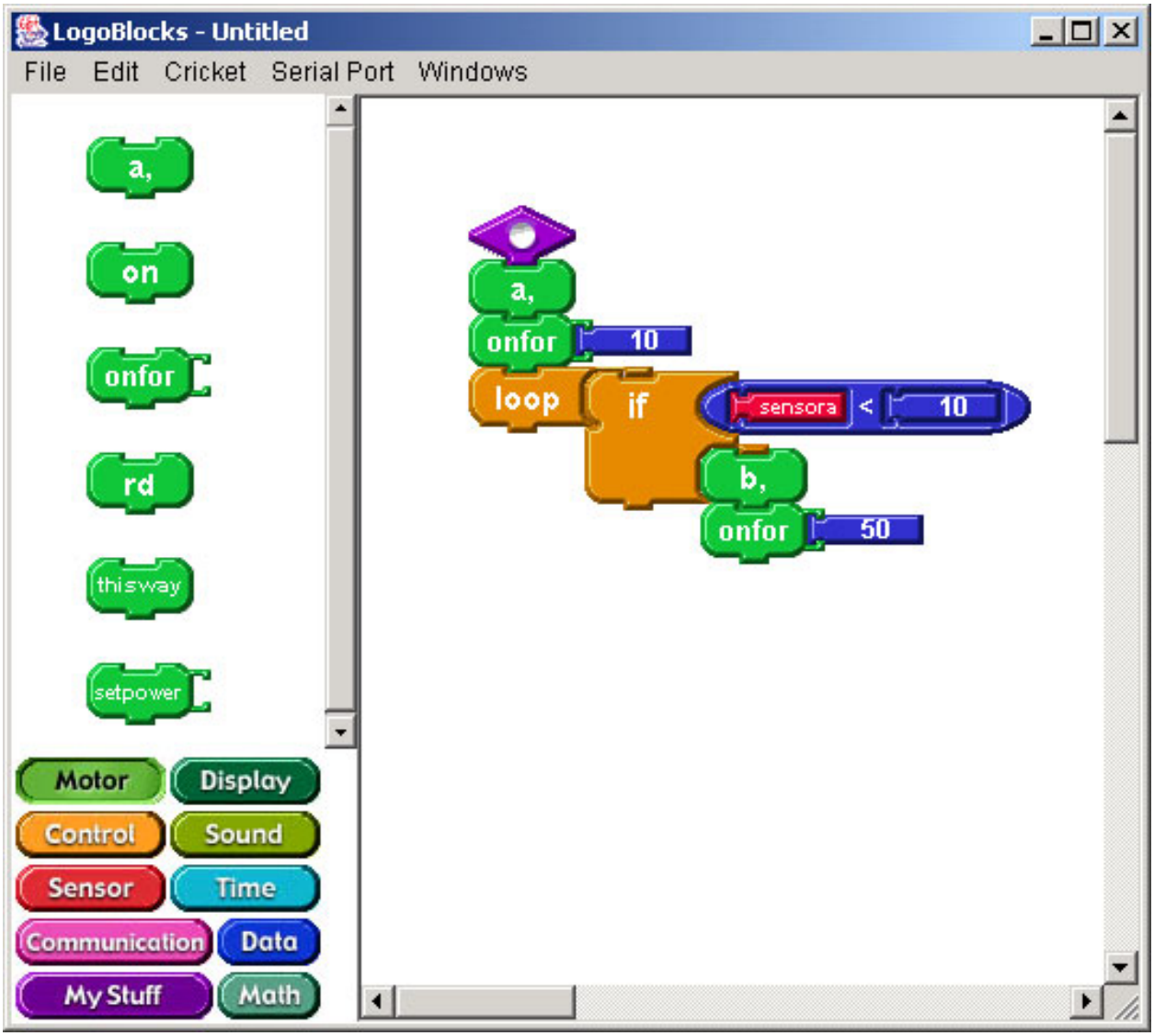


Figure 7: Logo Blocks interface[8]

2.1.1.1 *Logo Blocks*

Logo Blocks was developed as a visual analogue to BrickLogo in 1995 at the MIT Media Lab and was inspired by the desire to make the programming language more user-friendly. Logo Blocks met the challenge of building a programming language for kids (and novices) that

[8]https://learning.media.mit.edu/projects/gogo/gogo22/logoblocks.jpg

is both robust and easy to use. Logo Blocks breaks the code into small chunks of graphical blocks that may be dragged around the screen. Figure 7 shows the Logo Blocks interface that includes a palette of blocks with all possible blocks that are available, and users may choose different options (e.g., which motor port to turn on) by clicking the "ON" block.

2.1.1.2 *Scratch*

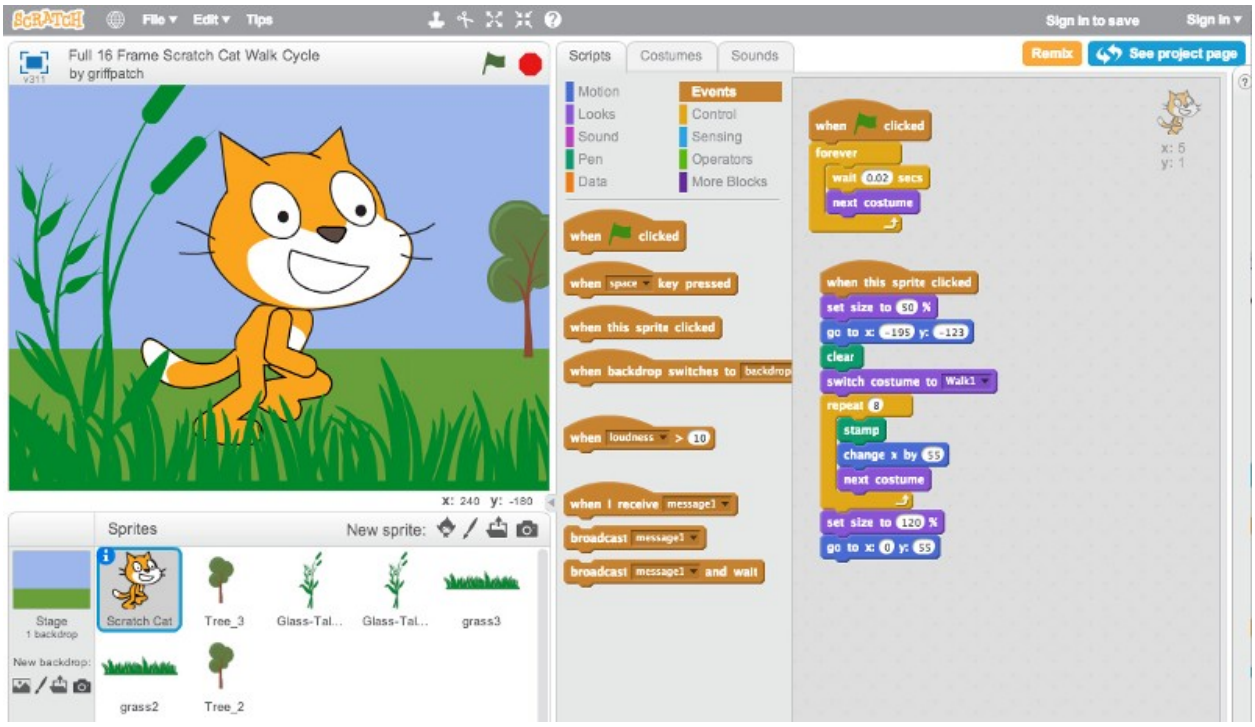


Figure 8: Scratch interface to create interactive animations and stories[9]

MIT's Lifelong Kindergarten [10] group was inspired by Logo Blocks and developed Scratch for young people to teach them computational thinking while making it exciting and fun at the same time. [80] It uses a building-block visual interface to create a scaffolding experience for novice programmers ("For example programmable sprite - a two-dimensional animation as

[9]https://about.vidcode.com/lovemyvidcode/2018/3/26/what-to-learn-after-scratch
[10]https://www.media.mit.edu/groups/lifelong-kindergarten/overview/

shown in Figure (8) contains multiple coded scripts which are programmed to run a sequence of operations which are activated and executed by sprite each time a specific event occurs" [81]). Users stack together programming elements, such as actions, events, and operators and programs are created by snapping together blocks, much like putting together a jigsaw puzzle. Figure 8 shows a Scratch interface where a block's shape indicates its purpose. The method of assembling blocks shows the flow of the program.

Scratch can be used to create animations, stories and games. It helps children to learn coding concepts and create interactive projects without needing to learn a text-based programming language. This means they will not be slowed down by their ability to remember complex code and syntax.

2.1.1.3 *Blockly*

Blockly is a client-side library created by Google in 2012 that adds a visual code editor to web and mobile apps. The Blockly editor (Figure 9) uses interconnecting, graphical blocks to describe programming concepts like variables, logical expressions, loops, and more. It enables users to implement programming principles without worrying about syntax or text editors [15]. For Blockly and its interfaces, the left side of the screen shows the code as the user link blocks together and can switch programming languages on the fly to see the differences in language syntax(right side of the screen) for the same basic program. This makes Blockly ideal for teaching code to a wide range of people, including older kids and adults who may not enjoy the younger-skewed sprite of Scratch [48]. Currently, Google and MIT are working together to develop the next generation of Scratch based on the Blockly platform, called Scratch Blocks.

[11] https://www.sclibrary.org/Home/Components/Calendar/Event/72570/3348?backlist

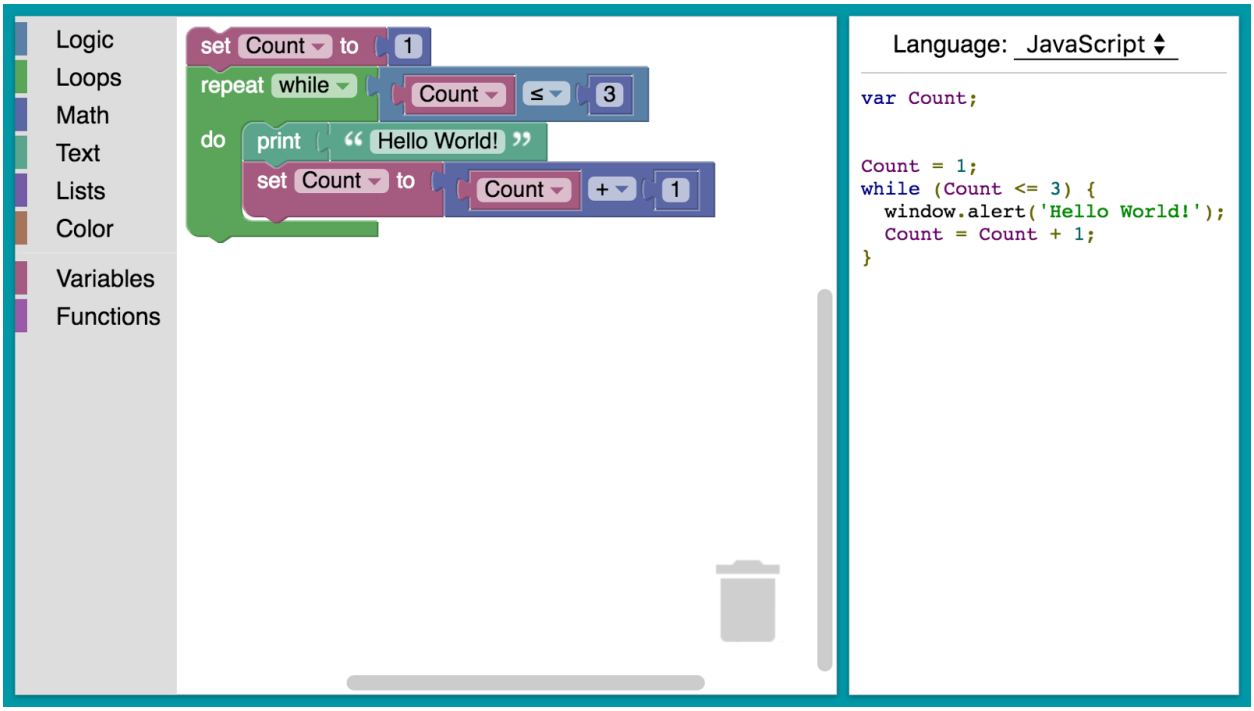


Figure 9: Blockly editor[11]

2.1.1.4 *Snap!*

Snap! is another example of a visual, drag-and-drop programming language. It is an extended re-implementation of Scratch that allows you to Build Your Own Blocks (BYOB) [87]. Snap! was inspired by Scratch, and targets not only kids but novice programmers (adults) and more advanced college students (non-Computer Science majors). The added capabilities like support for lists and procedures makes it suitable for introduction to more serious studies in Computer Science for college or high school students. With Snap! being a more extensive language, and its ease for defining new blocks using JavaScript — without changing the source code — can even assist in teaching and learning about concepts such as artificial intelligence.

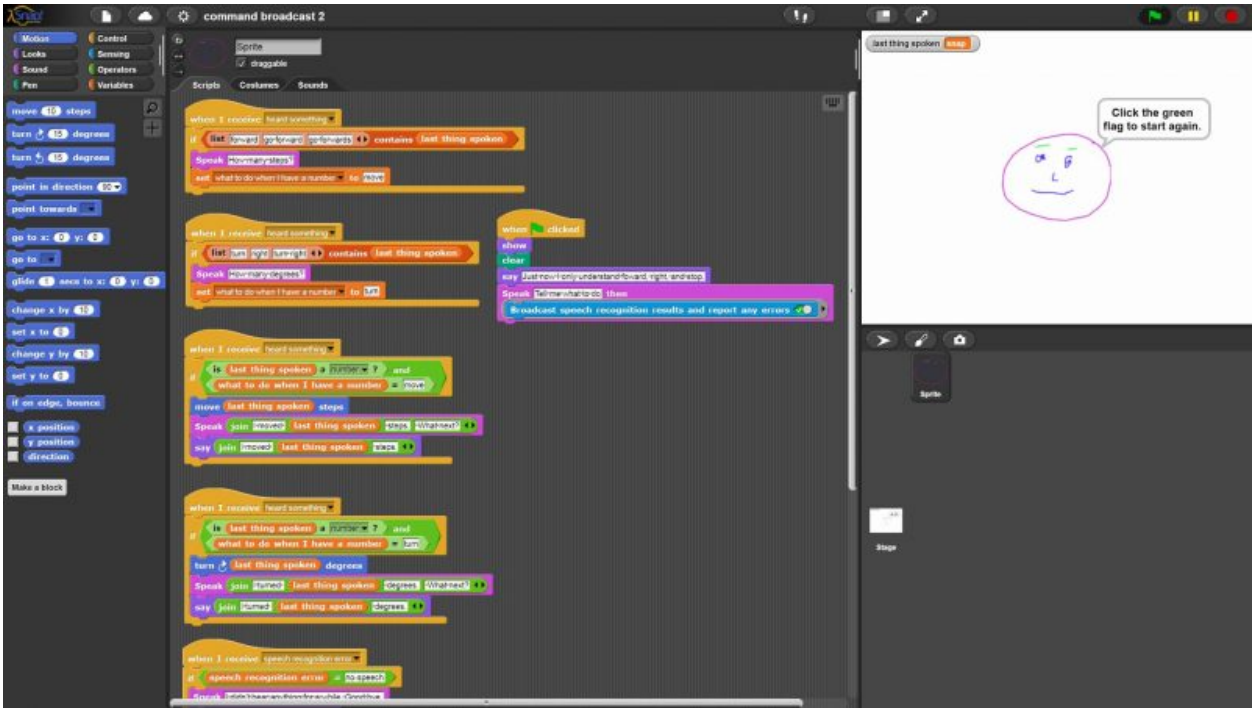


Figure 10: Example of custom blocks for machine learning in Snap![12]

2.1.1.5 *App Inventor*

App Inventor is an intuitive block-programming language that allows anyone to build apps for smartphones and tablets. The purpose of App Inventor is to encourage software development by empowering people to create their own applications for their Android phones and to move from technology consumption to technology development. MIT App Inventor allows students to shift their understanding of themselves from individuals who know coding to members of a community who are empowered to have a real impact in their daily lives and those of others. "For example- Students in the United States have developed apps to help a blind classmate navigate their school (Hello Navi); students in Moldova developed an app to help people in their country crowd source clean drinking water (Apa Pura); and as part of the

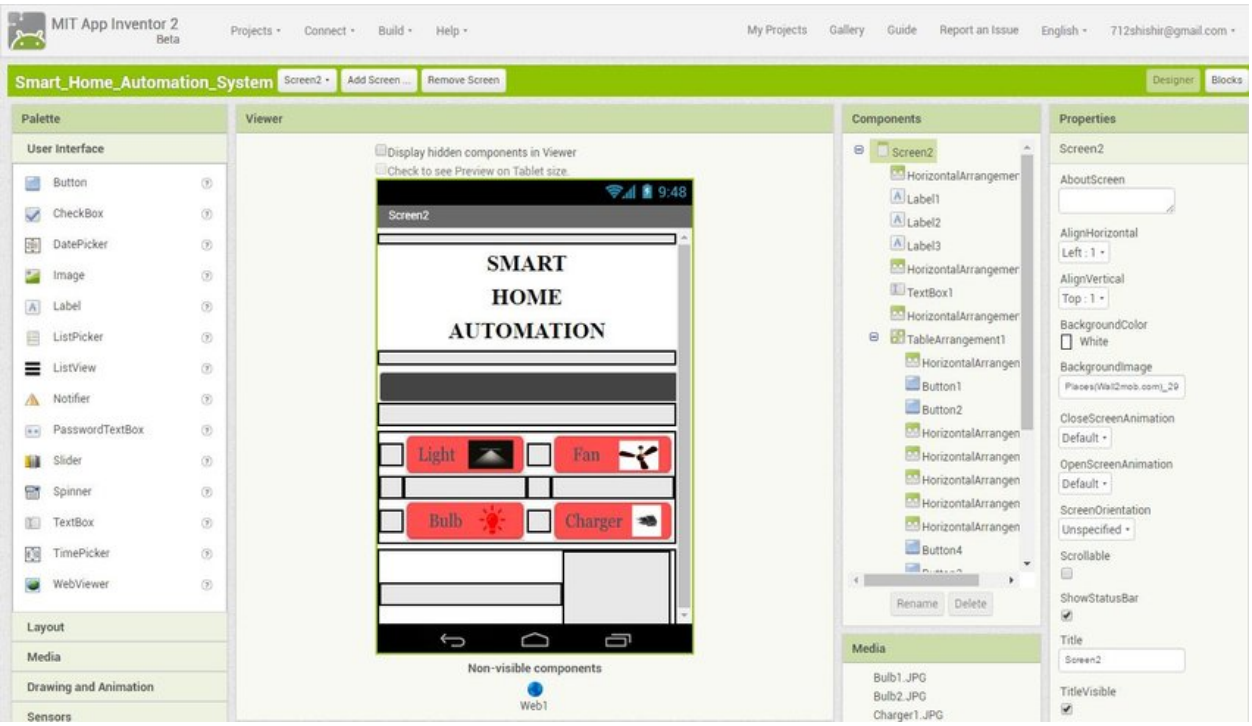


Figure 11: App Inventor example to build an android phone application[13]

CoolThink@JC project, students in Hong Kong created an app, "Elderly Guardian Alarm," to help the elderly when they got lost" [73].

2.1.1.6 *Alice*

Alice is a block-based programming environment that provides an interface to create animations with interactive narratives or program simple games in 3D. Unlike many of the puzzle-based programming applications, Alice motivates learning through creative exploration [5]. Research has shown that using Alice has positively impacted students' performance and attitude towards computer programming and learning OOP concepts [4]. Alice is widely used in schools and colleges, and even supports a full transition to the Java programming language

[12]https://s4scoding.com/adding-machine-learning-blocks-to-snap/snap-machine-learning-custom-blocks/
[13]https://www.researchgate.net/figure/MIT-App-Inventor-2-Component-Designer-GUI-Button-Layout-Design_fig3_321502885

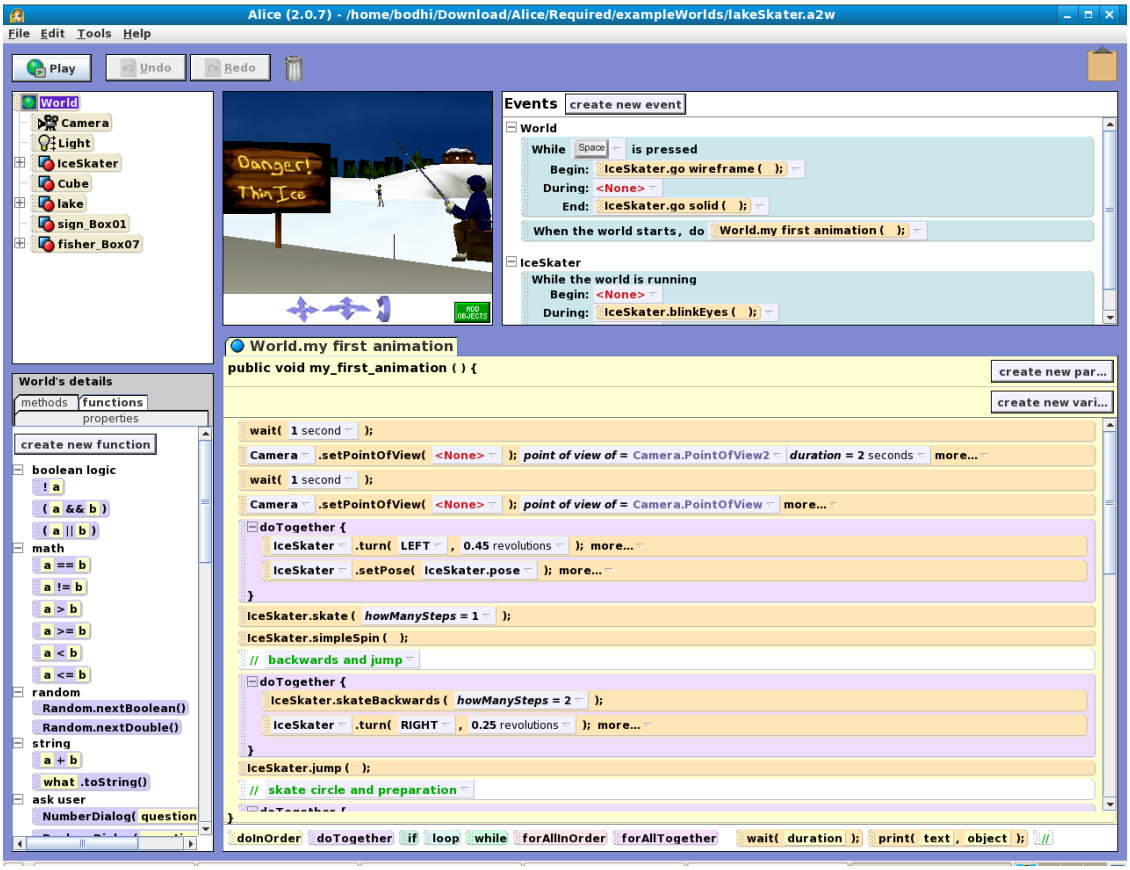


Figure 12: Alice programming to build interactive games and to learn Java[14]

by viewing the generated Java code in a side by side window (Figure 12). It also allows exporting the code into other IDEs (Integrated Development Environments), such as the NetBeans IDE to be able to extend the functionality by coding Alice code directly in Java.

2.1.1.7 *LEGO Mindstorms*

Lego Mindstorms is a software platform produced by Lego for the development of programmable robots. It is another visual programming environment to learn to program robotics that can be run on tablets and computers. Various schools use LEGO EV3 kit, and students learn to combine actions, variables, and events to manipulate their LEGO

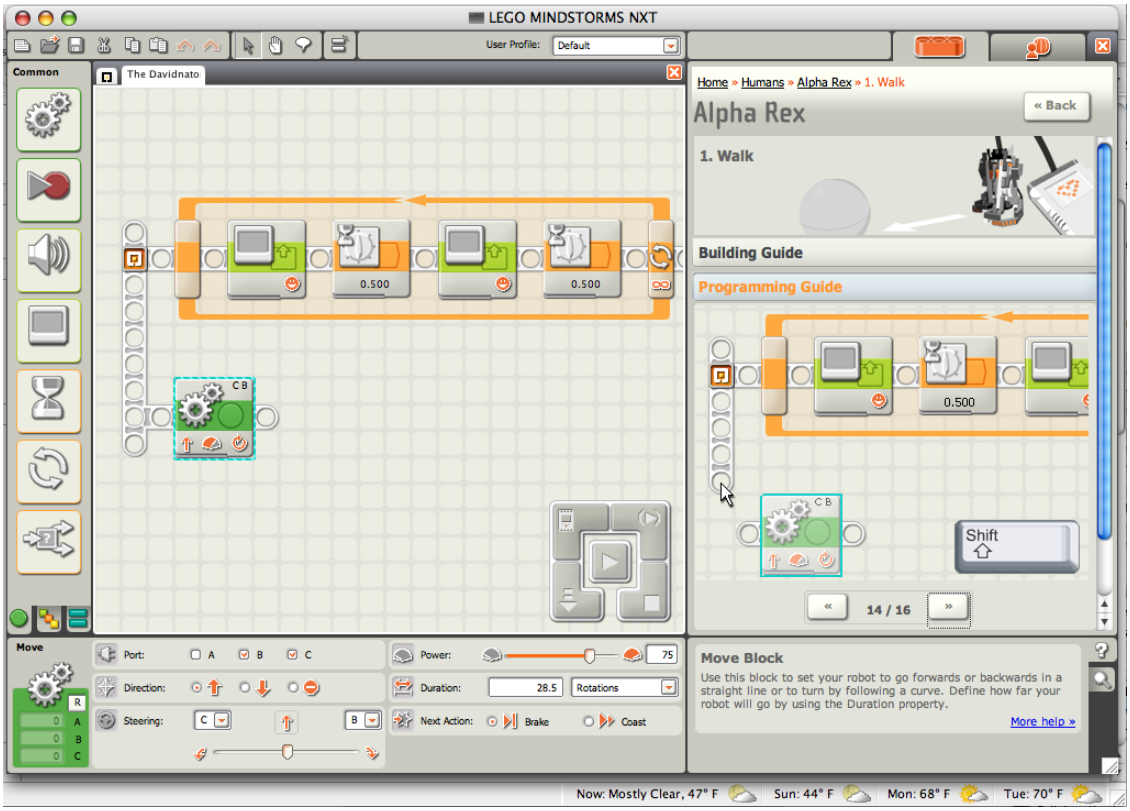


Figure 13: Lego Mindstorms editor to program robotics[15]

Mindstorms creations (Figure 13). The programming language is simple and easy for young kids and, on the other side, challenging and exciting for older ones and adults [48].

2.1.2 *Why Block-Programming?*

Technology is rapidly transforming the world, and software development is in high demand, making programming become an essential skill in the 21st century. Block-based programming allows for the creation of code in a way that is both visual (simple) and similar to

[14]https://www.researchgate.net/figure/Alice-Programming-Environment_fig1_228525906

traditional text-based coding (robust). Block-programming became predominant in introducing programming to kids and is used all over the world now [14].

Due to its widespread success in the field of education, it is often mistaken that block programming is just for kids. This idea is unnecessarily limited because block programming has already demonstrated potential in other domains [14].

Generally, programming languages are designed by programmers, for programmers [14]. This makes programming hard and time consuming for end-users. It is inaccessible for end-users to make any minor changes to the application or build a simple app for personal use without consulting developers. Visual and block-based programming have emerged as a solution to some of these problems, and have made programming accessible (and in some cases, faster) for non-programmers. Some advantages of block programming are:

Syntax-free programming: Block-based programming reduces the burden of using complex syntax and lets a user focus on implementing business logic in a fast and clear manner [11]. For example, in text-based programming like Java, missing a semicolon results in an error message and prevent the program from running. Furthermore, usage of block-based programming by professional software developers cuts the development costs and development time for small applications, as it is easier to maintain than traditional programming projects and benefits both the development team and customers [66].

Reduced Cognitive load: Traditional text-based code is hard to use because it presents a high cognitive load for novices. Using blocks reduces the cognitive load by chunking code into a smaller number of meaningful elements [11]. Blocks also help reduce cognitive load by teaching new programmers how to read larger chunks. In the Code.org Computer Science Principles course, blocks for JavaScript for loops are drawn just as an expert would see the

[15]https://www.asee.org/documents/conferences/k12/2012/2012K12_How_to_Start_aLEGO_MINDSTORMS_Roboti

code: as two chunks with a single block with a single socket for the loop upper bound (Figure 14) [11].

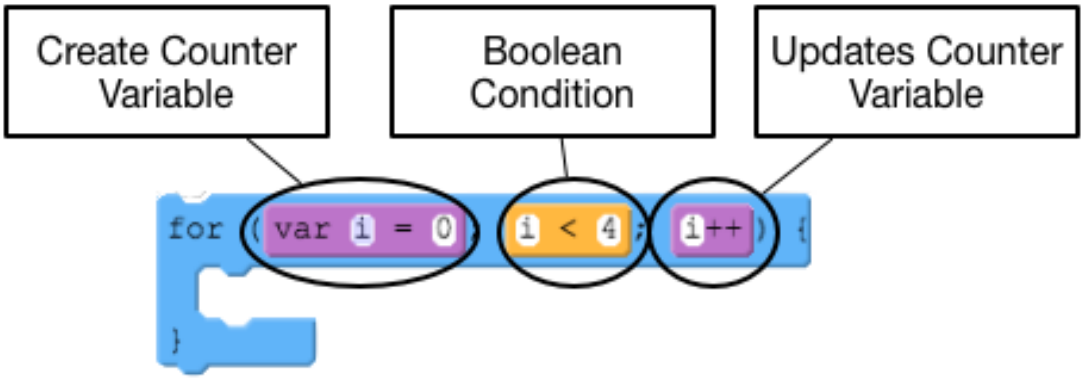


Figure 14: Example of For loop in Code.org[16]

Recognition vs Recall: Learning a new programming vocabulary is hard. Blocks simplify this problem because picking a block from a palette is more accessible than remembering a word: blocks rely on recognition instead of recall [14]. Block-based programming is a form of coding where the developer drag and drop blocks to issue instructions, these are lego-like blocks that are arranged together to create programs, and thus, developers do not have to memorize syntax to write code [14].

Easy to Understand: With visual and block-based coding, end-users or even novices, can understand how the program works at a glance. It makes the source code readable due to the use of colors and shapes. Block-based languages are presentable and understandable even to novices [14].

[16]https://studio.code.org/s/csp5-support/stage/5/puzzle/3

2.1.3 *How Block-Programming works*

Block-based programming languages utilize a primitive-as-puzzle-piece metaphor, and the pieces themselves provide visual cues to the user about how and where they can be used [97]. Programming in BBPEs involves a drag-and-drop interaction style with blocks to mitigate the low proficiency of typing skills (which is common amongst young learners). These blocks can be snapped (or joined) together to make valid syntactic statements that are executable scripts (or programs) with different functions. Block-based programming languages and environments help students overcome common barriers that novice programmers encounter such as selection, coordination, and usage barriers [54, 6].

As mentioned earlier, Scratch is one example of block-based programming. Scratch features sprites (programmable objects or characters). Figure 15 shows an example of how to draw a square on a screen, with sides of length 100. The script causes the sprite to move around the screen that draws a shape or pattern as directed. The sprite points in a direction, and moves 100 steps and then turns 90 degrees. It repeats this 4 times which completes the square of length 100. The way in which blocks are assembled represents the flow of the program. These blocks, when put together, forms a sequence of instructions. The shape of the blocks indicates its purpose. The "Pen Down" block will make sprite continuously pen a trail wherever it moves. The "repeat" block is "C" shaped, enclosing the instructions to be repeated. The "move" and "turn" blocks require inputs, that can be entered by clicking on the number field. If two blocks cannot be joined — often because they do not form a valid syntactic statement — the environment prevents them from snapping together. In addition

Figure 15: Scratch code to draw a square[17]

to the shape of blocks, other visual clues such as color and the nesting of blocks to denote scope are used to help novices [58].

Due to its ease of use for young learners, block-based programming has become widespread and adopted in the computer science education and STEAM learning communities, with

[17]https://www.researchgate.net/figure/Scratc-program-right-to-draw-a-square-with-side-of-length-100-The-equivalent-Logo_fig1_306234201

several tools being used today, including such as Scratch [80], Blockly [35], MakeCode [62], among others created in the research literature [19]. Studies have shown that these tools help users better understand concepts such as nested statements and return values of functions [77, 99].

2.2 NOVICES AND PROGRAMMING

Novices face difficulties when learning computer programming. Winslow summarised the problems of novice programmers as: 1) lack of adequate mental model; 2) fragile knowledge (surface knowledge, rather than strategies); 3) use of general problem-solving strategies (i.e. work from goal to solution) rather than strategies on a particular problem; and 4) use of control structures, line-by-line and bottom-up approaches in programming [101]. Studies have indicated that altering the learning condition of novices is promising for improving their performance [74].

The areas of learning difficulty are also widely recognized in the community of Computer Science education. [17] describes these areas of difficulty for novices in programming as: 1) problem orientation, i.e. understanding the problem to solve; 2) notional machine, a conceptual model of a computer system by which novices understand the execution of the program; 3) understanding the notation of various formal languages including both syntax and semantics; 4) programming patterns; and 5) the pragmatic skills of programming (including the life-cycle, testing, debugging, and tool use).

Novices can be classified into two groups: (1) students from mid or high- school or (2) non-professional end-user programmers. Block-based programming is often used as an educational tool for introductory Computer Science classes in schools, but universities and colleges such as MIT often conduct training for teachers to apply block-based programming

environments to work with different groups of students [34]. Looking closer at the success of block-programming for high-school and college students, [38] suggests that end-user programming can be one of the promising areas for blocks-based tools. As most end users have little or no formal training in programming, the learnability of blocks-based programming is likely to benefit this end-user as much as it benefits learners in education. [60] states that blocks-based programming environments are more distinct in several ways and the reasoning behind its creation, its target audience, and how the tool engages its users in computational thinking and learning to program, are valid reasons for their application in other domains. Some examples of domains where end-users use block-based approaches are as follows:

App development: MIT App Inventor [44] is used to enable relative novices to create apps that can run on Android devices. The intended audience is made up of newcomers who want to express themselves to the field of mobile computing through the creation of applications.

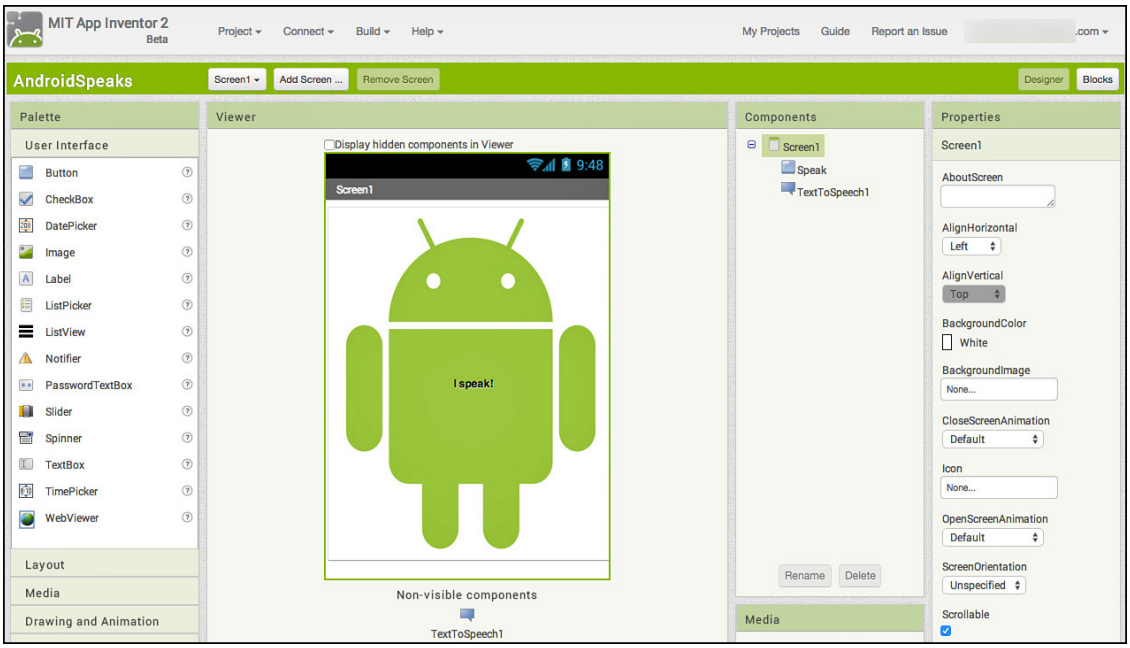


Figure 16: MIT App Inventor to build android applications[18]

[18]https://www.informit.com/articles/article.aspx?p=2270331

The creators of App Inventor tried to engage these novices with programming, something they might otherwise not choose to pursue due to several issues (e.g. difficulty) [60]. With App Inventor, users can build a mobile app without having expertise in programming, including apps that can directly improve their everyday life and those of their family and friends [45].

Figure 17: StarLogo TNG for modelling and simulation[19]

Simulation: StarLogo TNG [92] is a client-based modelling and simulation software. It facilitates the process of building and understanding simulations of complex systems [92]. StarLogo (Figure 17) is an ideal tool for beginners to explore complex systems and build rich games and simulation models using a 3D graphical interface with colored blocks that fit together and represent the small chunk of programming language.

Gaming: Gameblox [33] is a blocks-based programming tool explicitly designed to provide

[19]https://education.mit.edu/project/starlogo-tng/

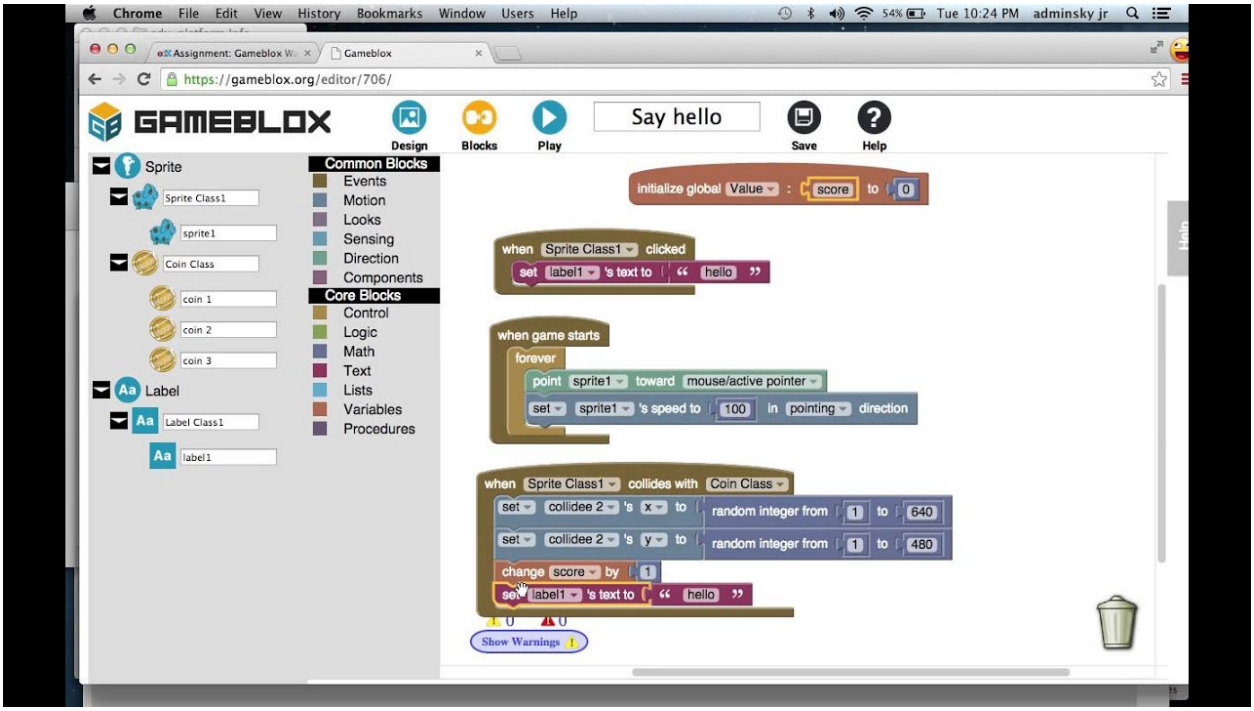


Figure 18: GameBlox to create games[20]

an easy on-ramp to game design. It is being developed at the MIT STEP Lab and allows anyone to make games. Gameblox provides an environment in which learners can effortlessly build entertaining games by implementing some of the complex programming tasks such as collisions, physics and gravity within the blocks [65]. It allows anyone to make games online that one can play both on a website and mobile devices. Advanced game designers might also find Gameblox a beneficial tool for rapid prototyping in the iterative design process [65].

IoT device development:

Node-RED [69] is a visual flow-based programming environment for binding together hardware devices, APIs and online services as an element of the IoT [70]. Node-RED is built on Node.js, utilizing full advantage of its event-driven, non-blocking model. Node-RED provides

[20]https://2rproduction.wordpress.com/2016/01/16/introduction-about-gameblox/
[21]https://www.instructables.com/id/Physical-Computing-Scratch-for-Arduino/

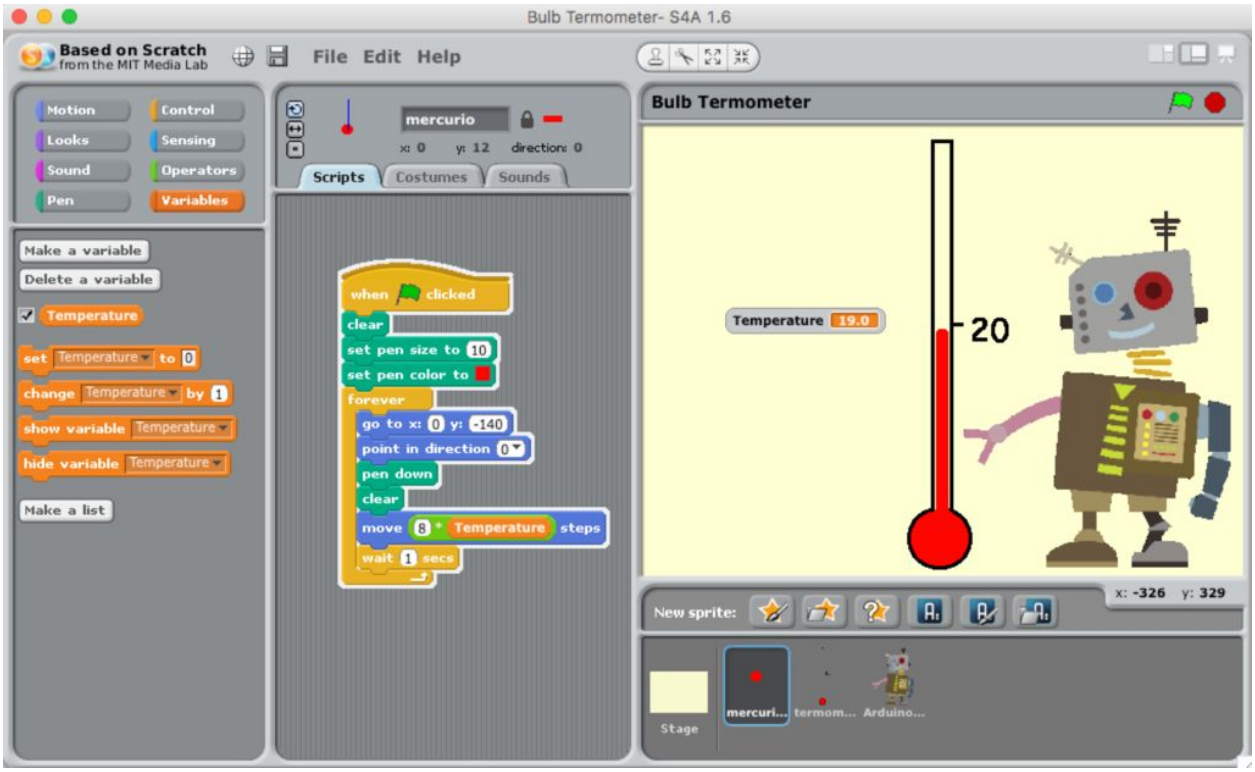


Figure 19: Scratch for Android to program Arduino boards for IoT products[21]

a browser-based flow editor making it easy to wire together flows using the wide range of nodes in the palette. Hardware devices such as Raspberry Pi[22] and Arduino[23] can be nodes [9]. Scratch for android (S4A)[24] is another tool, made for integration and experimentation with Arduino based IoT products. The s4a has a designated protocol stack for communication with Arduino boards. It also offers support for Android users to get connected with Arduino through HTTP using "Scratch based remote sensor protocol under the GNU General Public License v2 (GPL2)" [76].

[22]https://www.raspberrypi.org/
[23]https://www.arduino.cc/
[24]http://s4a.cat/

2.3 MOBILE LEARNING

Mobile learning is a method for learning on the move. It is about learning across various contexts, through social and content interactions. Narrowing the classroom content to make it available on mobile devices is not the complete utilization of mobile technology for learning. There is a need to design mobile learning applications creatively that connects people in real and virtual worlds, creating learning opportunities for people on the move, providing expertise on-demand, and supporting learning for a lifetime [47].

"Mobile learning values and defends in its own unique way the initiation of what is fundamentally new in the technological, social and cultural spheres of human life and activity." Mobile learning offers creativity and the opportunity to a radically new paradigm that encourages learners to abandon the limitations of the conventional ways of thinking, learning, communicating and designing [30].

Research indicates that these significant changes in mobile learning pose new difficulties for designers like - "What new design paradigms and applications can be associated with the use of mobile technology? How can we utilize their full significance within the context of traditional instructional design theory? Before the advancement of new forms of information and computer technology such as the current smart cellular telephones, the design paradigms that delivered the content of higher education remained mostly static" [30]. The exceptional potential integrated into mobile devices anticipate progressive changes in the educational domain. The simple and easy learning through the use of mobile devices makes it an even more potent tool of educational communication than the conventional forms and modes of traditional education [30].

Figure 20: Schools considering allowing students to bring their own devices to school for student engagement with mobile learning devices such as tablets and smartphones[25]

Augmented reality (AR) is a growing feature on mobile devices, reflected by the rise in mobile computing in recent years and the everyday ubiquity of Internet access across the world [31].

Most of the existing AR/MR educational applications focus on narrowing the classroom content for a topic, for example, mathematics, physics or other areas. Developers usually define this content, and there is no simple way for teachers to update existing or add new content. On the other side, some applications are easy to manage but are general-purpose AR displays, i.e., they are not education-oriented [25].

[25]https://www.theepochtimes.com/tech-pitches-ny-educators-to-embrace-smartphones-in-classrooms_222792.html

The mobile devices, in combination with the AR/MR technology, can be an advancement in technology-based learning, where learners explore and utilize the emergent AR/MR technology [25].

"The potential for Mobile AR applications in education supports situated learning, whereby a learner interacts and plays around with realistic objects with less risk than that associated with reality." [31] indicates that the key benefit of an AR environment is a 'sense of presence' that virtual or simulated environments typically lack it. "This technology, in combination with mobile devices and wireless connectivity, could enable learning anywhere and at any time" [25].

2.4 MOBILE LEARNING WITH AUGMENTED REALITY

Augmented Reality (AR) offers a new form of interactivity between physical and virtual worlds and enhances a user's perceptions of the real world [51]. In educational contexts, it has the potential to maintain a high level of motivation among children, and a positive impact on a student's learning experience [43]. AR technology used to display augmented content as a layer superimposed on the real environment can be divided into three categories, (1) mobile devices, such as Smartphone and tablet, (2) stationary units and (3) head-mounted display [68]. Head-mounted displays are rarely used in education as they are still expensive and immature. In contrast, students nowadays bring smartphones to school and are familiar with these devices.

One example of a successful Mobile AR application in education is from [43], where they created an AR application to teach the basic concepts of electromagnetism. Students could explore the effects of a magnetic field. Using a mobile device and physical objects (cables, magnets, battery, etc.) which were recognized using the camera of the mobile device (like

a tablet), students were able to see superimposed information such as the electromagnetic forces or the circuit behaviour using the tablet. This work highlighted that Mobile AR improves academic achievement and that instant feedback for objects was valuable for students.

More recent research has summarized the benefits of Mobile AR in education [103] as:

1. Engage, stimulate, and motivate students to explore lessons and concepts from different angles
2. Enhance learning where students could not feasibly gain real-world first-hand experience
3. Enhance collaboration between students and teachers
4. Foster student creativity and imagination
5. Aid students to control their learning at their own pace and on their own path
6. Create an authentic learning environment suitable for various learning styles

One demonstrated example of these benefits was shown when AR was introduced into engineering design [40], where high school students manipulated designs in AR. The effectiveness of AR in this context revealed a significant impact on STEAM education and its influence on the career trajectories of students.

Various learning styles enhanced by immersion in distributed-learning communities based on AR interfaces described by [27, 29], are as follows:

(a) Fluency in multiple media

(b) Learning based on collectively seeking, sieving, and synthesizing experiences, rather than individually locating and absorbing information from some single best source;

(c) Active learning based on experience (real and simulated) that includes frequent opportunities for reflection

(d) Expression through non-linear, associational webs of representations rather than linear "stories" (e.g., authoring a simulation and a web-page to express understanding, rather than a paper);

(e) Co-design of learning experiences personalized to individual needs and preferences

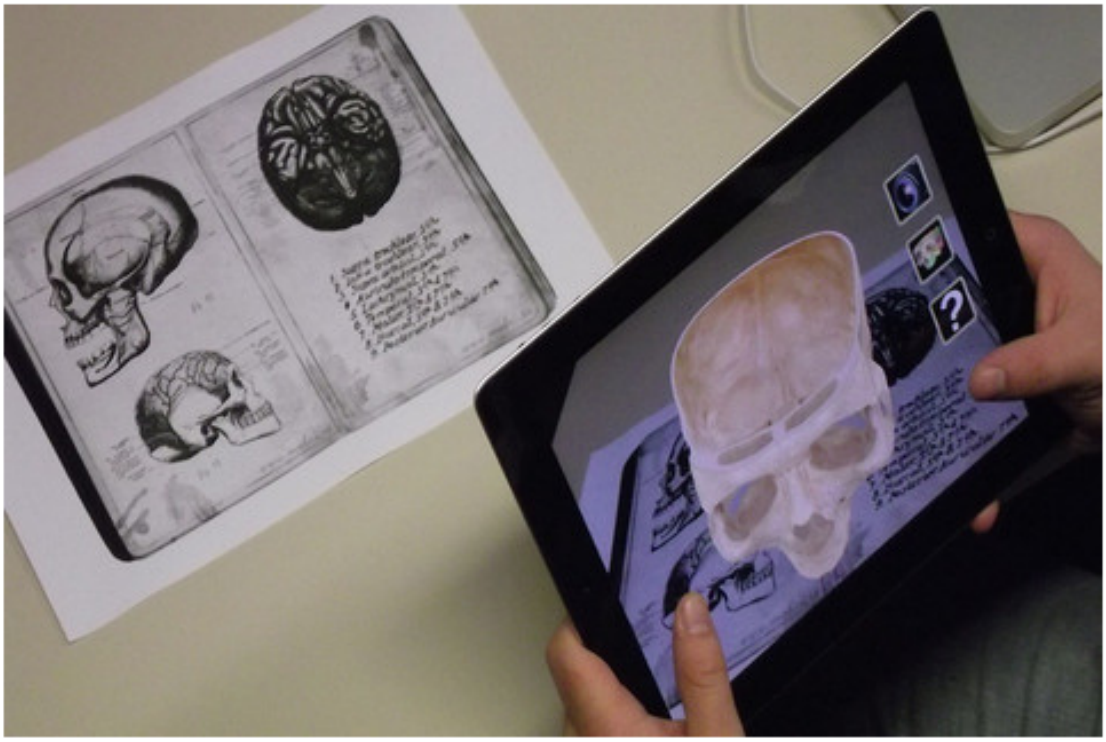


Figure 21: Learning Science with mobile AR[26]

Ultimately, research has shown that AR/MR has been mostly applied in educational settings to motivate students to learn different educational concepts [8]. Exploring uses beyond

[26]https://www.smartivity.com/pages/augmented-reality-pioneers

motivation, as well as early age learners, provides an interesting opportunity for further exploration in AR/MR.

2.5 BLOCK-BASED PROGRAMMING WITH MOBILE AR/MR

The integration of immersive environments — such as virtual reality (VR) and AR – with visual programming is a relatively new research area. Recently, [4] explored the impact of using a 3D visual programming tool on student's performance and attitude. The findings of their study indicated that their 3D programming tool was effective in improving student performance and outlook towards computer programming. Similarly, [26] demonstrated in their work that the benefits of AR/MR aspects like an interactive interface could extend to help students learn to code more efficiently and with more fun. Furthermore, [71] shows that 3D Block-based programming may provide an option for introductory classes, as well as the demonstration of 3D representation of possible objects. The study shows that these systems would make learning more enjoyable and interactive.

Figure 22: Cubely environment to learn block coding in VR[27]

Figure 22 shows an immersive VR programming environment for novice programmers to solve programming puzzles within a virtual world [93]. Their observational studies concluded that the students found it easier to work with the VR interface and preferred it over the traditional 2D block-based programming. Similarly, [52] presented their investigation into leveraging mixed reality (MR) to help students learn to code more efficiently and with more fun. Their research shows significant benefits with augmented reality (AR) to learn to code: (1) allowing teachers to tailor their instructional needs, and spark creativity and engagement among students in coming with program problems that interest them; (2) enabling users to

[27]https://www.semanticscholar.org/paper/Cubely

physically interact with a program, concretizing coding errors and providing real-time visual feedback that could aid user's understanding of the program and lessen the cognitive load.

In a block programming environment, it can be challenging to find and navigate to the relevant part of a blocks program in a 2D workspace, only part of which may be visible [11]. Including a 3D web view of a block-based programming language in Mobile AR/MR provides more screen space and flexibility to adjust the screen size as per users' convenience, which makes searching and navigation processes easier. Furthermore, with mobile AR/MR, users can experience intuitive touch interactions. [26] evaluated two interactive AR programming environments: (1) head-mounted AR with Microsoft HoloLens, (2) mobile AR with ARKit on an iPhone; together with a conventional 2D touch interface using Swift Playground on an iPad as the baseline. The study has shown that participants enjoyed using mobile AR the most, and they also completed programming tasks the fastest when using it.

As block-based programming tools like Gameblox and other block languages move toward the future, it is found that harnessing the capabilities of powerful environments, including virtual reality, augmented reality, mixed reality and server clusters running machine learning algorithms, into simple blocks provides a way for non-experts to interact with previously inaccessible technology creatively [50].

Overall, little research has explored the specific combination of AR/MR and block-based programming, both in education and the broader space of block-based programming tools. In this thesis, I explore this gap in the research space and propose an immersive environment that not only offers theoretical knowledge but also provides an interface for the practical implementation of the program that involves interactivity with hardware (cardboard, cables, Microbit etc.).

3

DESIGNING BLOCK-BASED PROGRAMMING TOOLS IN MR

3.1 STUDY I - OBSERVATION STUDY

To motivate and ground our work, we first conducted an initial observation study (see Appendix A for details) with a local Science, Technology, Engineering and Mathematics education company (STEM), STEM Learning Lab Inc., located in Calgary, Alberta, Canada [91]. Using their on-site maker space with equipment such as 3D printers, soldering equipment, and other tools, they offer hands-on workshops and camps for youth of all ages. Their topics include robotics, wearables, gaming, Minecraft and electronics. For many of their workshops and camps, they use block-based programming languages (i.e. Scratch [80], MakeCode [62]) to teach computing concepts and introductory programming. Our goal for this initial study was to observe current practices, attitudes and challenges around block-based programming, how it is used in classroom scenarios by both students and teachers, as well as get ideas on potential directions for block-based programming and MR.

3.1.1 *Protocol*

As STEM Learning Lab Inc. frequently operates 5-day long workshops, we participated as observers in a workshop focused on using block-based programming (specifically MakeCode [62]) and electronics (specifically the BBC Microbit [61]). The workshop had 25 students in Grade 3 and was facilitated by 5 teachers, each with at least 3 years of experience teaching computing and electronics using block-based interfaces. Each day of the workshop was broken down as follows:

Day 1: Introduction to programming concepts (e.g. loops, variables) using Scratch.

Day 2: Continued introduction to programming with Scratch, and introduction to programming electronics using Microbit and Microsoft MakeCode.

Day 3: Introduction to maker projects at the MakerSpace (e.g. 3D printing a button).

Day 4: Maker projects using the Microbit, where students were allowed to freely create.

Day 5: Additional construction for maker projects chosen by students, followed by final presentations.

For our initial study, our participation as observers was limited to Day 2 of the workshop. It was a 6 hours long class where in the first half of the class, scratch programming was continued and Microsoft Makecode was introduced to the students, and a physical Microbit and deployment of program on the Microbit was introduced in the second half of the program. The students and teachers were observed by roaming around the class-room and taking notes and pictures for further analysis.

3.1.2 *Results*

Overall, we observed both students and teachers having positive experiences with block-based programming using both the Scratch and MakeCode interfaces. From our observations, we synthesized the following themes from video analysis.

Blocks can (still) be puzzling: Occasionally, students faced some confusion as to what types of blocks fit with other blocks or even where to place them on the interface. This typically occurred after students had progressed through different concepts and more blocks began being introduced. This would also occur when more complex challenges arose in the workshop (e.g. shake and compass events on the Microbit). To solve this, teachers would either interrupt the class and explain the concept to all students through an impromptu presentation at their dedicated presentation desktop (Figure 23a), or they would explain it to students individually (Figure 23b).

Hardware deployment is difficult: A key challenge students faced, especially when the Microbit was introduced, was how to properly deploy their code from a computer to the Microbit itself. Students would often click download, to download their code onto the Microbit, but forget the steps to deploy it (i.e. copy the file onto the Microbit drive). Students often tried resetting the device (Figure 23d) or pausing their task and waiting for a teacher, slowing down their task.

Limited Resources: Due to the limited availability of resources (Microbits and laptops), students had limited access to resources and had to share Microbit and laptop with another student (Figure 23c). This led to an issue of co-ordination among students, where one of the students was quick to grasp concepts (e.g. nested conditions), whereas the other student faced issues in learning at the same pace. This limited the learning experiences for both students, and even made one student seemingly demotivated.

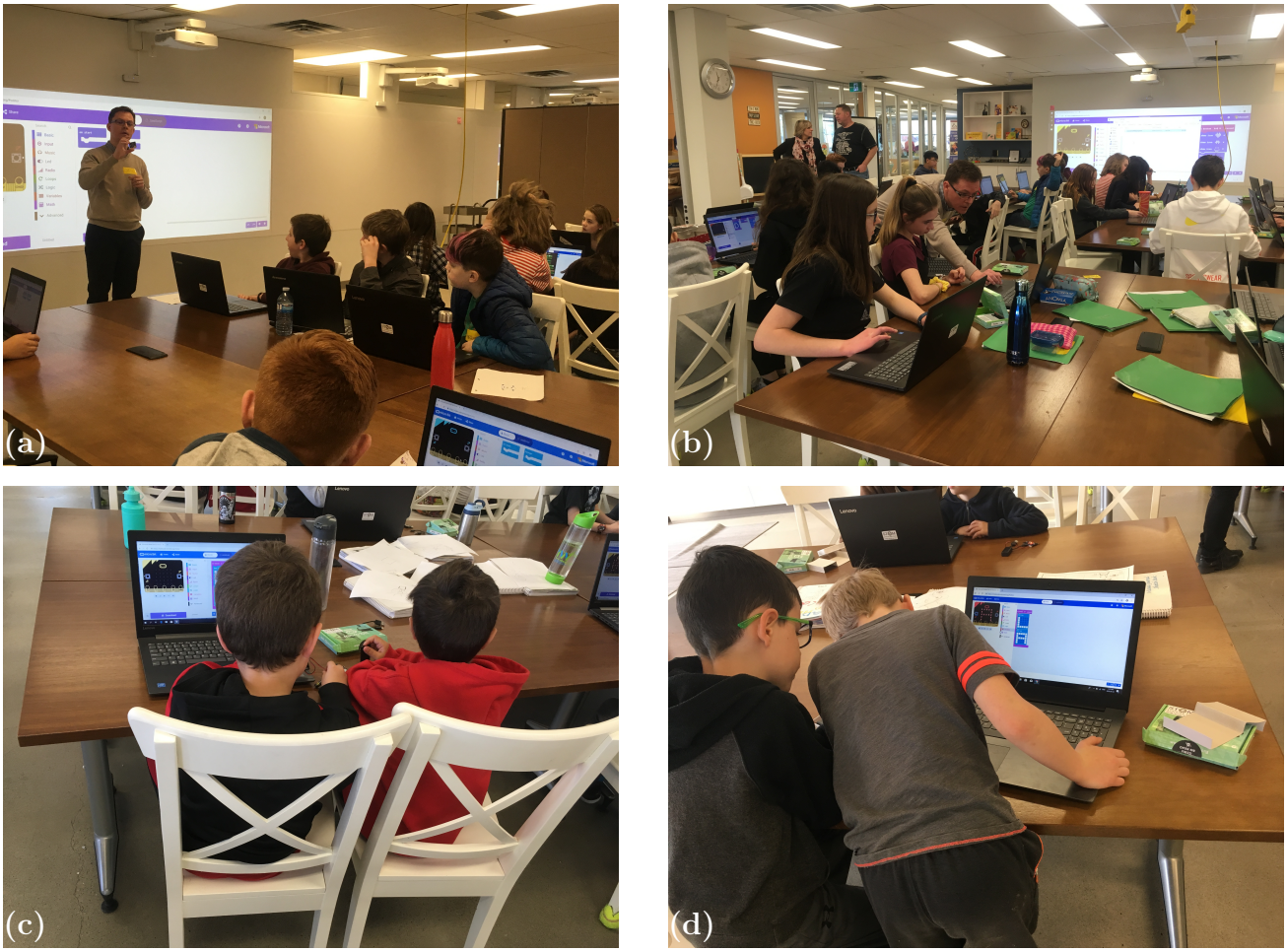


Figure 23: Observations from Study I: (a) A teacher presenting how blocks work together with a Microbit; (b) A teacher assisting students who are confused about why certain blocks don't fit together; (c) Students sharing laptop and Microbit; (d) Students stuck with deploying their block-based code onto a Microbit and trying to debug it themselves.

Confidence in Learning: Both block-based programming environments assisted in the activities with the Microbit for the students and teachers. We observed teachers frequently demonstrating programming-specific concepts (e.g. nested loops) and students grasping them relatively quickly. As students progressed through the workshop activities throughout the day, their confidence in their ability to accomplish tasks arose, as did the teachers. As a result, the teachers occasionally felt confident enough with the students and the Scratch or MakeCode interfaces to spontaneously teach more advanced topics.

3.1.3 *Discussion*

From our initial observations, we confirmed that several areas can be addressed by providing an alternative or supplemental experience, especially one using MR. Deployment with hardware were clear challenges for both students and teachers, beyond what was described in prior research, and simplifying this experience has great potential to improve the block-based programming experience with hardware and electronic devices. Although hardware was not our initial focus and the use of Microbit hardware is specific to the workshops offered by STEM Learning Labs, we witnessed a clear need from both the student and teacher perspective.

How blocks are presented should also be modified or even simplified further, especially with younger age groups. We observed that because students had different levels of skill in programming and dealing with electronics (despite being in Grade 3), it led to the spontaneous introduction of more advanced topics. This introduced a dynamic where not all students were able to follow-along and they would proceed in becoming stuck or frustrated in their tasks. Part of this challenge was due to the teacher being stuck to one method of presenting topics — their desktop at the front of the class (Figure 23a). While this might

be specific to the layout and equipment available for STEM Learning Lab, it does represent an interesting scenario for MR — to provide an environment where all students can gain practical knowledge with virtual objects.

3.2 STUDY II - SEMI-STRUCTURED INTERVIEWS

Following our observations and initial analysis from the workshop, we then conducted interviews (Appendix A) with local teachers to get more detailed, instructor-focused feedback on current practices and challenges with block-based programming. While our initial study was critical in grounding our work, through our interviews, we aimed to develop design considerations for a block-based programming system that uses MR to address current challenges, especially those we observed around electronics (specifically the Microbit).

3.2.1 *Protocol*

Through STEM Learning Lab, we recruited 5 teachers (3 female, 2 male) from various local schools in Calgary, Alberta, Canada, all of whom had experience in teaching block-based programming and maker activities that involved some form of electronics (e.g. Arduino, Microbit). Their teaching experience ranged from 2 years to 25 years total. Three of five teachers focus on teaching programming languages such as python, Java-script using block-programming to junior students, and teachers 2 and 3 focus on teaching Maths and Science subjects, including block-programming.

We used a semi-structured interview process, where we asked teachers several questions related to the methods they used for teaching block-based programming with electronics

(See appendix B for questionnaire). Each interview was 30 minutes long and the questions spanned topics such as challenges they currently face, opportunities for improvement in teacher and student experiences, as well as if and how their practices have changed over time.

3.2.2 *Results*

Following the interviews, we used a thematic analysis from our data using grounded theory [20] to generate key findings before implementing a prototype (similar to [18]). These findings confirmed our initial observations and provided additional insights from the perspective of teachers who would be facilitating STEM Learning Lab workshops using a (future) tool that combines block-based programming and MR for Microbit-based (and other electronics) workshops.

Simplicity vs. Accessibility: While several teachers noted that block-based programming was simple and "*...makes programming more accessible...*" (Teacher 4), it was also initially difficult for students to understand types of blocks, and where and how they fit together, and occasionally even how to begin some of the more complicated activities that use sensors like a magnetometer or an accelerometer. One teacher stated "*One thing that make it difficult is sometimes it feels like the amount of choices that the students have in blocks and what they are getting is overwhelming...and we say we are going to use this one green block over here and they are like there are so many and they get confused. So it is difficult to find a starting place*" (Teacher 4).

Touch vs. Mouse Interactions: The role of touch vs. mouse-based interactions is a significant factor in the usability of block-based programming interfaces and their perceived simplicity. All teachers noted, typically they either teach on desktops or laptops (which is also

how the STEM Learning Lab environment is arranged), but that many children (especially younger ones) were more familiar with touchscreens. One teacher even critically commented "*Most kids have never used a mouse and so explaining what left clicking and right clicking is difficult. Explaining to them that you need to click on this bottom corner or that corner of the track-pad to make it work, it takes them a day*" (Teacher 5).

Teachers also noted that as there was a lack of familiarity with mouse-based interfaces for students, it contributed to the students occasionally getting stuck, which limited their exploration, in many cases becoming "*I would like to do this and I am not sure how" type questions and then I usually just push them a little bit and ask 'have you thought of this or how would you do it in something else'...*" (Teacher 5).

Helping one vs. helping many: All teachers noted that while having a singular block-based programming interface was useful, it was not always ideal depending on the level of the students in their classroom, as not all students were equal in the level of their education, sometimes because of their socio-economic backgrounds. For example, a teacher stated "*other things besides reading like literacy or sometimes numeracy will constrain how far I can go with teaching coding.*" (Teacher 3).

Several teachers also preferred teaching blocks more as a group than one-on-one, as they "*prefer to help them together in the class, that is why I try to get them to contribute and I broadcast the screen and then they can tell me what they want to do and then we pull blocks and try to get rid of error and after that if they still have problem I help them one-on-one*" (Teacher 3).

Further, when it came to electronics alongside introductory programming tasks, several teachers made similar comments around teaching it in a group and more collaboratively, such as "*...rather than being on a screen right away we try to use some physical coding. Something that teaches them that sort of thought process before they get to the computer*" (Teacher 3).

Resource availability: In Study I, we observed all students were not having a dedicated machine while performing block-based programming tasks, several teachers commented that resources were continuously an issue (for both computers and electronics), and occasionally it magnified how they had to provide assistance and how children worked and learned together. For example, a teacher noted "*...it is the fact that they have to share computers, that is the biggest one, sharing a computer is a problem also they have different levels of understanding so some children understand better than the other*" (Teacher 3).

This issue was further magnified when resources didn't work (e.g. the internet) or when "*behaviour issues of students along with sharing resources impacted the lessons*" (Teacher 2).

Hardware is hard: Building upon the resourcing issue, the most common theme mentioned by teachers was that combining electronics devices with programming concepts was difficult, and learning one occasionally meant sacrificing concepts in another. For example, one teacher stated "*I was teaching a class this morning with Raspberry Pi and it was going a little too fast, kind of skipping some concepts that would have been important to know. Sometimes it's hard, especially for younger students to understand things like conditional statements or loops or something that leads the path of their program to get the results on the hardware.* " (Teacher 1).

This was also frequently mentioned when students were unable to "*...troubleshoot what's going wrong with their program*" (Teacher 1). Teachers also mentioned, whenever students did manage to successfully create a program, deployment became a significant challenge, and in several cases for teachers who use the Microbit, "*...there was an issue with how to download the code and deploy it to the device. The workaround we have been using lately is that WebUSB thing to kind of flash it to the device via USB cable*" (Teacher 1). Teachers also noted that concepts like file versioning, which is often necessary for the Microbit as it generates new code for each time 'download' is clicked, was "*tricky for some students*"

(Teacher 1). This is also linked to *Touch vs. Mouse Interactions*, as teachers noted that some students at early ages are not familiar with concepts around file-systems and other traditional desktop-based paradigms.

Completion vs. Understanding: The combination of block-based programming concepts alongside electronics often meant that teachers were required to balance time between ensuring their students were able to understand programming concepts and being able to complete a project. As one teacher noted "*Sometimes it is hard to maintain the balance between completing the project and making sure that they understand the whole concept like Why their code is affecting or causing a certain result*" (Teacher 1). As a side effect, some teachers believed that this in fact affected the interest in the tasks of some students. One teacher commented "*[it]...depends on the activity because there are student who are very interested in coding and very resilient in their problem solving and they just keep going*"(Teacher 5).

3.3 DESIGN CONSIDERATIONS

Based on the results of both our initial observation study and our follow-up interviews, as well as drawing upon relevant prior work previously discussed, we generated a set of initial design considerations for a system that combines block-based programming and MR. In addition, our design considerations were continually discussed and refined in collaboration with STEM Learning Lab Inc.

(D1) Improved physical task performance. One of the limitations we observed in the physical classroom environment for block-based programming was that the students found it challenging to understand the real-world value of the program and how to connect it with the real world objects. For example in a science project to monitor soil moisture,

it can be difficult for young students to understand how does the sensor on Microbit work to determine the moisture level of the soil, and how to connect the appropriate pins to get the code working. MR technology has the ability to bridge the gap between the virtual and real-world, allowing students to learn, experiment and perform thus helping them to acquire practical knowledge and experience [103].

(D2) Mixing physical and digital. As noted by several of the teachers, and from our own observations, the skill level of students ranged from completely inexperienced to very experienced. This often manifested in how students approached the Microbit. For example, in Study II, we observed some students having apprehension when beginning to use the Microbit despite being comfortable with the block-based programming aspect, while others preferred to use the Microbit and experiment with it physically (e.g. shaking it, click the buttons and seeing what happens) before beginning any programming.

To support this type of classroom environment for teachers and their students, combining physical and digital learning techniques could be a potential solution. Thus, we aimed to remove some of the aforementioned apprehension by enabling both types of interaction, where it is possible to manipulate both the physical Microbit and a virtual 3D model of the Microbit in MR in a similar manner. With the help of 3D models and a layer of digital information onto these 3D models in MR, it becomes easier to grasp the abstract and difficult content [103]. Additionally, providing digital resources can overcome some of the challenges teachers previously mentioned with physical resources in the classroom (e.g. not enough Microbits).

(D3) Limited presentation of blocks. Given the feedback that students occasionally felt overloaded with the number of blocks available to use when first starting many of the individual lessons in workshops or when teachers introduce a programming concept (e.g. how to use a temperature sensor), we endeavoured to present blocks in a more condensed format.

As this was already a challenge on a 2D computer screen [11], it also meant this was vital for how blocks are to be presented in the MR context. Allowing a teacher to dynamically select the blocks visible for students depending on their teaching material or topic, is one potential technique of condensing their presentation. Furthermore, adding a layer of digital information for the usage of blocks helps a student to understand the reason for using a block, and when and where to use it in a program. Integrating these solutions with MR technology, a student can learn at anytime and anywhere with their smartphones/tablets, resulting in increased motivation and a new way to understand abstract concepts [103].

(D4) Interactions inspired by touch. Both teachers and students are typically already familiar with touch-based devices, but in the case of younger children, their mental models of interactions with touch didn't necessarily map to the 2D mouse-based interfaces of block-based programming. This often led to difficulties and frustration, however, an obvious solution for this is to design a touch-based interface for devices like tablets or even smartphones. MR is accessible through these device and provides an exciting approach towards learning; which makes the lessons fun. As a result, it serves a positive impact on the students and keeps them engaged [43].

(D5) Reducing deployment friction. From both our initial observations and interviews, activities that involved loading code onto the Microbit was frequently a challenge, and we envision it in a similar experience with other electronics. Students were often unable to debug why it wasn't working, and teachers are sometimes unable to solve the technical challenges necessary to fix it. For example, the Microbit sometimes requires pairing over WebUSB in the browser, to load code directly onto it without the need for files and file versioning (which itself has challenges). If this isn't done correctly, then it becomes very difficult to load code onto the Microbit and it requires the pairing method with the browser to be restarted.

To reduce this friction, technologies like Bluetooth or Wi-Fi can be used. In building a prototype, we were concerned about how to support different types of electronics that are used in a classroom beyond what STEM Learning Labs supports in their workshops, but ultimately we settled on using Bluetooth as it was already available on the Microbit and other STEM-based electronics (e.g. Adafruit BlueFruit [2]).

3.4 THE PROTOTYPE

To support the scenarios and use cases described from our observations and interviews, we designed a mixed-reality block-based programming application that uses the Microbit, called *MakeMR*. We designed the application in collaboration with the STEM Learning Lab, who provided additional insights into how the teachers in their workshops would use the application. The aim here is to design a mobile application that not merely repackages the classroom material but utilizes the powerful features of mobile technology and provides practical and interactive learning i.e. practical implementation of the program by interacting with virtual 3D models.

The application was implemented for touch-based mobile devices (e.g. mobile phones, tablets) using a prototyping tool that supports both iOS-based version and the Android-based version. The block-based programming components of the application were implemented using MakeCode [62] as the framework. We chose to use mobile devices for a number of reasons: (1) we wanted to remove the dependency on a desktop and allow both teachers and students the freedom to move around in the classroom with a device that contained their code; (2) mobile devices typically have Bluetooth built in, which is useful for deployment (based on D5); (3) current mobile devices are powerful enough to run MR-based apps, and

a majority of students and teachers already have mobile devices, making it easier from a resource perspective (observed in Study I and discussed in Study II).

The prototype includes the following aspects of mobile MR technology:

Augmenting screen real estate: Including a 3D web view in mobile, MR provides unlimited space in MR and flexibility to adjust the screen size as per users' convenience, which may help in making the search and navigation processes easier.

Augmenting understanding: With the help of 3D models and a layer of digital information onto these models in MR, it becomes easier to grasp the abstract and difficult content.

Interaction with virtual objects: MR offers learners to interact and play around with virtual objects and validate their programs experimentally with less risk than that associated with reality. Research indicates that MR is "improving the success rate of physical interaction-related learning tasks and supports memory-related learning activities" [22].

For manipulating a virtual device, we provided a built-in 3D model of the Microbit that can be interacted in MR to understand its working and to explore features like sensors, pins, LEDs, buttons etc. (based on D2) (Figure 27). Similarly, block-based programming occurs using standard touch gestures available on Android and iOS (e.g. drag and drop, zoom, pinch, tapping, etc.). As mentioned earlier, several students were already familiar with touch-based interactions, so we chose a similar interaction model in MR that is not monotonous to kids and foster student creativity and imagination [103] (based on D4).

The interface follows a traditional workflow adopted for applications that utilize block-based programming languages, but in MR (on mobile devices). A user is first presented with a 3D interface where limited blocks that are relevant to the given learning activity are presented (Figure 24a). These blocks can be selected and modified by a user (in this case, a teacher) before the application is run. Throughout the interface, blocks are sized large

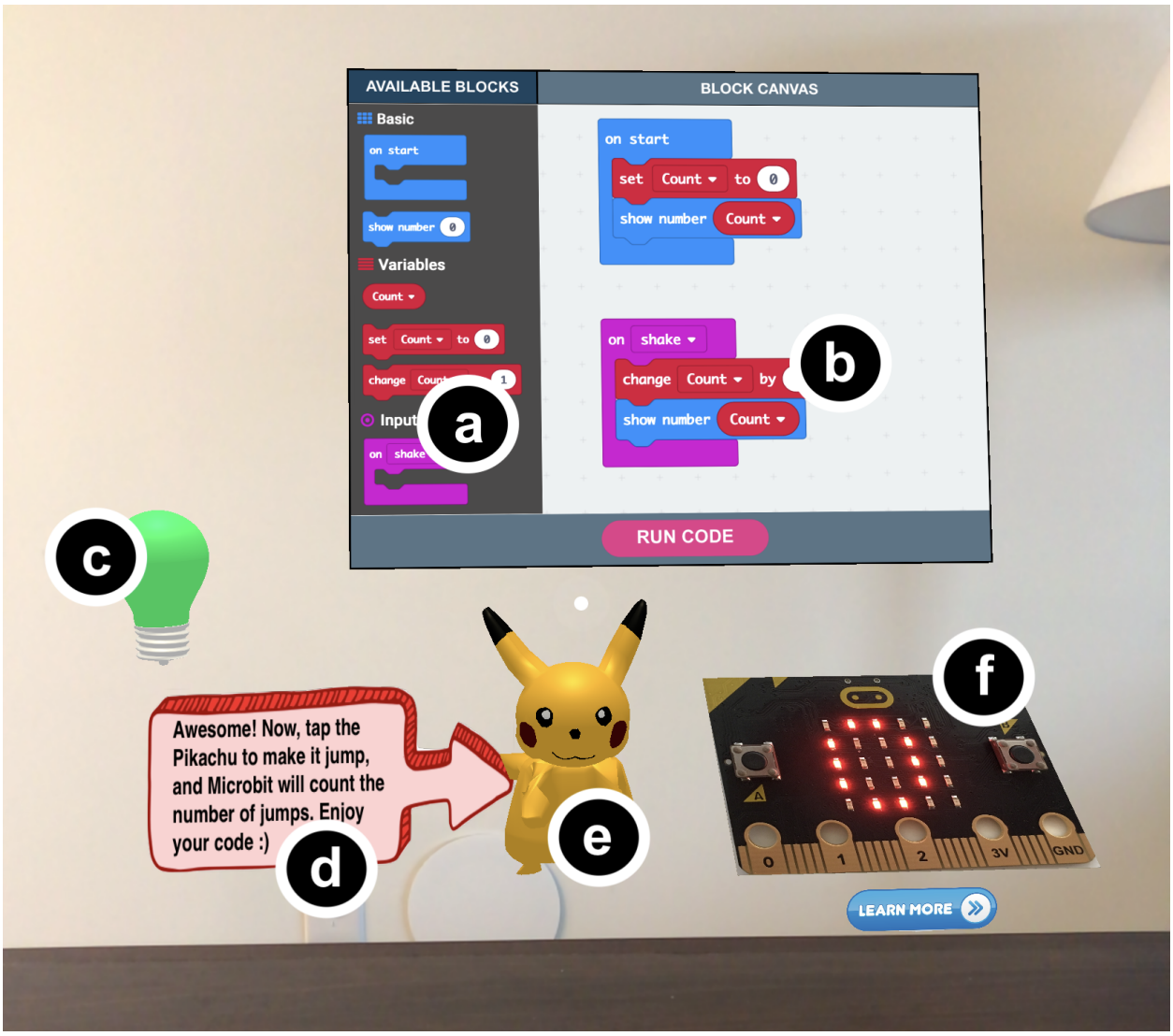


Figure 24: The different components of the *MakeMR* interface. (a) Blocks that can be dragged onto the canvas; (b) The canvas where blocks are placed for programming; (c) a debugging light bulb to indicate correct or syntactically incorrect code; (d) Instructions for block usage and to execute the code; (e) A virtual 3D model (Pikachu) to demonstrate the code; (f) A virtual Microbit.

enough that they can be easily interacted with on a mobile phone or a larger device. While using the application, users are guided with the instructions to select blocks and execute code; each block is presented with an explanation about why to use the specific block (based on D3) (Figure 24d).

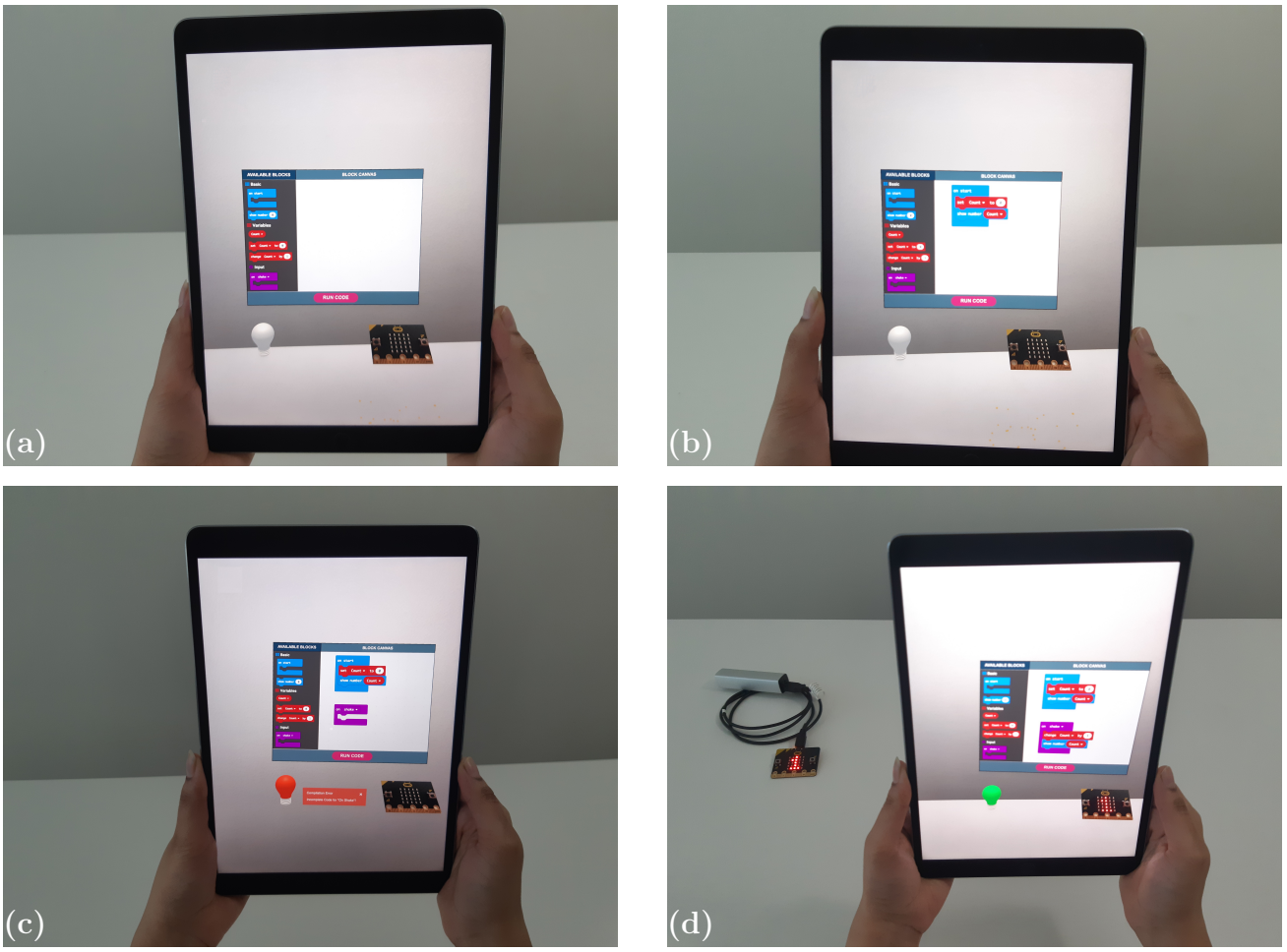


Figure 25: A brief walk-through of *MakeMR*. (a) After blocks have been pre-selected, a user (or student) is presented with the blocks and a blank canvas; (b) User selects appropriate blocks using touch gestures and are placed onto the canvas; (c) If the user selects a wrong block, a red light bulb indicates an error in the code; (d) After a user has created syntactically correct code (i.e. correct blocks fit and the 'puzzle' is complete), they are able to run it on a virtual Microbit and also have it deployed to a physical Microbit via Bluetooth.

When the application is run, the blocks chosen by a teacher are now available for others (i.e. the learners) using the application (Figure 25a). Tapping a block places it on the canvas, where it can then be modified using touch gestures (Figure 25b). Once a user completes the program, they can execute the code by clicking on the "Run" button. If an error occurs, a 'Red' light-bulb appears indicating an error in the users code (Figure 25c). If the code is executable, it is demonstrated on the virtual Microbit and a 'Green' light-bulb indicates the success of the program. If a Microbit is connected via Bluetooth, then the code will be sent to the physical Microbit (Figure 25d). A detailed interface of the prototype is shown in Figure 24.

The prototype supports of three learning activities as described below:

Learning about Microbit: In this learning activity, students learn about the working of Microbit and explore its features like sensors, pins, LEDs, buttons etc. The user scans the Microbit using their smartphone/tablet through which a virtual 3D model of the Microbit is available in MR (Figure 26).

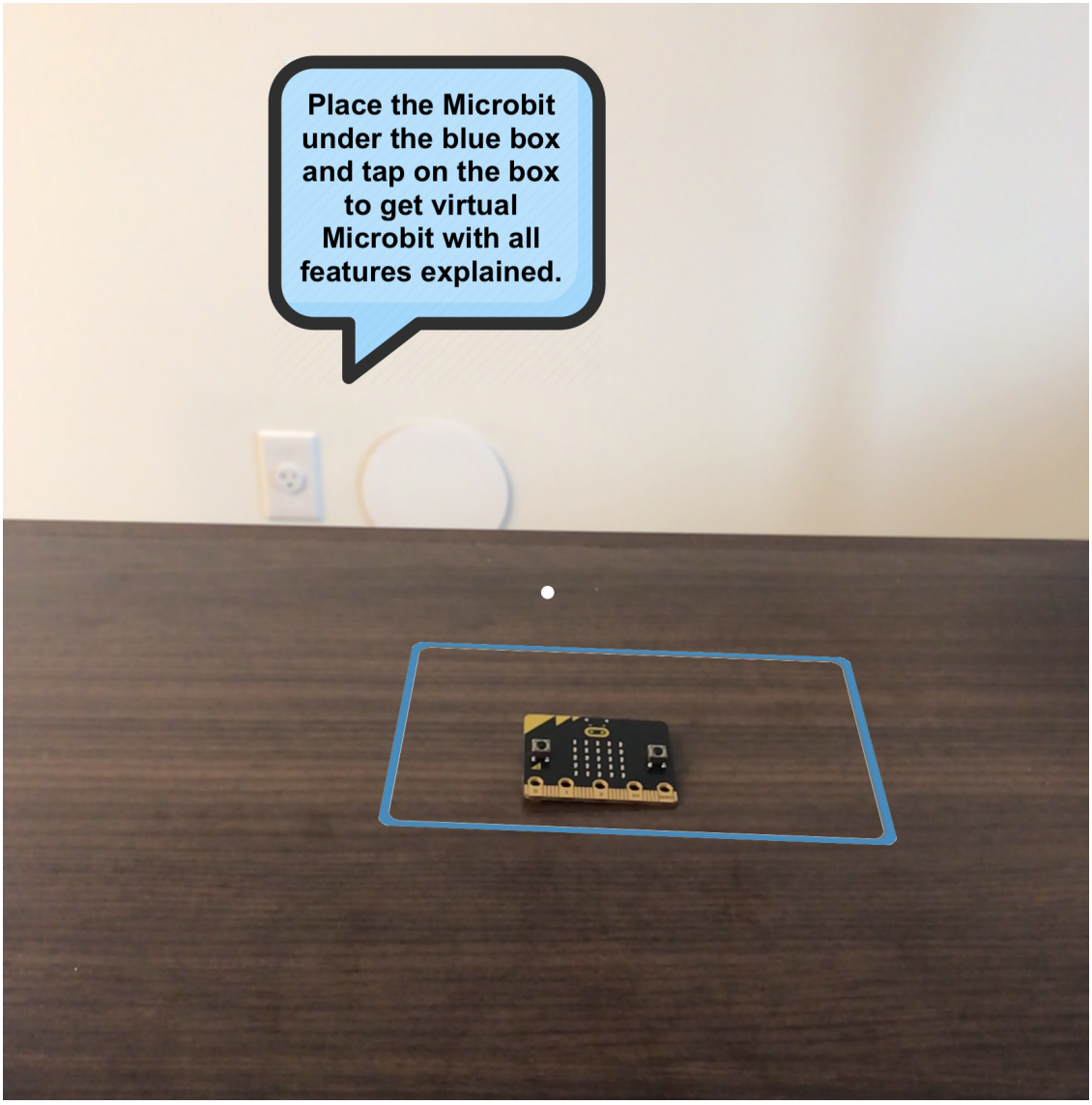


Figure 26: Exercise 1 - Scan the Microbit to get virtual microbit and learn about microbit features on it.

To learn about the various features of Microbit, the user focuses or tap on the various elements of the virtual Microbit, and the information is presented describing about the usage of that feature/element (Figure 27). The idea here is to make the user familiar with the electronic device (in this case, Microbit) with which they are interested in learning programming. This allows the user to learn about the working of the electronic device instantly. Due to the small size of the electronic devices like Microbit, it is difficult for teachers to show and explain each part of the devices to their students in a classroom environment. With instant learning in MR, students and non-programmers can scan an electronic device and learn its working to explore its usage to build and create other programs.

Figure 27: Information about Microbit LEDs is presented when user taps on the LEDs of virtual Microbit

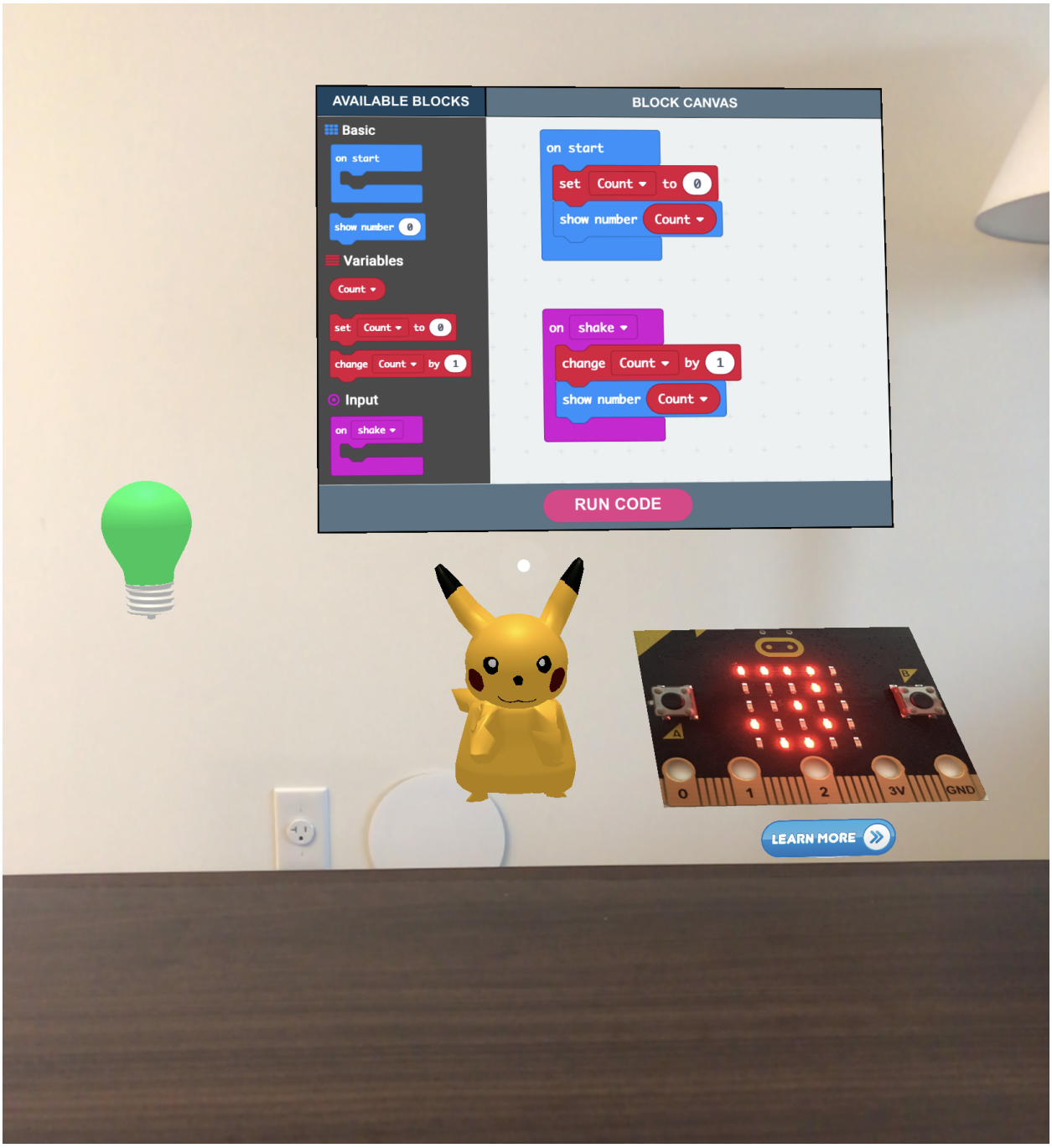


Figure 28: Exercise 2 - Counter Program in MR that counts how many times Pikachu jumps

Programming a jump counter: In this exercise, the user creates a program that counts the number of times the Microbit moves/shakes. This program uses the built-in accelerometer

of the Microbit, which senses the movement in the Microbit. The user could learn about this and other sensors by tapping on the "learn more" button below the virtual Microbit, which redirects the user to the first exercise, which makes it easier for the user to understand the working of this program. To create the program and understand the usage of each block used, the user can follow the instructions shown in the prototype. To demonstrate the real-world use case of the program, a Pikachu is presented on the successful execution of the program, and whenever Pikachu jumps, the counter on the Microbit increases indicating the use case of the program, i.e. how and when the program can be used in real-world scenarios as shown in Figure 28 (For example, it can be used to count the number of jumps, or number of steps or any other movement on the Microbit using the built-in accelerometer).

Building and programming a timing gate: This exercise explains the principle of timing gate and computes the time taken by a car to pass between the two gates. The Microbit has a clock that measures time. The prototype provides an interface where the user creates a program to record the timing when the car passes the two gates and take the difference between the recorded timing to compute the duration between the gates. The prototype also provides virtual objects such as a car, aluminum foils, road, cables and Microbit (Figure 29), using which users can build the timing gate and understand the working of the program. With the help of these virtual objects, users get the opportunity to implement the program and validate it experimentally and see the live results of their program. The sensor is made by tapping strips of foil on the road or the cardboard. The foil strips are connected to the Microbit to appropriate pins using the crocodile clip wires to detect the passing of the car. The bottom of the car is tapped with foil. As the car goes through the gates, it connects both foil strips, closes the circuit and triggers the event "on pin pressed". When the car passes both the gate, the user can see the result on the virtual Microbit, i.e. the time taken by the car to pass the two gates (Figure 30).

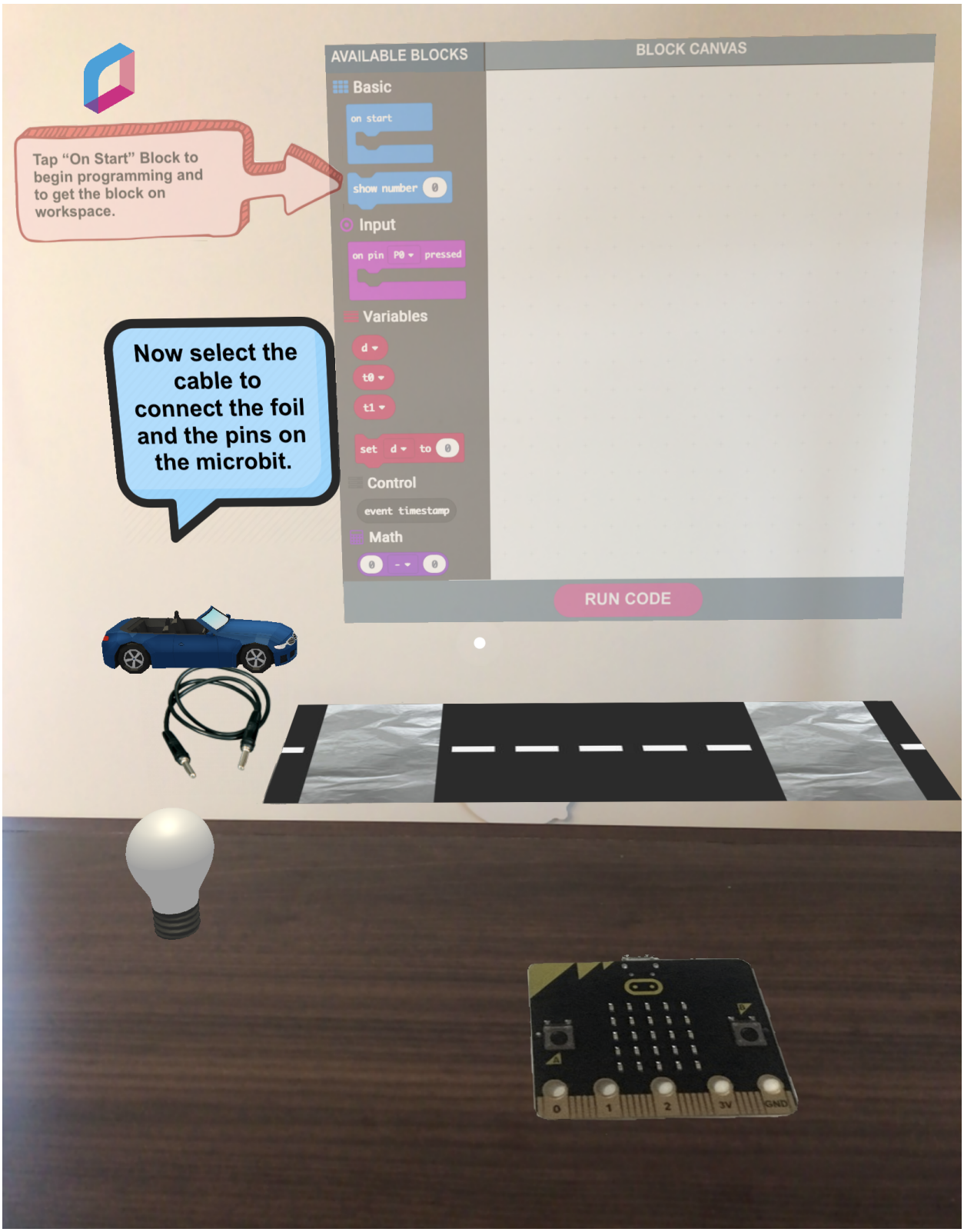


Figure 29: Exercise 3 - Building and programming of timing gate with virtual objects and blocks

Figure 30: Practical implementation of program with results shown on virtual Microbit.

Overall, the aim of our prototype was not to build an extensive list of features, but to provide a functional software prototype to demonstrate how the combination of block-based programming and mixed reality could work for learning block-based programming and validate our design with a small comparative study with non-programmers and a usability study, with teachers. As a result, we provided only basic programming blocks such as logic, variables, colours, sound and light and only simulated light and sound functionality for the virtual Microbit.

4

EVALUATING BLOCK-BASED PROGRAMMING TOOLS IN MR

4.1 STUDY III - COMPARATIVE STUDY

The purpose of this study (see Appendix C for detail) is to evaluate our design approach and objectively analyze the learning performance of novices with a MR-based mobile learning application in comparison to a non-MR based mobile application to learn block-based programming. We want to understand the impact of MR technology and if it could be more beneficial than a non-MR based mobile application to learn block-based programming. The participants of this study were recruited from India to understand the challenges and strategies of people from a different educational culture for learning programming and how non-programmers in India would respond to technology-oriented learning, wherein the textbook is the primary source of knowledge for both the teacher and the students. The learning performance is measured by statistically analyzing programming test scores. This comparative study helps us to understand the impact of mobile MR technology in learning block-based programming, by comparing the programming test results of the group of non-programmers using the MR-based mobile app against those using the non-MR based mobile app to learn block-based programming.

4.1.1 *Protocol*

4.1.1.1 *Participants*

The participants for this study were non-programmers from India. A total of 28 participants were recruited for this study, ranging between 20 to 50 years of age. The study was conducted in-person in India. We recruited via emails, and word-of-mouth recruitment. The participants were randomly assigned to either the experimental group - group A or the control group - group B. Fourteen participants were in group A in which they used the mobile MR prototype to learn block-programming concepts, and fourteen participants were in group B studying same content on a mobile app but without MR.

4.1.1.2 *Procedure*

The study went through four steps. Before starting the learning activity, participants were asked to fill a questionnaire about their background experience with programming, and they took a 10-minute pre-test that aimed to evaluate whether the two groups had an equivalent prior knowledge of basic programming concepts. The programming test consisted of 10 multiple-choice questions with a maximum score of 10. Next, the participants were shown a short demo video on how to use the application. During the learning activity, participants were given two learning exercises in the prototype to program physical electronics like micro:bit using block-programming. The first exercise was to program a counter, which counts the number of jumps or shakes on the micro:bit and the second exercise was to build and program a timing gate with micro:bit, which counts the amount of time taken by a car to pass two gates. The learning materials included text descriptions and virtual 3D models, as explained in the prototype section 3.4. The time taken by the participant to complete

the learning activity was recorded. After the learning activity, participants were asked to complete a 10-minute programming test to assess their learning performance. Similar to the pre-test, the post-test also consisted of 10 multiple-choice questions with a maximum score of 10. Questions for the programming tests were related to the learning activities, and all the questions were of the same difficulty level (See appendix D and E for pre and post study questionnaire). Both the pre-test and the post-test were developed in collaboration with the teachers of STEM Learning Lab. The control group went through the same procedure, except that they used the non-MR based mobile application. Table 1 explains the differences between the two prototypes used for this study. Both the groups were presented with the similar programming interface with same presentation of blocks and instructions to use the blocks (Figure 31), but the learning material for practical implementation and supplementary information, e.g. various features of Microbit was presented asynchronously (due to limited screen space) in a different tab in the browser through videos and images in the mobile app (Figure 32a and Figure 33a). In contrast, the practical implementation of the exercise was presented with virtual objects in MR as shown in Figure 32b, and a digital layer of information was augmented on the virtual Microbit to learn various features of Microbit (Figure 33b).

Learning methods comparison		
	Non-MR	**MR**
Block Presentation	Limited	Limited
Instructions	Text input	Text input
Practical Implementation	Explained with videos and images	Implementation with virtual objects in MR
Learning context and supplementary information	Information received asynchronously	Information received synchronously with the digital layers and virtual objects in MR

Table 1: Learning methods comparison used for this study

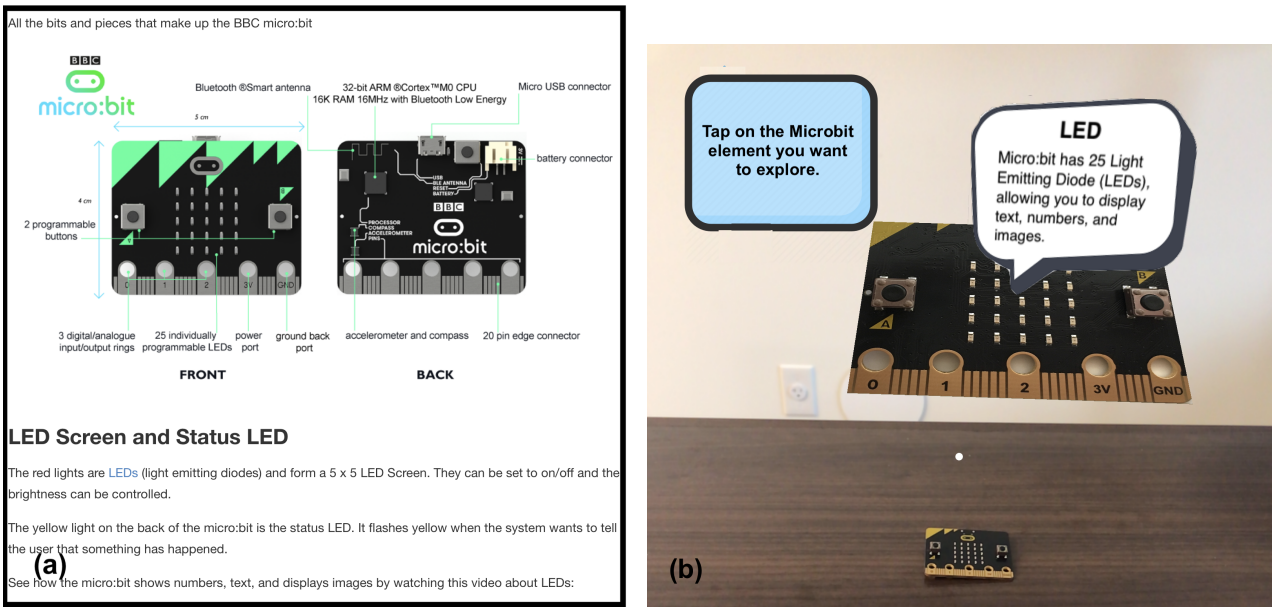


Figure 32: Microbit features explained in (a) non-MR and (b) MR app

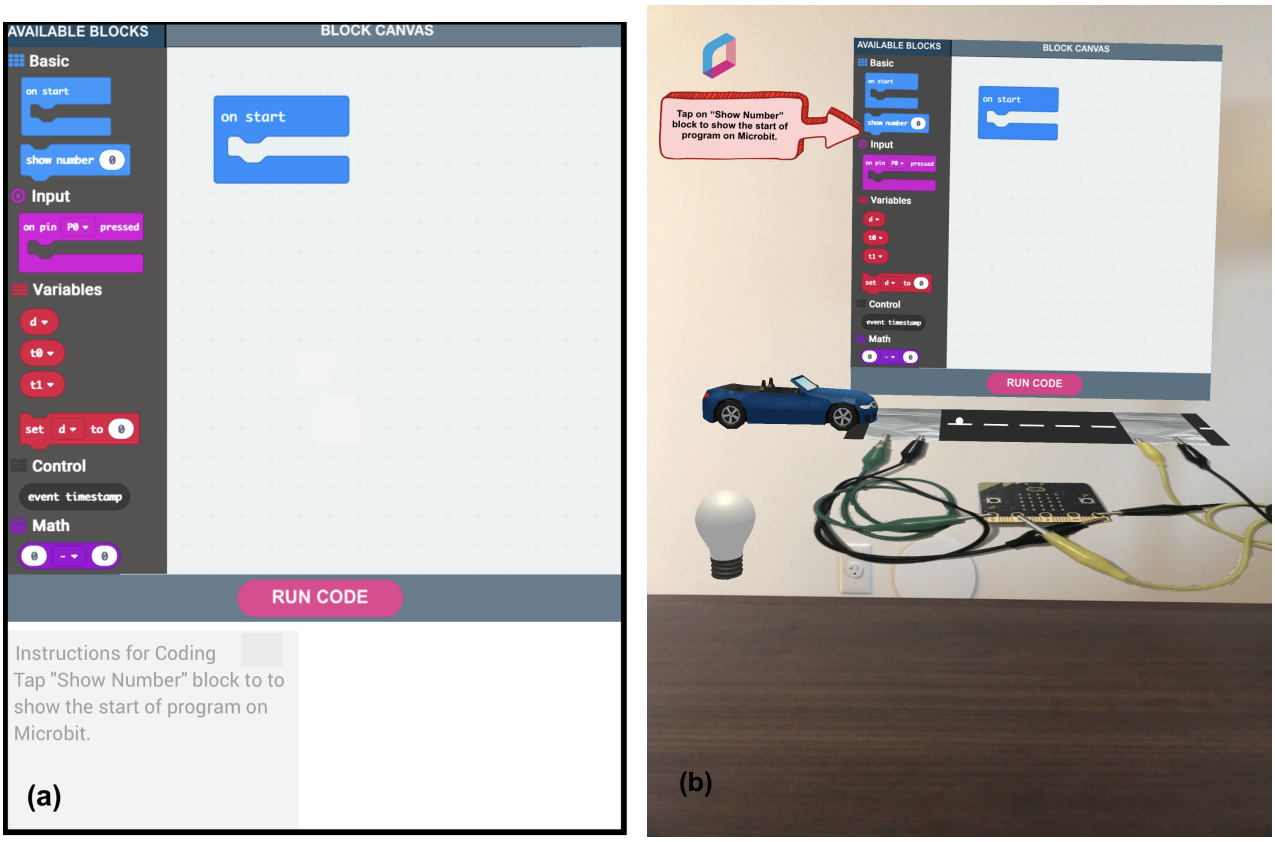


Figure 31: Block presentation in (a) non-MR and (b) MR app

4.1.2 *Statistical Tool and Analysis*

All descriptive statistics and inferential statistical analyses in this study were conducted in the software Statistical Package for Social Sciences (SPSS)[28]. The significance of the tests used in this study is evaluated based on the significance value or P-value. Smaller P-values means that the test is significant. For this study, the significance level was set to 0.05. i.e. if the P-value is higher than 0.05, there is no significant difference in the test and the null

[28]https://www.ibm.com/analytics/spss-statistics-software

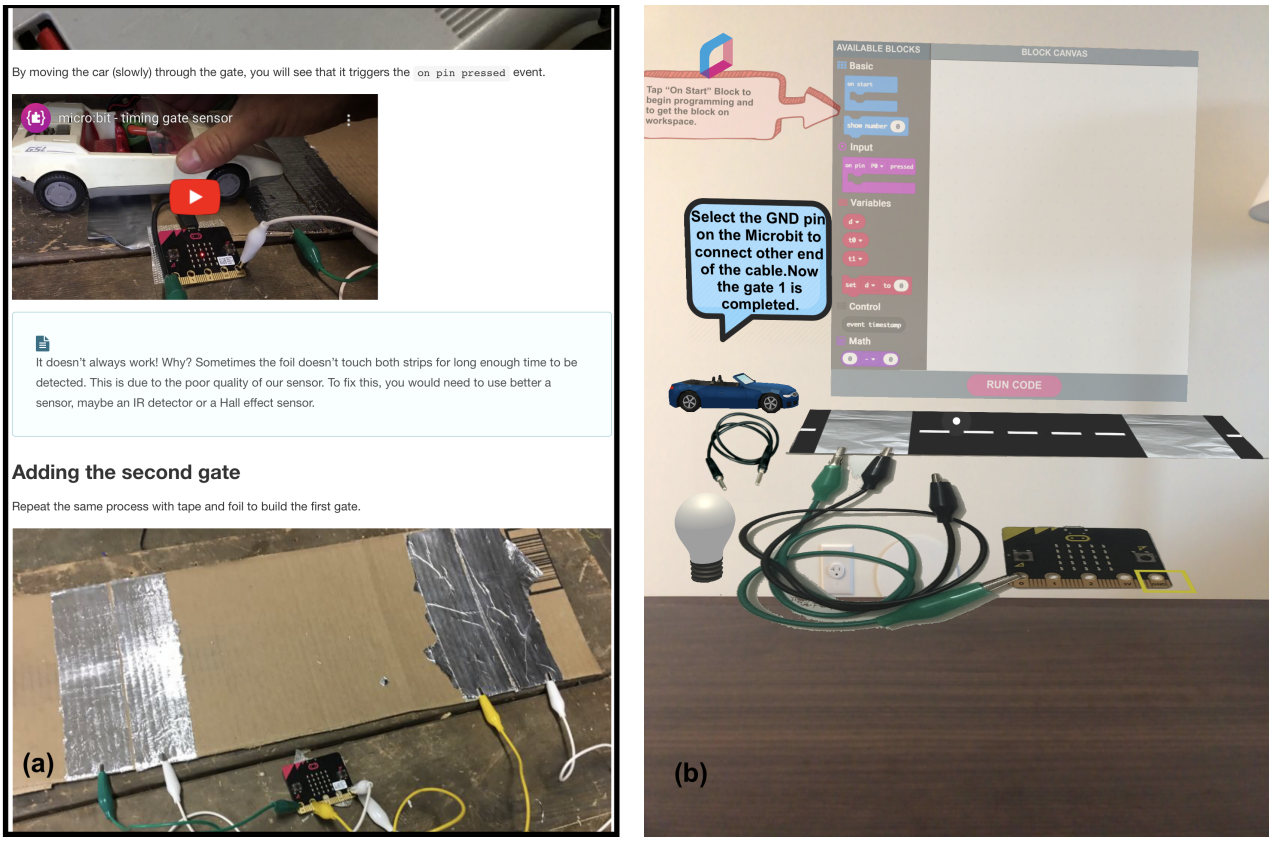


Figure 33: Practical implementation of program explained in (a) non-MR and (b) MR app

hypothesis can be accepted. The null hypothesis for all the tests in this thesis is: There is no significant difference between the two groups.

The statistical test used in this study is independent-sample t-test[29], also known as two-sample t-test. The independent sample t-test is used to compare the means of two sets of scores that are collected from a different set of people or are unrelated groups on the same continuous, dependent variable. It is appropriate to use the independent t-test only if the data passes some assumptions that are required to get valid results [90]. The assumptions for independent sample t-test are as follow-

[29]https://statistics.laerd.com/statistical-guides/independent-t-test-statistical-guide.php

- Assumption 1 - The dependent variable should be measured on a continuous scale, i.e. it should be measured at the interval or ratio level. In this study, the test score would be the dependent variable.

- Assumption 2- The independent variable should consist of two categorical, independent groups.

- Assumption 3 - The two groups are independent, i.e. there is no relationship between observations in each group or between the groups themselves. For example, there must be different participants in each group, with no participant being in more than one group.

- Assumption 4 - The dependent variable should be approximately normally distributed within each group.

- Assumption 5 - There needs to be homogeneity of variance, i.e. the variances of the two groups should be equal in the population.

In this study, test-scores are used as the dependent variable as it serves as a tangible measure for learning performance and fulfils the criteria for independent sample t-test, and the two groups, group A and group B, are used as an independent variable. The participants in each group were independent of the other group, and none of the participants were part of both groups.

Normality: Shapiro-Wilk[30] test was conducted to test whether the pre and post-test scores were produced by normal distribution within each group. The test results were not-significant that is null hypothesis is supported, indicating that the data is normally distributed: $W = 0.894$, $p = 0.093$ ($p > 0.05$) for group A and $W = 0.895$, $p = 0.096$ ($p > 0.05$) for group B as shown in table 2 and 3.

Shaipro-Wilk test results for pre-test			
Group	**Statistic**	**df**	**Sig.**
MR	0.888	14	0.075
Non-MR	0.894	14	0.093

Table 2: Shaipro-Wilk test results for pre-test

Shaipro-Wilk test results for post-test			
Group	**Statistic**	**df**	**Sig.**
MR	0.894	14	0.093
Non-MR	0.895	14	0.096

Table 3: Shaipro-Wilk test results for post-test

Independent Sample t-test					
Group statistics					
	Group	**N**	**Mean**	**Std. Deviation**	**Std. Error Mean**
Pre-test score	MR	14	5.50	1.09	0.29
	Non-MR	14	5.35	1.21	0.32
			Equal variances assumed	**Equal variances not assumed**	
Levene's Test for Equality of Variances	F		0.051		
	Sig.		0.822		
t-test for Equality of Means	t		0.327	0.327	
	df		26	25.70	
	Sig.(2-tailed)		0.746	0.746	
	Mean Difference		.14286	.14286	
	Std. Error Difference		0.43674	0.43674	

Table 4: Result of Independent sample t-test on Pre-test Score

Levene's test[31] for equality of variance was used to test whether the homogeneity of variance assumption was met. The test should not be significant to meet the assumption of the equality of variances. The result of Levene's test, $F(1,26) = 0.763$, $p=0.390$ ($p > 0.05$), indicate that the assumption of homogeneity of variance was met. Levene's test results are shown in the table 4 and 5.

The collected data (see Appendix F for complete data) from pre and post-test passes all the assumptions that are required to be analyzed using an independent t-test to get valid results, and we, therefore, used an independent samples t-test to compare the means. Independent sample t-test was conducted on pre-test scores and post-test scores of the two groups with the following null hypothesis: *All the participants in both the groups have equivalent programming knowledge.*

4.1.3 *Results and Discussion*

Both pre and post-test scores were statistically compared by the independent sample t-test with a significance level of 0.05. The mean and standard deviations of the pre-test were 5.50 and 1.02 for the experimental group, and 5.35 and 1.21 for the control group. The independent t-test results for the pre-test scores showed that the two groups did not differ significantly ($t(26) = 0.327$, $p = 0.746$ ($p < .05$)) meaning that the hypothesis mentioned above is supported, as shown in Table 4, that is, all the participants had statistically equivalent experience before learning the block-programming through the application.

After the learning activities, participants took a post-test. The t-test results for the post-test scores indicate that the difference between the test performance of the two groups was

[30]https://www.sciencedirect.com/topics/psychology/shapiro-wilk-test
[31]http://www.people.vcu.edu/wsstreet/courses/314_20033/Handout.Levene.pdf

Independent Sample t-test					
Group statistics					
	Group	**N**	**Mean**	**Std. Deviation**	**Std. Error Mean**
Post-test score	MR	14	8.35	1.21	0.32
	Non-MR	14	7.35	1.01	0.26

		Equal variances assumed	**Equal variances not assumed**
Levene's Test for Equality of Variances	F	0.763	
	Sig.	0.390	
t-test for Equality of Means	t	2.369	2.369
	df	26	25.14
	Sig.(2-tailed)	0.026	0.026
	Mean Difference	1.00	1.00
	Std. Error Difference	0.42211	0.42211

Table 5: Result of Independent sample t-test on Post-test Score

significant t(26) = 2.369, p = 0.026 (p < .05) (Table 5) and the null hypothesis can be rejected, implying that the average learning performance of the experimental group (M = 8.3, SD = 1.21) was better than that of the control group(M = 7.3, SD = 1.00).

Figure 34 shows the detailed analysis for each question of the post-test. The table shows the number of questions correctly answered by participants of both groups. Questions 3, 4, 6 and 9 were correctly answered by most of the participants of the MR group. These questions are related to the project building part of the learning activity, indicating that the combination of block-programming and MR may be beneficial to gain practical knowledge.

Participants, on average, took slightly longer to complete the learning activity with the non-MR based mobile app. The average time taken by the participants to complete the

QUESTION NO.	AR	NON-AR
Question 1	13	13
Question 2	9	10
Question 3	13	9
Question 4	14	10
Question 5	11	12
Question 6	12	8
Question 7	10	11
Question 8	10	10
Question 9	14	9
Question 10	11	11

Figure 34: Detailed analysis of each questions.

learning activity was 919.5 seconds for the experimental group - group A and 951.8 seconds for the control group - group B. This finding is reasonable since students in the control group had to spend their time reading and going through the videos and images and redirecting from one tab to another.

The results shown above indicate that the mobile MR approach can improve non-programmers' learning performance in learning block-programming concepts. The learning activities with the proposed MR-based prototype, connecting the real-world context with the digital learning resources, can be an efficient method in learning block-programming, especially for gaining practical knowledge. Using the MR-based mobile learning system, the participants learned from the real-world scenarios by interacting with the virtual models and validating the program experimentally in an integrated and organized way. The MR prototype carries multimedia representation, including static images, animations, and 3D virtual objects with which users can interact and gain practical knowledge in a single virtual space. On the other hand, in a traditional approach or a conventional mobile learning approach, the real-world objects

such as Microbit and the corresponding materials that are needed for practical learning (e.g. cables and cardboard) are presented separately and asynchronously. When observing these real-world objects, the students need to either read the corresponding materials from a mobile device or go through a video and put lots of effort into organizing the information by themselves, which prevents them from focusing on the learning objectives.

4.2 STUDY IV - DESIGN CRITIQUE

As we worked very closely with experts (specifically teachers) from STEM Learning Lab while developing *MakeMR*, we asked them to continually provide feedback throughout the various stages of our collaboration, including the *MakeMR* prototype.

In addition to the feedback from STEM Learning Labs, we also conducted follow-up semi-structured interviews with the same 5 teachers from Study II. In these interviews, we used the prototype as a technology probe [41] to demonstrate block-based programming in MR. As 3 of the 5 teachers had limited experience with AR/MR (e.g. only familiar with AR/MR games) and none had any experience teaching with AR/MR, we briefly explained it's principles and showed demo videos of existing AR/MR applications (e.g. Pokemon Go, the Google's Expeditions app [36]) and the MakeMR prototype. Each interview was 30 minutes long. The goal of this study was to discuss the potential of combining block-based programming and AR/MR in education for practical learning, as well as to brainstorm future development ideas. We present the feedback from both the teachers of STEM Learning Labs in general themes below, following a similar methodology to Study II.

4.2.1 *Ease of Interaction*

Generally, the feedback for being able to interact and program with the blocks, as well as the interactions with the virtual and physical Microbit was extremely positive, with teachers noting that "*...it makes their program come alive*" (Teacher 3) and "*this is certainly useful*" (Teacher 1). However, what we found interesting was, after seeing how block-based programming and MR mixed with the Microbit, the teachers we interviewed began suggesting entirely new interactions. For example, several teachers suggested "*...it would be good to have some more things, example math manipulatives and how you can use them to solve problems, if you are able to show that in the movement of them and how to manipulate it that would be really helpful*" (Teacher 2). This is interesting because it suggested that for teachers, having the mental model of a puzzle and demonstrating visually how that puzzle fits in MR would help their students who occasionally got lost when trying to choose what blocks to use, which we observed in Study I and the teachers commented on in Study II.

Teachers also liked the combination of physical and digital, and the linking between the virtual model of the Microbit and the physical one (D4), but several wanted an even tighter integration, such as "*...shaking the hologram in some way similar to the current version of Microbit*" (Teacher 1). For our prototype, we did not fully implement the on-board sensors of the Microbit, as mentioned earlier, but fully mimicking the Microbit and adding a layer of digital information in the MR environment does provide value, as several teachers noted with resourcing being an issue, having a complete virtual replacement is useful.

As we not only limited the amount of blocks presented, but enabled teachers to select the presentation of blocks with instructions for students (D1), we were excited to see that this was well received. One teacher strongly believed that this could help substantially with debugging because she could know what blocks all students were using. "*I always like working*

with coding problems that you can debug because then you can figure out if your code isn't working because maybe you set the colour wrong or maybe you set the order wrong or your code is not working because it is incomplete. It is easy to tell the difference and explain them why is it not working and it makes the debugging a lot easier" (Teacher 4).

4.2.2 *Collaboration in the Classroom*

As we chose to use a mobile device (specifically a tablet) as primary device for *MakeMR* (D2), all teachers felt it would allow them to move around in the classroom more rather than be stuck to a singular desktop, as mentioned by Teacher 3 "*...if schools have ipads/tablets then they can do it standing, move around the room and can see the code, instead of just sitting in front of computer*". The freedom also meant that teachers began focusing on how to collaborate with block-based programming and MR in their classrooms. One teacher noted "*Collaboration feature will be really helpful and would be really appealing for a lot of teachers. What we keep hearing is that there is a big push for the collaborative/co-operative working project and I feel like hologram would be great if it gives a collaborative experience, this is really in demand feature from a teachers perspective*" (Teacher 1).

We also received critical comments on the need for collaboration in *MakeMR* and beyond, for example *"When people can see things collaboratively it really helps, because it takes so many different approaches to solve a problem...[MakeMR] is awesome*" (Teacher 4). While our focus was not on problem solving and collaboration specifically, but instead on block-based programming with MR and the Microbit, there is a clear need to explore this area more in depth, especially as it is currently an under-explored area of research in the context we've described throughout this work.

4.2.3 *Beyond Electronics and Block-based programming*

Despite our limitation in supporting a single electronic device virtually and physically, all teachers saw significant value in combining block based programming in MR with physical electronics, particularly when resources for large classrooms can be a challenge for economically disadvantaged schools, classrooms or students (D4). Teachers found it interesting to have the virtual model that connects the code with the real-world context, and demonstrating it visually in MR was helpful for them to translate lessons into a real (albeit virtual) concept (D5).

One interesting outcome for teachers, however, was that it facilitated a discussion on how to combine different physical artifacts and domains in MR and blocked-based programming. For example, one teacher noted *"Instead of just a single model sitting there if there is an artifact through which they can explore and make discoveries out of that, e.g. for social studies if we have artifacts for different countries and let them see the 3d image and then they are able to get there and make discoveries based on that"* (Teacher 2). We limited our exploration of block-based programming and MR to strictly enabling teachers to demonstrate how to program electronics like the Microbit, however, there was a clear need to utilize this approach in other areas such as in IoT [53], and warrants future exploration.

4.3 STUDY LIMITATIONS

The findings of this study should be considered in light of some limitations that are important to discuss when interpreting the results. First, we used the prototype as a technology probe, which limited our ability to gather feedback from our other potential users: children. Moreover, the qualitative measures were solely present from teachers from a local STEM

learning lab who have experience with block-based programming languages. This lead to a small sample size for semi-structured interviews.

Second, the comparison is based on quantitative measures only, and no qualitative measures are present that could provide insights from novices/non-programmers. Furthermore, the participants of the comparative study were older than primary or high school students. The results for the comparative study show a significant statistical difference between the two groups. However, the difference in the mean of the test scores of the two groups is not very large and therefore based on the given the sample size and age group, it is difficult to establish a practical difference between the groups.

Another limitation that may affect the generalizability of the findings is that we focused only on the mobile-based MR systems only. The goal of this thesis was to propose a novel way for novices to learn block-based programming in an integrated way with easily accessible resources such as smartphones and tablets. However, other classes of MR, e.g. projection-based systems and a head-mounted display, can be used to analyze the impact of MR in learning block-based programming. Surely, given a wider range of participants and including other classes of MR technology, more robust and generalized insights about the combination of MR and block-based programming and design considerations for creators of block-based programming tools in mixed reality can be provided.

5

CONCLUSION AND FUTURE WORK

The work discussed in this thesis examined the challenges that are faced by the existing 2D block-based programming systems, and a potential solution to address these challenges. Chapter 1 described the motivations and research questions behind this thesis work. An overview of block-based programming environments, different types of block-based programming environments, and how these can be combined with mobile MR technology for enhanced practical learning experiences are provided in Chapter 2. This was to provide a background that is necessary for understanding the challenges that novices face when working with block-based programming tools and how mobile learning with MR can solve some of these challenges. From this understanding, an observational study and multiple semi-structured interviews were then performed, along with a sample implementation presented in Chapter 3. We evaluated the prototype using a comparative study with non-programmers and gathered feedback from teachers using semi-structured interviews to answer the research questions presented in Section 1.5. This was presented in Chapter 4.

5.1 CONTRIBUTIONS

The first contribution of this thesis is an overview of the current research space of block-programming languages for learning basic programming concepts. This is presented in the form of a literature review in Chapter 2 that discusses the background of block-programming, different tools and applications for block-programming and the various domains where block-programming is used. The survey helps to understand the advantages of block-programming and how is it helpful for novices to learn programming concepts. This answered the first research question, *"What is the current state of research in block-based programming environments?"* that was directed towards understanding current research around block-programming. I also explore the challenges that novices face with existing block-programming applications and what potential solutions (e.g. mobile learning and mixed reality) could be for these challenges.

The second research question, *"What are the challenges with existing block-programming environment from an educators' perspective?"* was investigated in the observational study presented in Section 3.1 and discussed in detail in Section 3.1.3. The results from the study revealed that several areas, such as practical learning with virtual objects to build projects and artifacts can be addressed with alternative modalities such as mobile MR. These results were further validated in our next study with teachers from a local STEM education company. The feedback from semi-structured interviews was used to synthesize design considerations for a block-based programming environment, which is the second contribution of this thesis. This answers the third research question, *"What are the key design considerations for tools that combine block-based programming and mixed reality?"* and they are discussed in detail in Section 3.3. These design considerations can be applied when implementing a system that combines block-programming and mixed reality to enhance learning experiences

for novices. To support the scenarios and use cases described from our observations and interviews, I designed a MR-based mobile application to learn block-based programming, called MakeMR, discussed in detail in Section 3.4, which is the third contribution of this thesis. The experimental research indicates that the prototype improves students' learning performance programs by linking the real-world context with digital learning resources. It offers an integrated way of learning by utilizing the unlimited virtual space in MR where users can learn programming concepts and build the project in a single space.

The fourth contribution of this thesis is a preliminary design critique and a discussion about combining mixed reality and block-based programming and its impact on learning basic programming concepts. This could serve as a starting point for educators and non-programmers to select a block-based programming environment to teach and learn block-programming. The fourth research question, *"What is the impact of MR-based mobile app on the performance of a non-programmer while learning programming concepts?"* was investigated in a comparative study presented in Section 4.1. The results from the study indicate that the MR-based mobile application is more effective for learning basic concepts of block-programming than a non-MR based mobile application. The impact of the MR-based mobile app on learning block-programming is further discussed in our fourth study, which was described in Section 4.2. This was accomplished by gathering feedback from teachers about the potential of combining block-based programming and MR, particularly in classroom environments, as well as future development ideas. This answers the fifth research question, *"What challenges and opportunities does a system that combines block-based programming and mixed reality face?"*.

5.2 FUTURE WORK

In this thesis work, in addition to the design considerations, I presented a system that combined MR and block-based programming for novice programmers. Future work is to implement a fully functional, MR-based block-programming application and to conduct workshops with a STEM learning partner and follow it throughout the 5-day process, as discussed earlier in Section 3.1. A complete one week workshop with a larger group of teachers and students will be critical in understanding their perspective better towards this combination of block-programming and MR. Following this, we plan to extend our approach to support collaboration by using a shared MR experience for students and teachers, i.e. sharing the same view of the programming interface and virtual models while programming.

Secondly, as this thesis focused only on the educational domain and some specific tasks using Microbit, there are several areas left to explore for future work. They include robotics, programming IoT devices and electronics, and even machine learning.

Lastly, a key area is to examine gestures for block-based programming applications in MR. During the studies with the implementation described in this thesis, participants (i.e. teachers) showed interest in mid-air interactions rather than touch-based interactions on a screen in MR. I believe that focusing on designing an effective gesture set for block-based programming tools in MR is important and can provide additional value beyond just the scenarios in education.

5.3 CONCLUSION

Block-programming has been used widely for teaching and learning basic programming concepts. In this thesis, we explore the current state of research for block-programming and

the challenges with existing block-programming environments and how mobile learning and mixed reality (MR) can be combined to solve some of these challenges. I present design considerations for the implementation of tools that combine mobile-based mixed reality and block-based programming. I approached the design by working with a local STEM organization, who frequently offers block-based programming workshops to teach programming and computational concepts to youth using the BBC Microbit, in addition to other block-based tools such as Scratch, Minecraft and App Inventor. Through our collaboration, I observed a workshop and performed follow-up semi-structured interviews with teachers, which were then synthesized into design considerations. I then implemented a prototype system — *MakeMR* — by utilizing these design considerations and then evaluated the prototype and design approach using a comparative study with non-programmers and performed another follow-up interview with teachers, using MakeMR as a technology probe and present a preliminary design critique and discussion about combining mixed reality and block-based programming. The study results indicate that learning from scenarios that present relevant materials (e.g. text, images, virtual models, animations) and supplementary digital information in an integrated and organized way can benefit students in improving their learning performance.

Overall, this thesis demonstrates that the combination of block-based programming and mobile MR may provide value to learn programming, such as enhancing the practical learning environment, as well as potentially making technology and resources more accessible to learners.

Part I

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] Adafruit. Adafruit Circuit Playground Express, July 2019. Retrieved July 7, 2019 from https://learn.adafruit.com/adafruit-circuit-playground-express/overview. (Cited on page 10.)

[2] Adfruit. Bluetooth : Adafruit Industries, Unique & fun DIY electronics and kits, July 2019. Retrieved July 7, 2019 from https://www.adafruit.com/index.php?main_page=category&cPath=255. (Cited on page 57.)

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

[4] K. Al-Tahat. The Impact of a 3d Visual Programming Tool on Students' Performance and Attitude in Computer Programming: A Case Study in Jordan. *Journal of Cases on Information Technology (JCIT)*, 21(1):52–64, 2019. (Cited on pages 16, 17, 18, 25, and 42.)

[5] Alice. Alice – Tell Stories. Build Games. Learn to Program. (Cited on page 25.)

[6] Andrew J. Ko, B. Myers, and H. Aung. Six Learning Barriers in End-User Programming Systems. In *2004 IEEE Symposium on Visual Languages - Human Centric Computing*, pages 199–206, Rome, 2004. IEEE. (Cited on page 30.)

[7] R. T. Azuma. A Survey of Augmented Reality. page 48. (Cited on page 8.)

[8] J. Bacca, S. Baldiris, R. Fabregat, S. Graf, and D. Kinshuk. Augmented Reality Trends in Education: A Systematic Review of Research and Applications. *Educational Technology and Society*, 17:133–149, Oct. 2014. (Cited on pages 8, 10, and 41.)

[9] N. Bak, B.-M. Chang, and K. Choi. Smart Block: A Visual Programming Environment for SmartThings. volume 2, pages 32–37, 2018. (Cited on pages 10 and 36.)

[10] D. Bau. Droplet, a blocks-based editor for text code. *J. Comput. Sci. Coll.*, 30(6):138–144, June 2015. (Cited on page 2.)

[11] D. Bau, J. Gray, C. Kelleher, J. Sheldon, and F. Turbak. Learnable Programming: Blocks and Beyond. *Communications of the ACM*, 60(6):72–80, May 2017. arXiv: 1705.09413. (Cited on pages 4, 5, 28, 29, 44, and 56.)

[12] A. Begel and E. Klopfer. Starlogo tng: An introduction to game development. 01 2005. (Cited on page 17.)

[13] A. Begel and M. Resnick. Logoblocks: A graphical programming language for interacting with the world. 1996. (Cited on page 16.)

[14] c. Block coding. Block based coding. (Cited on pages 28 and 29.)

[15] Blockly. Blockly. (Cited on page 22.)

[16] P. Bontá, A. Papert, and B. Silverman. Turtle, art, turtleart. In *Proceedings of the 2010 Conference Constructionism*, 2010. (Cited on page 17.)

[17] B. D. Boulay. Some Difficulties of Learning to Program:. *Journal of Educational Computing Research*, Jan. 1995. (Cited on page 32.)

[18] V. Braun and V. Clarke. Using thematic analysis in psychology. *Qualitative Research in Psychology*, 3(2):77–101, Jan. 2006. (Cited on page 51.)

[19] B. Broll, A. Lédeczi, P. Volgyesi, J. Sallai, M. Maroti, A. Carrillo, S. L. Weeden-Wright, C. Vanags, J. D. Swartz, and M. Lu. A visual programming environment for learning distributed programming. In *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education*, SIGCSE '17, pages 81–86, New York, NY, USA, 2017. ACM. (Cited on pages 16 and 32.)

[20] K. Charmaz and L. L. Belgrave. Grounded Theory. In *The Blackwell Encyclopedia of Sociology*. American Cancer Society, 2015. (Cited on page 51.)

[21] K.-H. Cheng and C.-C. Tsai. Affordances of Augmented Reality in Science Learning: Suggestions for Future Research. *Journal of Science Education and Technology*, 22(4):449–462, Aug. 2013. (Cited on page 8.)

[22] T. Chiang, S. Yang, and G.-J. Hwang. An Augmented Reality-based Mobile Learning System to Improve Students' Learning Achievements and Motivations in Natural Science Inquiry Activities. *Educational Technology and Society*, 17:352–365, Oct. 2014. (Cited on pages 2, 11, and 58.)

[23] H. Crompton. A Diachronic Overview of Mobile Learning: A Shift Toward Student-Centered Pedagogies. page 10. (Cited on page 6.)

[24] H. Crompton, D. Burke, and K. H. Gregory. The use of mobile learning in PK-12 education: A systematic review. *Computers & Education*, 110:51–63, July 2017. (Cited on page 6.)

[25] J. Cubillo, S. Martín, M. Castro, G. Díaz, A. Colmenar, and I. Botički. A learning environment for augmented reality mobile learning. volume 2015-February, 2015. (Cited on pages 7, 10, 38, and 39.)

[26] N. Dass, J. Kim, S. Ford, S. Agarwal, and D. H. P. Chau. Augmenting Coding: Augmented Reality for Learning Programming. In *Proceedings of the Sixth International Symposium of Chinese CHI*, ChineseCHI '18, pages 156–159, New York, NY, USA, 2018. ACM. event-place: Montreal, QC, Canada. (Cited on pages 2, 10, 42, and 44.)

[27] C. Dede. Planning for Neomillennial Learning Styles. page 6. (Cited on page 40.)

[28] J. Devine, J. Finney, P. de Halleux, M. Moskal, T. Ball, and S. Hodges. Makecode and codal: Intuitive and efficient embedded systems programming for education. *SIGPLAN Not.*, 53(6):19–30, June 2018. (Cited on pages 3 and 10.)

[29] M. Dunleavy, C. Dede, and R. Mitchell. Affordances and Limitations of Immersive Participatory Augmented Reality Simulations for Teaching and Learning. *Journal of Science Education and Technology*, 18:7–22, Feb. 2009. (Cited on page 40.)

[30] M. O. M. El-Hussein and J. C. Cronje. Defining Mobile Learning in the Higher Education Landscape. page 10. (Cited on pages 6 and 37.)

[31] E. FitzGerald, A. Adams, R. Ferguson, M. Gaved, Y. Mor, and R. Thomas. Augmented reality and mobile learning: the state of the art. Oct. 2012. (Cited on pages 38 and 39.)

[32] S. Fleck and G. Simon. An Augmented Reality Environment for Astronomy Learning in Elementary Grades: An Exploratory Study. page 10. (Cited on page 8.)

[33] Gameblox. Gameblox. (Cited on page 34.)

[34] T. Glushkova and I. Academy. Application of Block Programming and Game-Based Learning to Enhance Interest in Computer Science. *Journal of innovations and sustainability*, 2:21–32, Mar. 2016. (Cited on pages 1, 5, and 33.)

[35] Google. Blockly. Retrieved June 16, 2019 from https://developers.google.com/blockly/. (Cited on pages 3 and 32.)

[36] Google. Bring your lessons to life with Expeditions, July 2019. Retrieved July 7, 2019 from https://edu.google.com/intl/en_ca/products/vr-ar/expeditions/. (Cited on page 80.)

[37] A. Hamzah, A. Persada, and A. Hidayatullah. Towards a framework of mobile learning user interface design. pages 1–5, 2018. (Cited on pages 6 and 7.)

[38] R. Holwerda and F. Hermans. A usability analysis of blocks-based programming editors using cognitive dimensions. volume 2018-October, pages 217–225, 2018. (Cited on page 33.)

[39] M. S. Horn, C. Brady, A. Hjorth, A. Wagh, and U. Wilensky. Frog pond: A codefirst learning environment on evolution and natural selection. In *Proceedings of the 2014 Conference on Interaction Design and Children*, IDC '14, pages 357–360, New York, NY, USA, 2014. ACM. (Cited on page 17.)

[40] Y.-S. Hsu, Y.-H. Lin, and B. Yang. Impact of augmented reality lessons on students' STEM interest. *Research and Practice in Technology Enhanced Learning*, 12(1):2, July 2016. (Cited on page 40.)

[41] H. Hutchinson, W. Mackay, B. Westerlund, B. B. Bederson, A. Druin, C. Plaisant, M. Beaudouin-Lafon, S. Conversy, H. Evans, H. Hansen, N. Roussel, and B. Eiderbäck. Technology probes: Inspiring design for and with families. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '03, pages 17–24, New York, NY, USA, 2003. ACM. (Cited on page 80.)

[42] M.-B. Ibáñez and C. Delgado-Kloos. Augmented reality for stem learning: A systematic review. *Computers Education*, 123:109 – 123, 2018. (Cited on page 11.)

[43] M. B. Ibáñez, A. D. Serio, D. Villarán-Molina, and C. D. Kloos. Experimenting with electromagnetism using augmented reality: Impact on flow student experience and educational effectiveness. *Computers & Education*, 71:1–13, 2014. (Cited on pages 39 and 56.)

[44] A. Inventor. MIT App Inventor | Explore MIT App Inventor. (Cited on page 33.)

[45] A. M. Inventor. App inventor. (Cited on page 34.)

[46] K. Kahn, R. Megasari, E. Piantari, and E. Junaeti. AI programming by children using snap! Block programming in a developing country. volume 2193, 2018. (Cited on page 10.)

[47] D. Kamilali and C. Sofianopoulou. MICROLEARNING AS INNOVATIVE PEDAGOGY FOR MOBILE LEARNING IN MOOCS. page 5, 2015. (Cited on pages 6 and 37.)

[48] L.-M. Karch. Get Your Kids Started Coding With These Programming Languages. (Cited on pages 22 and 27.)

[49] H. Kaufmann and D. Schmalstieg. Mathematics and geometry education with collaborative augmented reality. *Computers & Graphics*, 27(3):339–345, June 2003. (Cited on page 8.)

[50] C. Kelleher, J. Maloney, P. Medlock-Walton, E. Patton, and D. Wendel. Invited panel: The future of blocks programming. In *2017 IEEE Blocks and Beyond Workshop (B B)*, pages 99–101, Oct. 2017. (Cited on page 44.)

[51] M. Kesim and Y. Ozarslan. Augmented Reality in Education: Current Technologies and the Potential for Education. *Procedia - Social and Behavioral Sciences*, 47:297–302, Dec. 2012. (Cited on page 39.)

[52] J. Kim, K. Marotta, J. Leo, S. Agarwal, S. Li, and D. H. Chau. Mixed Reality for Learning Programming. In *Proceedings of the Interaction Design and Children on ZZZ - IDC '19*, pages 574–579, Boise, ID, USA, 2019. ACM Press. (Cited on page 43.)

[53] B. Knowles, S. Beck, J. Finney, J. Devine, and J. Lindley. A scenario-based methodology for exploring risks: Children and programmable iot. In *Proceedings of the 2019 on Designing Interactive Systems Conference*, DIS '19, pages 751–761, New York, NY, USA, 2019. ACM. (Cited on pages 10 and 83.)

[54] R. Kraleva, V. Kralev, and D. Kostadinova. A methodology for the analysis of block-based programming languages appropriate for children. *Journal of Computing Science and Engineering*, 13(1):1–10, 2019. (Cited on pages 2, 4, 16, 18, and 30.)

[55] A. Kurihara, A. Sasaki, K. Wakita, and H. Hosobe. A Programming Environment for Visual Block-Based Domain-Specific Languages. *Procedia Computer Science*, 62:287–296, Jan. 2015. (Cited on page 4.)

[56] T.-J. Lin, H. B.-L. Duh, N. Li, H.-Y. Wang, and C.-C. Tsai. An investigation of learners' collaborative knowledge construction performances and behavior patterns in an augmented reality simulation system. *Computers & Education*, 68:314–321, Oct. 2013. (Cited on page 8.)

[57] S. Lye and J. Koh. Case Studies of Elementary Children's Engagement in Computational Thinking Through Scratch Programming: Foundations and Research Highlights.

In *Computational Thinking in the STEM Disciplines: Foundations and Research Highlights*, pages 227–251. Aug. 2018. (Cited on page 19.)

[58] J. Maloney, M. Resnick, N. Rusk, B. Silverman, and E. Eastmond. The scratch programming language and environment. *Trans. Comput. Educ.*, 10(4):16:1–16:15, Nov. 2010. (Cited on page 31.)

[59] mBlock. Playing with colored-blocks coding to Learn Programming | mblock-coding robots for kids, Nov. 1995. (Cited on page 19.)

[60] P. Medlock-Walton, K. J. Harms, and E. T. Kraemer. Blocks-based Programming Languages: Simplifying Programming for Different Audiences with Different Goals. page 2. (Cited on pages 33 and 34.)

[61] B. Microbit. Micro:bit Educational Foundation, July 2019. Retrieved July 7, 2019 from https://microbit.org/. (Cited on pages 10 and 46.)

[62] Microsoft. Microsoft MakeCode for micro:bit. Retrieved July 7, 2019 from https://makecode.microbit.org/. (Cited on pages 3, 32, 45, 46, and 57.)

[63] Microsoft. Kodu, July 2019. Retrieved July 7, 2019 from https://www.microsoft.com/en-us/research/project/kodu/. (Cited on page 10.)

[64] Microsoft. Microsoft makecode arcade, July 2019. Retrieved July 7, 2019 from https://arcade.makecode.com. (Cited on page 17.)

[65] MIT-education. Gameblox, Jan. 2018. (Cited on page 35.)

[66] S. N. H. Mohamad, A. Patel, R. Latih, Q. Qassim, L. Na, and Y. Tew. Block-based programming approach: challenges and benefits. In *Proceedings of the 2011 International*

Conference on Electrical Engineering and Informatics, pages 1–5, July 2011. ISSN: 2155-6822. (Cited on page 28.)

[67] M. Monga, M. Lodi, D. Malchiodi, A. Morpurgo, and B. Spieler. Learning to program in a constructionist way. page 15, 2018. (Cited on pages 1 and 2.)

[68] B. Nielsen, H. Brandt, and H. Swensen. Augmented Reality in science education–affordances for student learning. *Nordic Studies in Science Education*, 12:157, Sept. 2016. (Cited on page 39.)

[69] Nodered. Node-RED. (Cited on page 35.)

[70] NodeRED. Node-RED, Aug. 2019. Page Version ID: 911072905. (Cited on page 35.)

[71] F. R. Ortega, S. Bolivar, J. Bernal, A. Galvan, K. Tarre, N. Rishe, and A. B. Barreto. Towards a 3 D Virtual Programming Language to Increase Number of Women in Computer Science Education. 2017. (Cited on page 42.)

[72] S. Papert. *Mindstorms: children, computers, and powerful ideas*. Basic Books, New York, 1980. (Cited on page 2.)

[73] E. W. Patton, M. Tissenbaum, and F. Harunani. MIT App Inventor: Objectives, Design, and Development. In S.-C. Kong and H. Abelson, editors, *Computational Thinking Education*, pages 31–49. Springer, Singapore, 2019. (Cited on page 25.)

[74] D. N. Perkins, S. Schwartz, and R. Simmons. Instructional strategies for the problems of novice programmers. In *Teaching and learning computer programming: Multiple research perspectives*, pages 153–178. Lawrence Erlbaum Associates, Inc, Hillsdale, NJ, US, 1988. (Cited on page 32.)

[75] C. Pimmer and N. Pachler. Mobile learning in the workplace. Unlocking the value of mobile technology for work-based education. pages 193–204. Jan. 2014. (Cited on page 7.)

[76] P. P. Ray. A Survey on Visual Programming Languages in Internet of Things, 2017. (Cited on page 36.)

[77] F. J. Rodríguez, K. M. Price, J. Isaac, K. E. Boyer, and C. Gardner-McCune. How block categories affect learner satisfaction with a block-based programming interface. In *2017 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, pages 201–205, Oct. 2017. (Cited on pages 17 and 32.)

[78] P. Sarkar, J. S. Pillai, and A. Gupta. ScholAR: A Collaborative Learning Experience for Rural Schools Using Augmented Reality Application. In *2018 IEEE Tenth International Conference on Technology for Education (T4E)*, pages 8–15, Dec. 2018. (Cited on page 8.)

[79] K. L. Schrier. Revolutionizing History Education: Using Augmented Reality Games to Teach Histories. page 290. (Cited on page 8.)

[80] Scratch. Scratch - Imagine, Program, Share. Retrieved June 16, 2019 from https://scratch.mit.edu/. (Cited on pages 3, 21, 32, and 45.)

[81] Scratch. Scratch programming, Dec. 2015. (Cited on page 22.)

[82] P. Seow, C.-K. Looi, M.-L. How, B. Wadhwa, and L.-K. Wu. Educational Policy and Implementation of Computational Thinking and Programming: Case Study of Singapore. In S.-C. Kong and H. Abelson, editors, *Computational Thinking Education*, pages 345–361. Springer, Singapore, 2019. (Cited on page 19.)

[83] B. E. Shelton and N. R. Hedley. Using augmented reality for teaching Earth-Sun relationships to undergraduate geography students. In *The First IEEE International Workshop Agumented Reality Toolkit,*, pages 8 pp.–, Sept. 2002. (Cited on page 8.)

[84] L. Shuib, S. Shamshirband, and M. H. Ismail. A review of mobile pervasive learning: Applications and issues. *Computers in Human Behavior*, 46:239–244, May 2015. (Cited on page 7.)

[85] A. Sipitakiat, P. Blikstein, and D. P. Cavallo. Gogo board: Augmenting programmable bricks for economically challenged audiences. In *Proceedings of the 6th International Conference on Learning Sciences*, ICLS '04, pages 481–488. International Society of the Learning Sciences, 2004. (Cited on page 11.)

[86] W. Slany. Tinkering with pocket code, a scratch-like programming app for your smartphone. 2014. (Cited on page 17.)

[87] Snap. Snap! Build Your Own Blocks. (Cited on page 23.)

[88] P. Software. Picoblocks, July 2019. Retrieved July 7, 2019 from `https://http://www.picocricket.com/download.html`. (Cited on page 17.)

[89] S. Somanath, L. Oehlberg, J. Hughes, E. Sharlin, and M. C. Sousa. 'maker' within constraints: Exploratory study of young learners using arduino at a high school in india. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, CHI '17, pages 96–108, New York, NY, USA, 2017. ACM. (Cited on page 11.)

[90] SPSS. Independent t-test in SPSS Statistics - Procedure, output and interpretation of the output using a relevant example | Laerd Statistics. (Cited on page 74.)

[91] STEM. STEM Learning Lab |, July 07. Retrieved July 7, 2019 from https://stemlearninglab.com/. (Cited on page 45.)

[92] S. TNG. StarLogo TNG, Oct. 2016. (Cited on page 34.)

[93] J. Vincur, M. Konopka, J. Tvarozek, M. Hoang, and P. Navrat. Cubely: Virtual Reality Block-based Programming Environment. page 4, 2017. (Cited on pages 1, 10, and 43.)

[94] K. Wang, C. McCaffrey, D. Wendel, and E. Klopfer. 3d game design with programming blocks in starlogo tng. In *Proceedings of the 7th International Conference on Learning Sciences*, ICLS '06, pages 1008–1009. International Society of the Learning Sciences, 2006. (Cited on page 17.)

[95] C. Weichel, M. Lau, D. Kim, N. Villar, and H. W. Gellersen. Mixfab: A mixed-reality environment for personal fabrication. In *Proceedings of the 32Nd Annual ACM Conference on Human Factors in Computing Systems*, CHI '14, pages 3855–3864, New York, NY, USA, 2014. ACM. (Cited on page 11.)

[96] D. Weintrop and U. Wilensky. Robobuilder: A program-to-play constructionist video game. In *Proceedings of the 2012 Conference Constructionism*, 2012. (Cited on page 17.)

[97] D. Weintrop and U. Wilensky. To block or not to block, that is the question: Students' perceptions of blocks-based programming. pages 199–208, 2015. (Cited on pages 1, 3, 11, 16, and 30.)

[98] D. Weintrop and U. Wilensky. To block or not to block, that is the question: Students' perceptions of blocks-based programming. In *Proceedings of the 14th International Conference on Interaction Design and Children*, IDC '15, pages 199–208, New York, NY, USA, 2015. ACM. (Cited on page 2.)

[99] D. Weintrop and U. Wilensky. Using Commutative Assessments to Compare Conceptual Understanding in Blocks-based and Text-based Programs. July 2015. (Cited on pages 16 and 32.)

[100] M. H. Wilkerson-Jerde. *The Deltatick Project: Learning Quantitative Change in Complex Systems with Expressive Technologies*. PhD thesis, Evanston, IL, USA, 2012. AAI3499652. (Cited on page 17.)

[101] L. E. Winslow. Programming pedagogy—a psychological overview. *ACM SIGCSE Bulletin*, 28(3):17–22, Sept. 1996. (Cited on page 32.)

[102] D. Wolber, H. Abelson, E. Spertus, and L. Looney. *App Inventor 2: Create Your Own Android Apps*. O'Reilly Media, Beijing, 2nd edition edition, Oct. 2014. (Cited on pages 10 and 17.)

[103] S. C.-Y. Yuen, G. Yaoyuneyong, and E. Johnson. Augmented Reality: An Overview and Five Directions for AR in Education. *Journal of Educational Technology Development and Exchange*, 4(1), June 2011. (Cited on pages 40, 55, 56, and 58.)

[104] N. Zamin, H. A. Rahim, K. S. Savita, E. Bhattacharyya, M. Zaffar, and S. N. K. M. Jamil. Learning Block Programming using Scratch among School Children in Malaysia and Australia: An Exploratory Study. In *2018 4th International Conference on Computer and Information Sciences (ICCOINS)*, pages 1–6, Aug. 2018. (Cited on page 5.)

Part II

APPENDIX

A

ETHICS - OBSERVATION STUDY AND SEMI-STRUCTURED INTERVIEWS

Name of Researcher, Faculty, Department, Telephone & Email:

Title of Project:

Assessing the Impact of Augmented Reality on Collaborative Programming Education

This consent form, a copy of which has been given to you, is only part of the process of informed consent. If you want more details about something mentioned here, or information not included here, you should feel free to ask. Please take the time to read this carefully and to understand any accompanying information.

The University of Calgary Conjoint Faculties Research Ethics Board has approved this research study

Purpose of the Study

The aim of this research is to explore and assess the impact of Augmented Reality on collaborative programming education. The goal is to combine concepts of collaboration in AR, with concepts of AR in education to support collaboration in a learning environment and analyze a shared learning environment that can help students in understanding fundamental programming concepts.

What Will I Be Asked To Do?

If you agree to participate in this study, you will be asked to participate in the following different research activities:

1. You will be asked a short questionnaire about technology use and your background experience. This questionnaire should take no more than 5 minutes to complete.

2. You will be asked a brief set of questions about the teaching process and challenges faced in a short (no more than 20 minutes) interview.

The whole process was designed to last no longer than approximately 30 minutes, although if you feel that you can discuss longer on certain topics or may have additional insights you think that are important, feel free to talk about it.

Your participation in this research is voluntary. You may refuse to participate altogether or in part. You may withdraw from participation in this study at any time without penalty or loss of benefits

What Type of Personal Information Will Be Collected?

Should you agree to participate we will be audiotaping your responses and taking notes. Other than these audio recordings, no other personal identifying information (such as your name) will be collected. By default, in all written publications and presentations based on this research, you will remain anonymous and your comments from the interviews will be referred to with either a participant number or a pseudonym.

In order to better communicate the results of this research in written publications and presentations or if required we will also be collecting some photographs to better understand teaching environment setting i.e. classroom organization, desks etc. Every face in any of these photos will be pixelated to ensure privacy and anonymity. We will never reveal your name in association with your image.

Please note that, where intended reporting of photographed images includes a public display, the researchers will have no control over any future use by others who may copy the images and repost them in different formats or contexts, including online

I grant permission to be audiotaped: Yes: ___ No: ___

I grant permission to be photographed: Yes: ___ No: ___

I grant permission to have my company's name used: Yes: ___ No: ___

Are there Risks or Benefits if I Participate?

There is no known harms or risks associated with the participation in this study.

What Happens to the Information I Provide?

Participation in this research is completely voluntary and confidential. You are free to discontinue participation at any time during the study. Any information you contribute up to the point at which you choose to discontinue, your participation will be retained and used in the study. No one except the researchers will be allowed to see or hear any personally identifiable information unless you have given permission for us to share photographs of you in our interview or focus group, in publications or presentations of this research. The audio tapes, questionnaires and interview data will be kept on password-protected university computers or in a locked cabinet only accessible by the researchers. The data will be stored for five years, after which it will be permanently erased.

Signatures

Your signature on this form indicates that 1) you understand to your satisfaction the information provided to you about your participation in this research project, and 2) you agree to participate in the research project.

In no way does this waive your legal rights nor release the investigators, sponsors, or involved institutions from their legal and professional responsibilities. You are free to withdraw from this research project at any time. You should feel free to ask for clarification or new information throughout your participation.

Participant's Name: (please print) _____

Participant's Signature: _____

Date: _____

Researcher's Name: (please print) _____

Researcher's Signature: _____

Date: _____

Questions/Concerns

If you have any further questions or want clarification regarding this research and/or your participation, please contact:

A copy of this consent form has been given to you to keep for your records and reference. The investigator has kept a copy of the consent form.

B

INTERVIEW QUESTIONNAIRE

Background and Experience:

1. How many years of in general teaching experience do you have?
2. What areas do you focus on?
3. How many years of experience do you have in teaching programming?
4. What material do you use for teaching programming? e.g. books, online videos, guided tutorials, hands-on material etc.
5. Do you have prior experience with using augmented reality?
    a. If yes, where would you rank yourself - Novice, Intermediate, or Expert
    b. How many years of experience do you have in using AR apps or AR games?

General Teaching Questions:

1. How many sessions do you usually run during a workshop?
2. How long is each session?
3. What are the different steps of each session?

   (these questions will be repeated for every step)
    a. What specific issues do you encounter for step X?

       (follow-up question to get more details about the issue)
        i. Is this issue due to human factors or hardware limitations or software limitations?
4. How frequently do students get stuck and need your help while teaching?
5. How comfortable are you with the material and current teaching techniques?

C

ETHICS - COMPARATIVE STUDY

Name of Researcher, Faculty, Department, Telephone & Email:

Title of Project:

Comparative study (AR-based and Non-AR based mobile application for block programming)

This consent form, a copy of which has been given to you, is only part of the process of informed consent. If you want more details about something mentioned here, or information not included here, you should feel free to ask. Please take the time to read this carefully and to understand any accompanying information.

The University of Calgary Conjoint Faculties Research Ethics Board has approved this research study - REB19-1667.

Purpose of the Study

The aim of this research is to understand how an AR-based mobile application can be effective for learning block-based programming. The goal is to understand the impact of AR-based mobile applications on the performance of non-programmers for learning programming.
We are interested in testing our approach, and understanding the ways in which our approach can be improved. These findings are to guide further studies and implementations to better conform to user interests.

What Will I Be Asked To Do?

If you agree to participate in this study, you will be asked to participate in the following different research activities:

1. Participants will be asked to complete a short questionnaire about their background experience with programming.
2. A short demo on how to use the application.
3. Participants will go through two learning exercises in the application to program physical electronics like microbit. Microbit is a pocket sized computer with sensors, LEDs, and buttons that can be used to program and create games and different science projects.

- First exercise will be building and programming a timing gate with microbit which

count the amount of time taken by a car to pass the two gates.

- Second exercise will be a step counter which count the no. of steps or shakes on the microbit.

4. After completing the learning exercise, participants will be asked to complete in a post-study questionnaire to assess learning performance.
The time taken by the participant to complete the survey will be recorded.

The whole process was designed to last no longer than approximately 60 minutes, although if you feel that you can discuss longer on certain topics or may have additional insights you think that are important, feel free to talk about it.

Your participation in this research is voluntary. You may refuse to participate altogether or in part. You may withdraw from participation in this study at any time without penalty or loss of benefits

What Type of Personal Information Will Be Collected?

Should you agree to participate, you will be asked to provide your background experience with programming and your age.

There are several options for you to consider if you decide to take part in this research. You can choose all, some or none of them. Please put a check mark on the corresponding line(s) that grants us your permission to:

I wish to remain anonymous: Yes: ___ No: ___

I grant permission to save questionnaire data: Yes: ___ No: ___

I grant permission for material created to be collected and shared in publications or presentations of this research: Yes: ___ No: ___

Are there Risks or Benefits if I Participate?

There are no known harms or risks associated with participation in this study.

What Happens to the Information I Provide?

Participation in this research is completely voluntary and confidential. You are free to discontinue participation at any time during the study and you can withdraw your data within 2 weeks after completing the final questionnaire. Any information you contribute up to the point at which you choose to discontinue, your participation will be retained and used in the study. No one except the researchers will be allowed to see or hear any personally identifiable information unless you have given permission for us to share photographs of you, in publications or presentations of this research. The questionnaire data will be encrypted and will be kept on password-protected university computers or in a locked cabinet only accessible by the researchers. The data will be stored for five years, after which it will be permanently erased.

Signatures

Your signature on this form indicates that 1) you understand to your satisfaction the information provided to you about your participation in this research project, and 2) you agree to participate in the research project.

In no way does this waive your legal rights nor release the investigators, sponsors, or involved institutions from their legal and professional responsibilities. You are free to withdraw from this research project at any time. You should feel free to ask for clarification or new information throughout your participation.

Participant's Name: (please print) _____

Participant's Signature: _____

Date: _____

Researcher's Name: (please print) _____

Researcher's Signature: _____

Date: _____

Questions/Concerns

If you have any further questions or want clarification regarding this research and/or your participation, please contact:

A copy of this consent form has been given to you to keep for your records and references. The investigator has kept a copy of the consent form.

D

PRE-STUDY QUESTIONNAIRES AND TEST

Pre-Study Questionnaire

Background

1. Age _____

2. Do you have any experience with programming?

   None **Novice** **Intermediate** **Advanced** **Expert**

3. Do you have any experience with block-programming?

   None **Novice** **Intermediate** **Advanced** **Expert**

Programming test

1. What is a "loop"?

   A. allows a program to run continuously until told otherwise
   B. an outcome produced depending on a decision made by computer
   C. identifies errors in the code and fix them
   D. a memory location that stores something

2. What is "string"?

   A. Images
   B. Text, number, symbols
   C. a counter
   D. an input

3. What is an IF statement?

   A. a comparison statement
   B. a conditional statement
   C. a set of instructions that are repeated until a certain condition is met
   D. None of the above

4. What are “variables” used for in programming?

A. displaying something on screen
B. storing data for later use
C. executing a sequence of instructions
D. all of the above

5. What is the output for the program

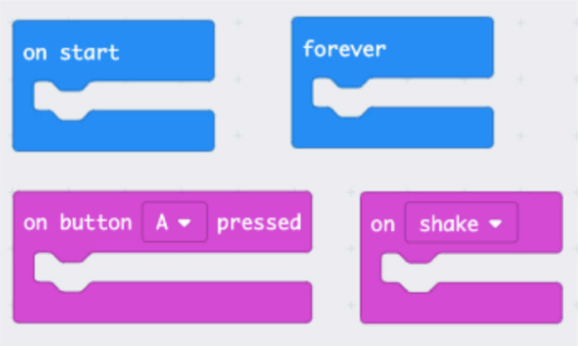
```
var sum =  21;
if ( sum != 20 )
  print("You win ");

else
  print("You lose  ");


 print("the prize.");
```

A. You win
B. You lose
C. You win the prize.
D. You lose the prize.

6. Which block will repeat your code?



A. On start
B. Forever
C. On button A
D. On shake

7. If you want power from the device (Microbit), what pin would you need?

   A. GND
   B. 0
   C. 1
   D. 3V

8. Arrays are collection of the same data type under a single identifier.

   A. True
   B. False

9. This type of variable has only two values; true or false.

   A. Number
   B. String
   C. Boolean
   D. Loop

10. This type of variable holds a string of alphanumeric characters.

    A. Number
    B. String
    C. Boolean
    D. Loop

E

POST-STUDY QUESTIONNAIRES AND TEST

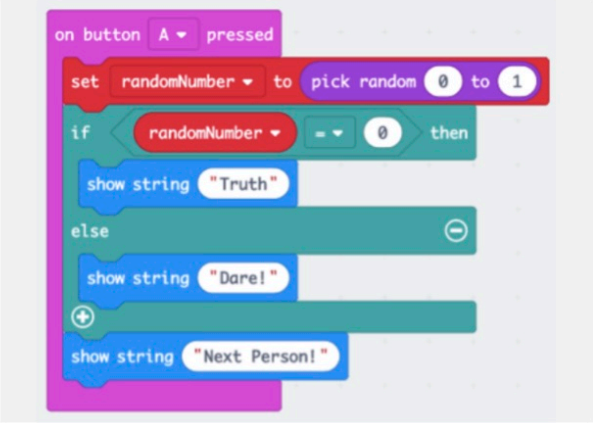
Post-Study Questionnaire

Programming questions

1. Which event is triggered when buttons (A and B) are pressed?

   A. On Button Pressed
   B. On Pin Pressed
   C. On Shake
   D. Show Number

2. What will be displayed if the random number is 1?



   A. “Truth”
   B. “Dare!”
   C. “Next person!”
   D. “Dare” “Next Person!”

3. Which pin is used to close the circuit?

   A. 0
   B. 1
   C. 2
   D. GND

4. Which device on the back of the Micro:bit lets you detect movement?

   A. Compass
   B. Bluetooth
   C. RAM
   D. Accelerometer

5. The LED lights on the Micro:bit are for?

   A. input
   B. output
   C. both input and output
   D. none of the above

6. Which of these detects the earth's magnetic field, allowing you to detect which direction microbit is facing?

   A. Temperature sensor
   B. Accelerometer
   C. Compass
   D. Pins

7. Which block will assign a value to a variable ?

   A. On start
   B. Forever
   C. On button A
   D. On shake

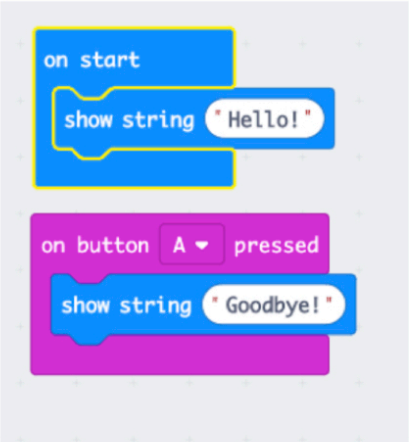
8. How many LED lights does a micro:bit have?

   A. 6x6 - 36 lights
   B. infinite amount
   C. 5x5 - 25 lights
   D. it does not have any LED lights

9. Which two large pins are related to power supply?

   A. 0 and 1
   B. 0 and 2
   C. 3V and GND
   D. 1 and 2

10. What would happen at the start on your micro:bit screen?



   A. 'Hello'
   B. 'Goodbye'
   C. 'Hello' 'Goodbye'
   D. Hello Goodbye

F

RAW RESULT FOR COMPARATIVE STUDY

Participant	Age	Programming Exp.	Block Programming Exp.	Pre-test Score	Post-test Score	Time taken to complete Exercise	Group
P1	34	Novice	Novice	6	10	896	AR
P2	23	Novice	Novice	7	9	864	AR
P3	26	Novice	None	4	9	810	AR
P4	27	Novice	None	6	9	1150	Non-AR
P5	27	None	None	4	7	978	Non-AR
P6	30	Novice	None	6	9	756	AR
P7	32	Novice	Novice	5	7	974	Non-AR
P8	35	None	None	5	7	708	AR
P9	44	None	None	4	7	1094	Non-AR
P10	26	Novice	None	4	6	1078	Non-AR
P11	47	None	None	5	6	738	AR
P12	22	Novice	Novice	6	8	792	AR
P13	28	None	None	4	9	905	AR
P14	31	Novice	None	6	8	1032	Non-AR
P15	49	None	None	5	7	814	AR
P16	37	None	None	4	7	1020	Non-AR
P17	35	Novice	Novice	8	9	1124	Non-AR
P18	29	Novice	None	6	9	885	AR
P19	33	None	None	5	7	779	AR
P20	24	Novice	None	7	8	1059	Non-AR
P21	28	Novice	Novice	7	10	930	AR
P22	30	Novice	None	6	8	996	Non-AR
P23	27	Novice	Novice	5	8	984	Non-AR
P24	36	None	None	5	6	1058	Non-AR
P25	34	Novice	Novice	6	7	996	Non-AR
P26	34	None	None	7	9	908	AR
P27	29	Novice	None	4	8	863	AR
P28	30	None	None	5	6	1174	Non-AR