



INTRODUCTORY PROGRAMMING ECOSYSTEM FOR CHILDREN WITH MOBILE APPLICATION

Anastasios Tsimplinas

Master's thesis
November 2014
Information Technology

TAMPEREEN AMMATTIKORKEAKOULU
Tampere University of Applied Sciences

ABSTRACT

Tampereen ammattikorkeakoulu
Tampere University of Applied Sciences
Degree Program in Information Technology

ANASTASIOS TSIMPLINAS:
Introductory Programming Ecosystem for Children with Mobile Application

Master's thesis 73 pages
November 2014

Programming has an enormous presence in everyday life of 21 century. New generation students are surrounded by computer technology and will possibly do in the future an occupation that has not been invented yet. Digital literacy is the ability to understand and use digital technologies effectively for everyday tasks. Digital literacy is as important for children today as reading and writing skills.

The aim of this thesis was to design and implement a mobile application for tablets that could introduce children to the basics of programming logic with an easy and interactive way. Children by playing with a friendly user interface will be able to understand better of what happens inside computers and also improve their math and logic skills. Using the tablet, children can develop their code by arranging different shapes-pieces-images-blocks that represent simple programming commands as part of a game. The blocks include basic functions as “move”, loops as “repeat” and conditions as “if”. Additionally children they could see the logic results and actions in reality as interaction with a Lego Mindstorms™ EV3 robot. This will make them also more curious with the magical world of robotics.

This master's thesis starts with an introduction on the importance of teaching young children concepts of programming and we continue with the exploration of the background and current state solutions in the area of children programming. After the taxonomy of the various programming environments we present comparative studies between the different interfaces. Based on the comparisons and studies we have explored, we propose a mobile application for tablets that is isomorphic with a tangible programming language that will create a full introductory programming ecosystem, ready to bridge the gap between the tangible and graphical solutions on the area of programming for children. As a conclusion we present the different issues raised during the design and development phase of our application and the future work we intent to carry out.

Keywords: children programming, mobile solution, robots, tangible programming

ACKNOWLEDGEMENTS

There many people to thank and acknowledge for all their support, help and motivation they have given over the last two years. This thesis is dedicated to my parents Theodoros and Theodora Tsimplina for their endless support and encouragement throughout my life.

I would like to sincerely thank my supervisor Esa Kujansuu for his guidance during this study. I have been fortunate to have an advisor who gave me the freedom to explore on my own, and at the same time the guidance to recover when my steps faltered. I would like also to thank Theodosios Sapounidis for the long discussions that helped me sort out the technical details of my work and lectures on related topics that helped me improve my knowledge in the programming for kids area.

A very special thanks goes out to Konstantinos Miliotis and Danai Skournetou for our philosophical debates, exchanges of knowledge, skills, and venting of frustration during my graduate program, which helped enrich the experience.

TABLE OF CONTENTS

1 INTRODUCTION	6
2 BACKGROUND	7
2.1 Children and programming	7
2.2 Text based languages	8
2.2.1 Text based programming	8
2.2.2 Text based programming languages	8
2.3 Graphical based languages.....	11
2.3.1 Graphical programming.....	11
2.3.1 Graphical based programming languages.....	12
2.4 Tangible based languages	16
2.4.1 Tangible programming	16
2.4.2 Tangible programming systems.....	17
2.5 Mobile based systems	24
3 STUDIES	28
3.1 Comparisons between the different interfaces	28
3.2 Comparisons in programming.....	29
3.3 Summary of limitations regarding tangible programming tools	30
4 DESIGN AND DEVELOPMENT	32
4.1 Interaction design for kids	32
4.2 Icon Design	33
4.3 The Lego Robot	37
4.3.1 Connection with the robot	37
4.3.2 Motors and Sensors	38
4.4 Mock ups	40
4.5 Software Requirements.....	42
4.5.1 The Purpose of the Project.....	42
4.5.2 Goals of the Project	43
4.5.3 The Stakeholders	44
4.5.4 Work Partitioning	48
4.5.5 Data Model	51
4.5.6 Product Boundary	53
4.5.7 Functional & Non-functional software requirements	58
5 SUMMARY AND FUTURE WORK	65
5.1 Issues that arose during problem analysis	65
5.1.1 Design new icons.....	65
5.1.2 Size of icons	65
5.1.3 Length of program script	66
5.1.4 Connection to robot	67
5.1.5 Operating system-Software distribution	67
5.1.6 Number of robots supported	68
5.1.7 Hardware constraints	68
5.1.8 Marketing strategy.....	68

5.2 Conclusion69

REFERENCES.....70

1 INTRODUCTION

Not long ago educating children to become adept at reading and writing was considered enough to provide them with the necessary skills to explore the world of knowledge. But a quiet revolution has started in the past few years in education matters. Digital literacy is considered as an important trait as reading and writing. Many countries like the United Kingdom and Estonia are incorporating into their educational curriculums lessons of programming, even from the first grades of elementary school.

By departing from the classic approach that computers are just like cars,-someone need not know about internal combustion engines in order to drive a car-, educating children in the art of programming has many obvious and substantial benefits. By promoting team work, sharpening problem solving skills, learning to create algorithms in a children-friendly manner will be more important in the years to come than just learning a new foreign language, or how to paint. Without exaggeration we might see coding as the new *lingua franca* and who is a better ambassador for this new universal language than children, the future and hope for every society.

In this thesis we discuss the topic of educational programming software for children. We provide an overview of existing technologies utilizing different interfaces to educate children, like text-based programming environments, graphical programming environments, tangible programming systems and mobile systems. We suggest a mobile graphical isomorphic equivalent of a tangible programming system which will operate on a mobile device, e.g. tablet. Furthermore our system will utilize a Lego Mindstorms™ robot connecting wirelessly to the tablet, which will perform like an actor for playing out the various programming scripts. The user in mind is any child or classroom of children wanting to learn to program in a fun and interactive manner, but we believe that our approach is better suited for children aged 4~10.

2 BACKGROUND

2.1 Children and programming

In contradiction to what is happening to automated knowledge activities the ones that require knowledge or skills that were acquired by repeating practice (e.g. the skills of writing, reading, multiplication), the process of problem solving presupposes a mental function in which we need to develop different strategies to approach the problem. When trainees -children- learn a new programming language (Logo for instance) in order to accomplish a given task, what really matters is that apart from the end result or the programming language itself, is the user experience. By experience we mean the process of developing the necessary problem solving strategies, coming up with ideas and testing their validity, dealing with errors on problem diagnosis in a positive manner, increasing a children's confidence in its own judgment, since the tutor shares the same belief and in general the preoccupation of the apprentice with the process of learning.

The pedagogical value of these activities that require thought, is that they enable the child to learn to think more effectively and in other areas apart from programming, either by adopting more flexible and adaptive strategies and logic, or by accelerating the transition to more advanced mental stages that mold new knowledge to long-term gnosis. Subsequently learning to program with children must first be an immersive and fun experience. In the following subsections we present an overview of the different programming interfaces with chronological order of appearance.

2.2 Text based languages

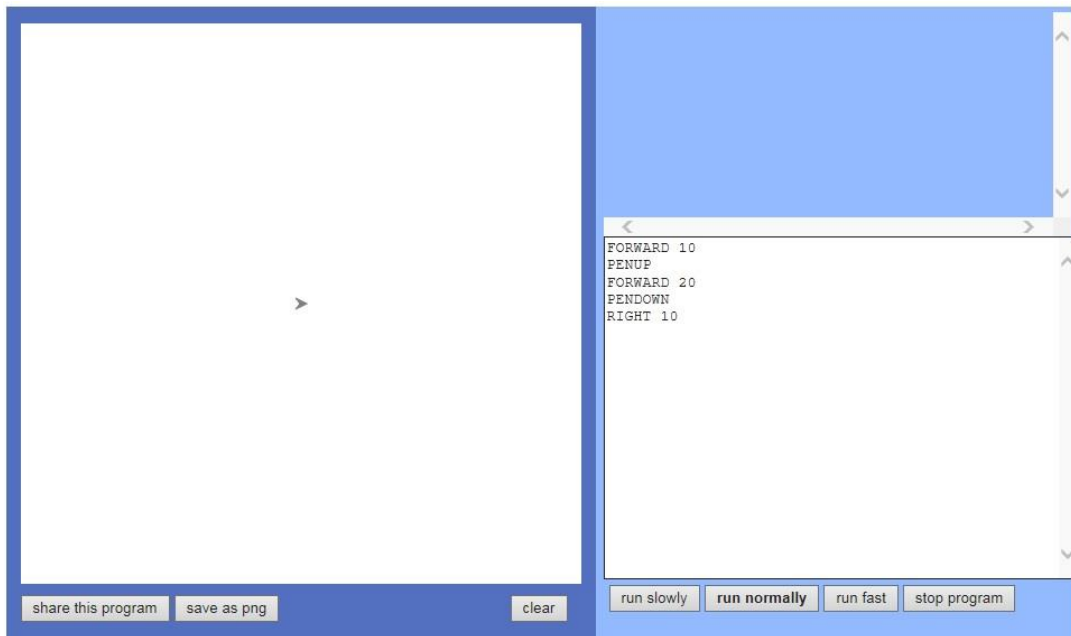
2.2.1 Text based programming

Text based programming implies that the user will type commands-or select them from a menu- that will form a structured program which will be either compiled or interpreted. The output of the program will be either in the form of messages or more likely in some form of on-screen graphical representation (e.g. the movement of a turtle). As in every text based programming language mastering the syntax of the language takes an initial amount of time which might make the learning process for children less attractive. It has been pointed out that text based languages (Kelleher, C., & Pausch, R., 2005) are more suitable as programming learning aids for children from ages 10 and above that already have some level of experience from graphical and tangible programming systems.

2.2.2 Text based programming languages

Logo is an educational programming language (Logo webpage 2014) designed in 1967 by Daniel G. Bobrow, Wally Feurzeig, Seymour Papert and Cynthia Solomon. Today the language is remembered mainly for its use of "turtle graphics", in which commands for movement and drawing produced line graphics either on screen or with a small robot called a "turtle". The language was originally conceived to teach concepts of programming related to LISP programming language and only later to enable what Papert called "body-syntonic reasoning" where students could understand (and predict and reason about) the turtle's motion by imagining what they would do if they were the turtle. There are substantial differences between the many dialects of Logo, and the situation is confused by the regular appearance of turtle graphics programs that mistakenly call themselves Logo. Logo is generally known as an interpreted language, although recently there have been developed compiled Logo dialects—such as Lhogho or Liogo. It is a compromise between a sequential programming language with block structures, and a functional programming language.

Logo's most-known feature is the turtle (derived originally from a robot of the same name), an on-screen "cursor" that showed output from commands for movement and small retractable pen, together producing line graphics. It has traditionally been displayed either as a triangle or a turtle icon (though it can be represented by any icon). Turtle graphics were added to the Logo language by Seymour Papert in the late 1960s to support Papert's version of the turtle robot, a simple robot controlled from the user's workstation that is designed to carry out the drawing functions assigned to it using a small retractable pen set into or attached to the robot's body.



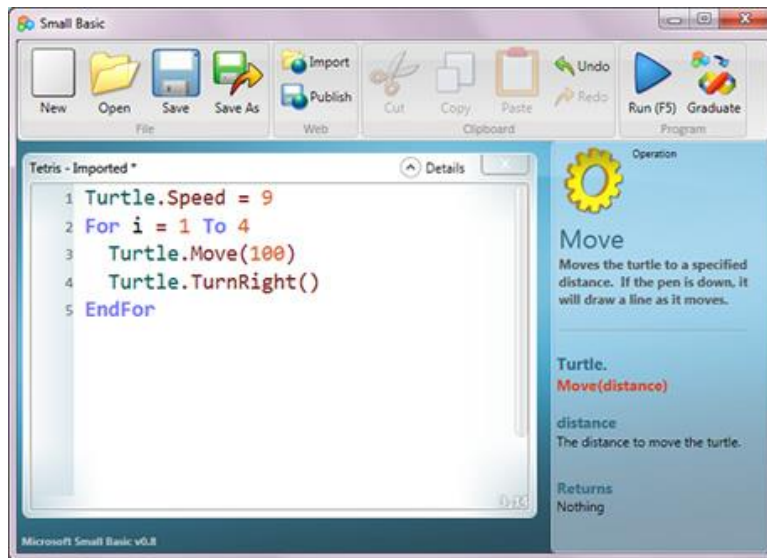
PICTURE 1. A screenshot of a Logo interpreter

Small Basic is a project (Small Basic webpage 2014) that is focused at making programming accessible and easy for beginners. It consists of three distinct pieces:

- The Language
- The Programming Environment
- Libraries

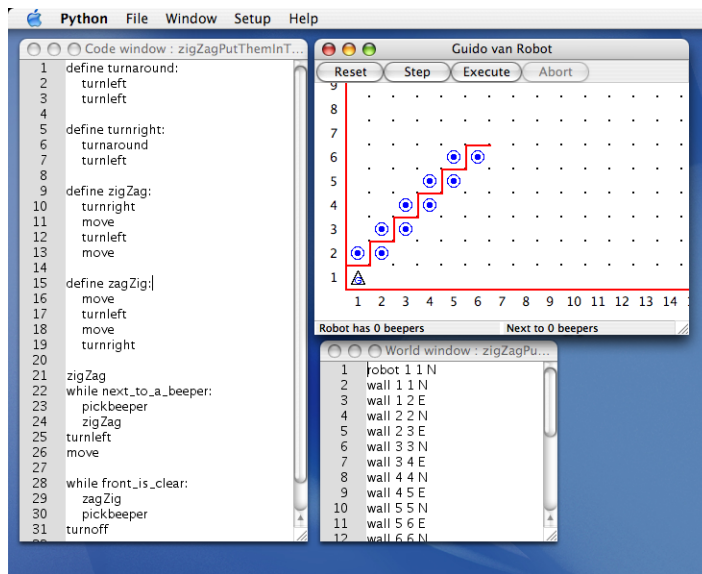
The Language draws its inspiration from an early variant of BASIC but is based on the modern .Net Framework Platform. The Environment is simple but rich in features, offering beginners several of the benefits that professional programmers have come to expect of a

worthy IDE. A rich set of Libraries help beginners learn by writing compelling and interesting programs. Small Basic is intended for beginners that want to learn programming. In internal trials Small Basic has had success with kids between the ages of 10 and 16. However, it's not limited to just kids; even adults that had an inclination to programming have found Small Basic very helpful in taking that first step.



PICTURE 2. A screenshot of a Small Basic IDE

Guido van Robot (GvR) is an educational tool (Guido van Robot webpage 2014) to help students learn the Python programming language, named after the creator of Python, Guido van Rossum. GvR uses the idea behind Karel the Robot, making the learning of Python programming more interesting. Using GvR, a student writes a program that controls a 'robot' that moves through a city consisting of a rectangular grid of streets (left-right) and avenues (up-down). Guido van Robot uses a minimalistic programming language providing just enough syntax to help students learn the concepts of sequencing, conditional branching, looping and procedural abstraction. It permits this learning in an environment that combines opportunities for problem-solving with instant visual feedback. In short, it is an interactive, introductory programming language that focuses on learning the basic concepts of programming, applicable in any high-level language.



PICTURE 3. A screenshot of GvR

2.3 Graphical based languages

2.3.1 Graphical programming

Graphical or visual programming involves the task of creating a structured program through the combination of graphical elements on a program canvas or timeline. Through intuitive graphical representations of programming concepts a user can piece together some kind of programming puzzle on screen and thereafter observe its execution on screen. Little or no typing of commands is required to complete the above task. The creation of graphical programming tools for children is a wide field of research since the early 1960's. Based mainly on graphical interfaces and utilizing the theories of constructivism by Papert (Papert, 1980), a large number of programming languages was created for both children and novice users. These graphical based programming approaches, incorporated simple syntax, nested loops, and control structures through graphical representation, allowing children to program by dragging and connecting icons on computer screens.

2.3.1 Graphical based programming languages

Scratch is a free desktop and online multimedia authoring tool (Scratch webpage 2014) that can be used by students, scholars, teachers, and parents to easily create games and provide a stepping stone to the more advanced world of computer programming or even be used for a range of educational and entertainment constructivist purposes from math and science projects, including simulations and visualizations of experiments, recording lectures with animated presentations, to social sciences animated stories, and interactive art and music. Viewing the existing projects available on the Scratch website, or modifying and testing any modification without saving it requires no online registration. Scratch allows users to use event driven programming with multiple active objects called "sprites". Sprites can be drawn — as either vector or bitmap graphics — from scratch in a simple editor that is part of the Scratch, or can be imported from external sources, including webcam.

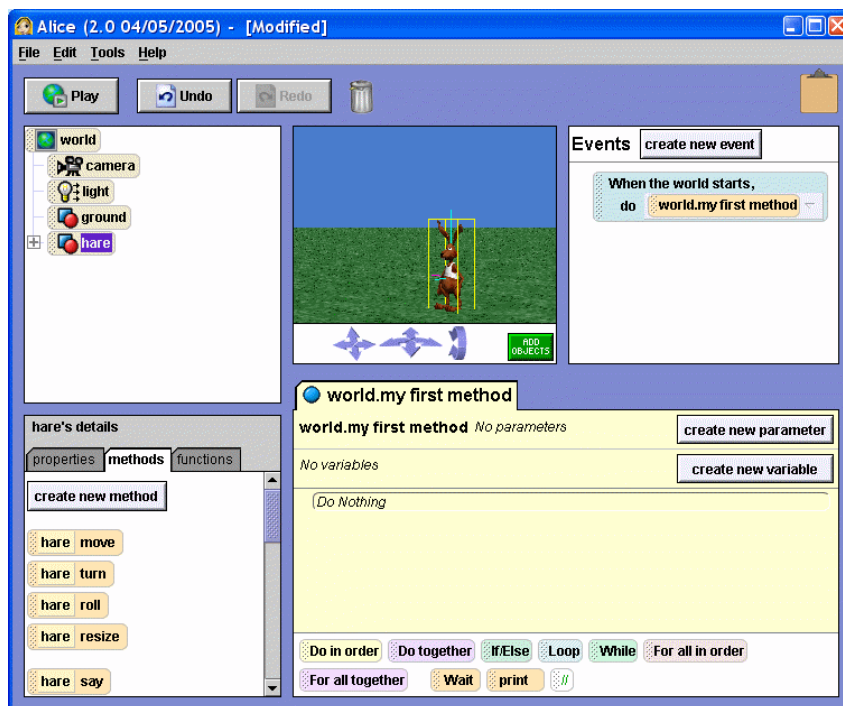


PICTURE 4. Scratch editor screenshot

Alice is an innovative 3D programming environment (Alice webpage 2014) that makes it easy to create an animation for telling a story, playing an interactive game, or a video to

share on the web. Alice is a freely available teaching tool designed to be a student's first exposure to object-oriented programming. It allows students to learn fundamental programming concepts in the context of creating animated movies and simple video games. In Alice, 3-D objects (e.g., people, animals, and vehicles) populate a virtual world and students create a program to animate the objects.

In Alice's interactive interface, students drag and drop graphic tiles to create a program, where the instructions correspond to standard statements in a production oriented programming language, such as Java, C++, and C#. Alice allows students to immediately see how their animation programs run, enabling them to easily understand the relationship between the programming statements and the behavior of objects in their animation. By manipulating the objects in their virtual world, students gain experience with all the programming constructs typically taught in an introductory programming course.



PICTURE 5. Alice programming environment screenshot

Kodu is a visual programming tool (Kodu game lab webpage 2014) which builds on ideas begun with Logo in the 1960s and other current projects such as AgentSheets, Squeak and Alice. It is designed to be accessible by children and enjoyable by anyone. Kodu is

available to download as an Xbox 360 Indie Game. There is also a PC version in an open beta which is available to anyone at their website. Kodu is different from those other projects in several key ways:

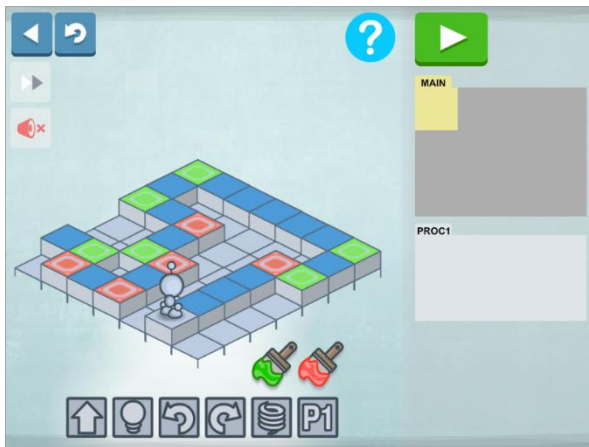
- It avoids typing code by having users construct programs using visual elements via a game controller
- Rather than a bitmapped or 2D display, programs are executed in a 3D simulation environment, similar to Alice

Kodu Game Lab has also been used as an educational learning tool in selected schools and learning centers.



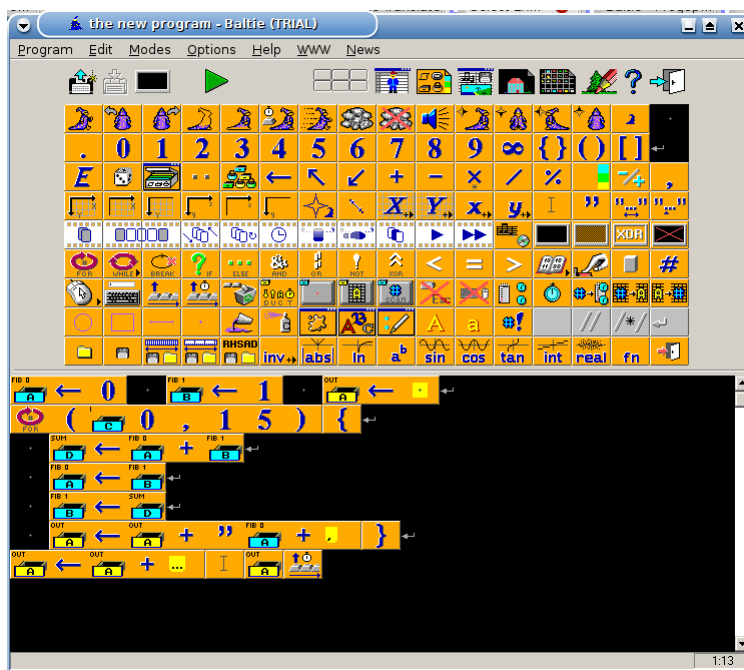
PICTURE 6. Kodu Game Lab programming tool screenshot

Lightbot is a visual programming game designed to teach basic instruction sequencing, procedures, recursive loops, and conditionals (Lightbot webpage 2014). In Lightbot, players guide a robot to light up blue tiles to solve levels, utilizing a small set of symbols representing actions and procedure calls.



PICTURE 7. Lightbot programming game screenshot

Baltie is an educational graphic oriented programming tool (Baltie webpage 2014) for children, youth (and adults). Baltie is also main character of this software a little wizard keen to execute miscellaneous commands and to conjure pictures (tiles) in his scene. With Baltie's help children will quickly realize what a computer is and how to master and program the computer. All that by playing. Baltie can be used also for exercising logical thinking. It makes no demands on child's knowledge, only playfulness and imagination are required. It is used in many countries in the basic schools. The new version of Baltie 4 fully supports C#.



PICTURE 8. Baltie educational programming tool screenshot

2.4 Tangible based languages

2.4.1 Tangible programming

Recent research in tangible user interfaces, as they were defined by Ishii and Ullmer, (Ishii, H., & Ullmer, B. 1997), created excellent opportunities for the pioneering implementation of technology inside school classes (Ichida et al., 2004; Itoh et al., 2004). An area that seems to have benefitted by this kind of technology is that of tangible programming environments, with application mainly to education, but not exclusively to it. (McNerney, 2004; Blackwell, 2003). A tangible programming environment may have the same usage results, with a text-based or graphical-based programming language. The peculiarity of tangible environments has to do with instead of using graphical on-screen objects or selecting on-screen commands, real-world objects are being utilized to fulfill the programming process.

Programming in general seems to be a demanding task for novices of all ages (Kelleher & Pausch 2005). Users not only find it difficult to comprehend a cumbersome syntax with awkward-sounding commands, but in addition how to master the programming environment itself (Cockburn & Bryant, 1997). Lowering the learning threshold of a programming environment, is considered as one big advantage of tangible user interfaces. Given the fact that users no longer need to learn how to use a mouse or keyboard, they only need to have the natural ability to operate real every-day objects like cubes or puzzles (Smith, 2007). As a result, it is estimated that tangible systems reduce the knowledge burden of a person having to master a programming environment and thus its attention is being focused at the task of learning how to program (Marshall, 2007).

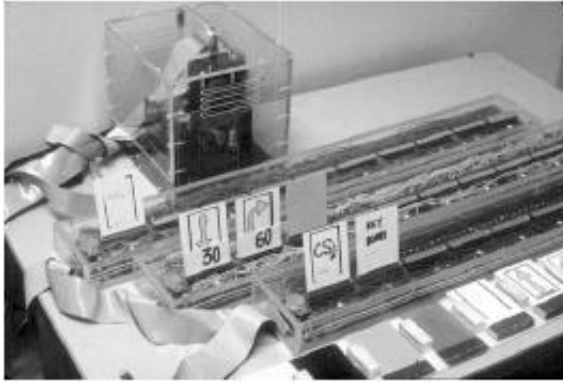
Various systems that have influenced the creation of tangible user interfaces are the following: (a) AlgoBlocks (Sapounidis, T., & Demetriadis, S. (2009) & Kato, 1993), the first system to introduce programming commands in the form of cubes, (b) Tangible Programming Brick (McNerney, 2001), the first system that incorporated parameters alongside commands, (c) Electronic Blocks (Wyeth & Purchase, 2000), that allowed

children to construct robots or simple programmable mechanisms by connecting tangible programming elements together, (d) Tern (Horn & Jacob, 2007), which was the first to introduce scanning and recognition systems in order to translate the commands issued by the user, into a programming sequence.

2.4.2 Tangible programming systems

Radia Perlman, researcher at M.I.T. media lab, at the late 70's, understood that most children under 11 to 14 years old, were not ready to start programming in the traditional way, e.g. by entering Logo commands in a PC with a keyboard (Kelleher & Pausch, 2005). One of the biggest problems the children faced when it came to programming, was not only writing the code but the user interface also. Perlman then started to design some interfaces that would allow even pre-elementary education children to learn to program a turtle. Those efforts matured to the first interface of this kind Tortis - Slot machine (Kelleher & Pausch, 2005). From that day and onwards, different design approaches to tangible programming followed and they will be presented in brief below.

The Slot machine (Kelleher & Pausch, 2005; McNeerney, 2004), was the first system fabricated that introduced plastic cards that could be inserted in three differently-colored stacks. At the left extreme of each stack one could see a "Do it" button. When someone pushed the button, a virtual turtle executed the command that was printed on the card's label. Furthermore upon execution of the command, a light was turned on below the corresponding card. The Slot machine offered among others, some important functions as in the direct manipulation of the executing program, by adding, rearranging, or even removing cards. Furthermore Perlman introduced later on, special cards that provided procedure call capabilities.



PICTURE 9. The Tortis slot machine

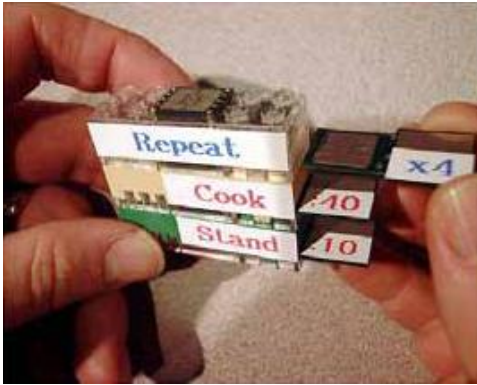
AlgoBlock (Sapounidis, T., & Demetriadis, S. (2009) & Kato, 1993) is a tangible educational programming language for elementary education and high school students. The system consists of a collection of cubes that can be connected to form a program. The cubes are then connected to a PC for the program to be executed. Each cube corresponds to a command that is similar to the Logo instructions. AlgoBlock was constructed to promote collaboration. They act as a tool for collaborative activities and help trainees to build programs through social interactions and discussion. The trainees have the task of steering a submarine depicted on screen, by utilizing Logo-like statements. The statements available include, move forward, turn left/right, return etc. Each of the above command-statement has a corresponding cube.



PICTURE 10. Algo block

Research by Timothy S. McNerney (McNerney, 2001) on tangible user interfaces, began from 2000 with the construction of the Tangible Programming Brick, which was another tangible programming language. The researchers decided to build a one dimensional system

in which they stacked Lego bricks and thus creating a sequence of commands. To make the system even more powerful they equipped the bricks with a slot on the side. In this slot one could insert a electronic card to act as a parameter for the command, but along the way it became evident that someone could insert other things like switches, sensors etc.



PICTURE 11. Tangible programming brick (McNerney, 2001)

Electronic blocks (Wyeth & Purchase, 2000) are designed in such a way that children can connect them like any ordinary cube. By placing the electronic blocks one on top of the other, children build programs that perform different tasks. Each electronic block has an input and an output and when connected, the output of one cube controls the input of the other. There are three kind of blocks.

- Sensor Blocks
- Action Blocks
- Logic Blocks

Logic blocks detect light, sound and touch in the surrounding environment. Sensor blocks are those that signal Action blocks to perform some operation. Logic blocks have an in-between role. By placing them amid sensor and action blocks, we have the capability of altering the anticipated command. Even very young children, can use a collection of electronic blocks. A simple game with action blocks can produce some sort of events that the children can find interesting and engaging.



PICTURE 12. Electronic blocks (Wyeth & Purchase, 2000)

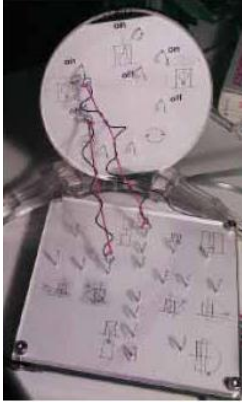
AutoHAN (Blackwell & Hague, 2001) is a networked programming architecture that allows programming between different media devices in a house. A part of the AutoHan architecture are the media Cubes. It is a simple programming language suitable for users who can operate a simple VCR remote controller. Someone can argue that programming a VCR is far less challenging than from programming in a PC. But if we want to issue a command to record video from the front door security camera for 5 minutes from the moment the motion detector is triggered, then we can state that programming is involved. Each cube of the system represents a function of a home media appliance. If for example the cube depicts play/pause, the user can associate this cube with the CD player. The combination of more than one cube that represent different media functions, is the process needed to complete the creation of the program.



PICTURE 13. Auto-Han media cubes (Blackwell & Hague, 2001)

Tangible programming with strings is a device that was created in order to construct simple programs that control toy robots. (Patten, Griffith, & Ishii, 2000). For the users to create a program in this system, events must be associated with actions which in turn shall be executed as a response to those events. These associations are presented as images that are connected with a string (actually it is a wire twisted like a string). The user associates

events that are created by the robot's sensors, with actions that are executed as a response to those stimulations. Thus the system creates a program that can be loaded on the robot's memory, and to function according to the way the strings were connected.



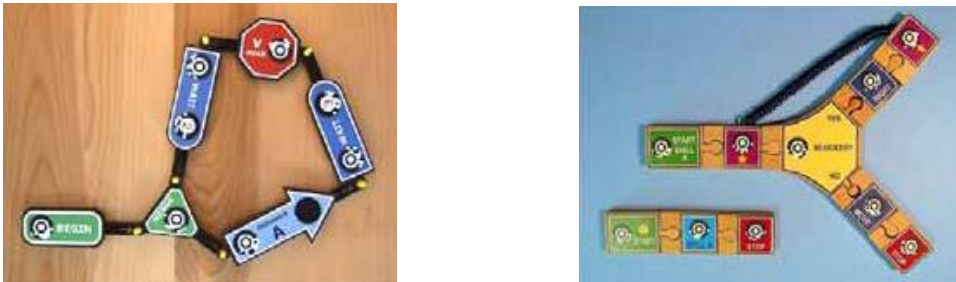
PICTURE 14. Programming with strings (Patten, Griffith, & Ishii, 2000)

The Game Blocks system (Smith, 2007) is comprised of large cubes which are placed on rails and thus creating a sequence of commands. The relative position of the cubes is important, since it expresses a logic sequence, which in turn is a program. The system includes in total 6 commands for the control of a humanoid robot. The available commands are: forward, back, body left, body right, head left, head right. An interesting feature of the system is that it is not necessary to incorporate electronic circuits inside each cube, since the task of recognizing the cubes is carried out by electronic circuits on the rails. Furthermore, it is worth pointing out that the system need not be connected to a PC, since the computational power is being provided by microcomputers integrated inside the circuits.



PICTURE 15. Gameblocks (Smith, 2007)

Quetzal & Tern (Horn & Jacob, 2007), are yet two more tangible programming languages with educational orientation. These languages use solid objects without electronic circuits and electrical power requirements. The programming elements of each language are identical and resemble puzzle pieces. Quetzal is a language used to control a Lego Mindstrom robot, while Tern is used to control a virtual robot in a PC screen. The philosophy of both languages is common, since they were built by the same people, in the same time period. Students using these languages program in offline mode and exploit a portable scanning system to recognize the commands. This scanning system scans the puzzle pieces and recognizes which commands exist and how they are connected, to form a program. Finally, it is noteworthy to state that both languages allow the user to enter parameters at the commands.



PICTURE 16. Quetzal Tern (left-right) (Horn & Jacob, 2007)

The T_ProRob system by (Sapounidis, T., & Demetriadis, S. 2011) consists of 28 cubic commands and 16 smaller cubic parameters. The users of this system can order the cubic commands and program the Lego Mindstrom (NXT) robot to run the sequence of commands that have been formed by the cubes pushing just one button. The set of T_ProRob parameters are smaller cubes which are connected to the commands and changing their operation. The user connects on the basis (Master Box) the commands in order to form the program. Then by pushing the run button, which is on the master-box, the communication between the blocks and the master-box starts in order to have a successful reading of the program. The next task which is undertaken by the master-box is to communicate with a remote computer using Bluetooth or RS 232. This computer records in a Database information about the commands that have been used and also statistical data concerning the program which was created by the user. Once the computer finishes the recording, it sends the program to a NXT robot using Bluetooth so as to run it.

V_ProRob subsystem by Sapounidis, T., & Demetriadis, S. N. (2012) has the same commands -parameters with those offered by the tangible T_ProRob. The subsystem is a graphical isomorphic equivalent of T_ProRob. It accumulates the specific features and the capabilities of T_ProRob. For instance, V_ProRob informs the users about tests and errors on the icons of the commands and parameters the same way that has been done with T_ProRob. It offers to the users a reliable alternative to program the Lego Mindstrom robot with a simple and easy graphical environment via a mouse.

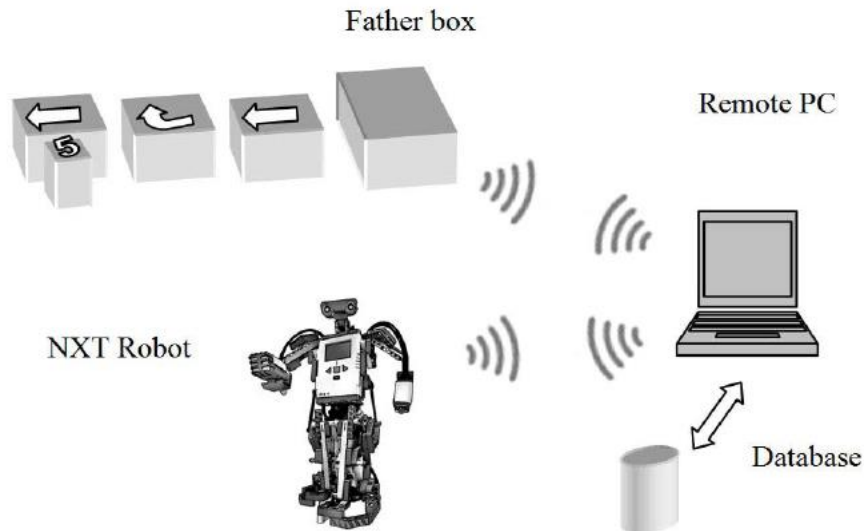
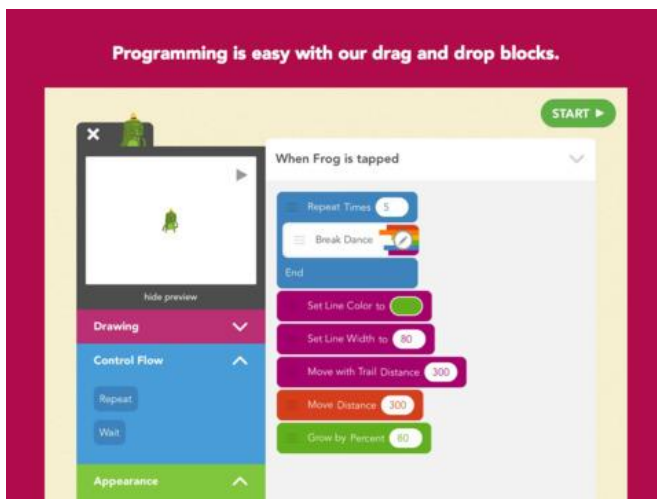


FIGURE 1. T_ProRob system (Sapounidis, T., & Demetriadis, S. 2011)

2.5 Mobile based systems

The mobile revolution has affected many parts of our everyday lives and has also revolutionized education. Mobile devices have been transformed into powerful learning tools and technology will play a big part in the future of the classroom. As mobile learning programs become more ubiquitous, a lot of attention has been given into all the possibilities of integrating mobile devices into formal schooling tools. Traditional schools and universities are trying to leverage the enormous opportunities for innovation in this area and they are investing in tablets for both their students and staff. Technology has spread in many devices like smart-phones and tablets are now full featured programmable apparatuses. Thousands of apps have been designed specifically for education.

Hopscotch is a visual introduction to programming for kids ages 8-12. It allows kids to drag and drop colorful blocks of code to create their own programs (Hopscotch webpage 2014). Children can select preset characters or create text objects and manipulate them by dragging-and-dropping method blocks. For example, you can move an object by a set amount on the X-Y axis, change the scale, or repeat actions. Hopscotch is available on the iPhone and the iPad.



PICTURE 17. Hopscotch screenshot

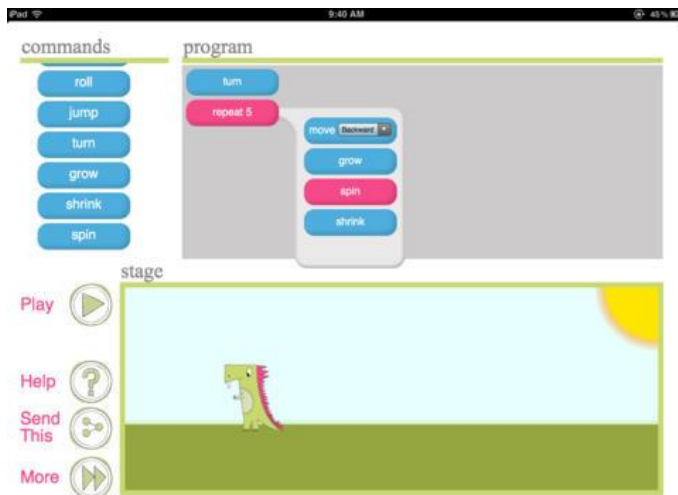
ScratchJr is an introductory programming language that enables young children (ages 5-7) to create their own interactive stories and games (Scratch junior webpage 2014). Children

snap together graphical programming blocks to make characters move, jump, dance, and sing. Children can modify characters in the paint editor, add their own voices and sounds, even insert photos of themselves and then use the programming blocks to make their characters come to life. ScratchJr was inspired by the popular Scratch programming language, used by millions of young people (ages 8 and up) around the world.



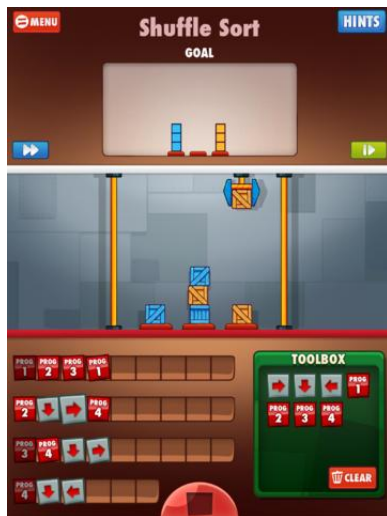
PICTURE 18. ScratchJr screenshot

Daisy the Dinosaur by Hopscotch Technologies (Daisy webpage 2014) introduces children to basic computer programming. A challenge mode tutorial shows how to make the dinosaur move, jump, shrink and grow using drag and drop instructions. Without explicitly using the terms, it demonstrates looping and conditional programming.



PICTURE 19. Daisy the Dinosaur screenshot

Through Cargo-Bot children (ages 6-12) write programs to control a robotic arm (Cargo-Bot webpage 2014). The game asks students to program an automated cargo crane to pick and drop colored boxes in a particular pattern in particular places. It gives kids hands-on experience with computer science concepts like logic and problem solving. Kids will practice tackling a big problem by breaking it down into smaller problems to solve.



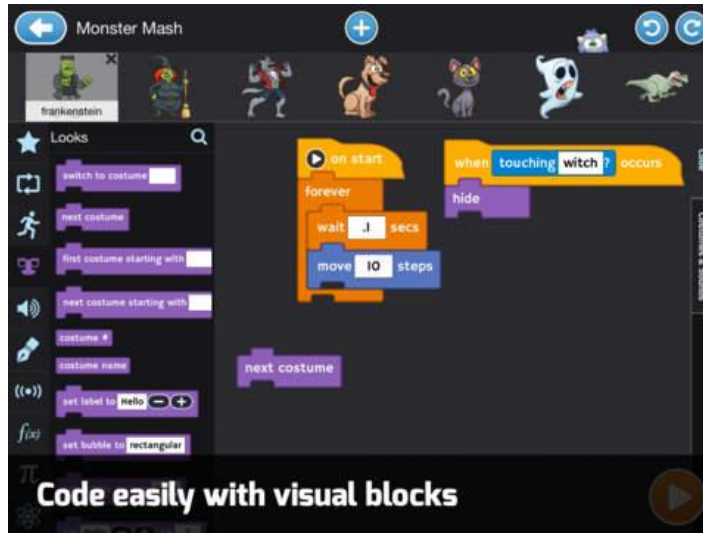
PICTURE 20. Cargo-Bot screenshot

Move the Turtle (ages 6-12) is an iPad app for teaching basic computer programming to young children (Move the turtle webpage 2014). Kids find the game's goal—to move a turtle around the screen using programming instructions. Kids learn how to build their programs using the command tiles on the chalkboard. Commands include Move, Turn, Pen, Color, Repeat, Sound, Position, and Conditions. You can reorder the commands and see how the program changes.



PICTURE 21. Move the Turtle screenshot

Tynker (ages 9-12) iPad app is based on collections of puzzles, solved by stringing together commands in sequences using a drag and drop interface (Tynker webpage 2014). Introduces concepts like sequencing, repetition and conditional logic. Reinforces basic geometry concepts while using programming to draw angles and lines. Tynker is inspired by visual programming languages such as Scratch from MIT, Alice from CMU, and other programming languages like Logo, SmallTalk, and Squeak.



PICTURE 22. Tynker screenshot

3 STUDIES

3.1 Comparisons between the different interfaces

Although a big part of tangible systems has to deal with systems aimed at children, only a limited number of researches have utilized children and tried to compare graphical and tangible user interfaces. Specifically in the field of puzzle solving (Xie, L., Antle, A. N., & Motamedi, N. 2008), presented the results of a comparative research that included physical, tangible and graphical interfaces to solve the puzzle. The children were occupied with puzzle solving, using all three interfaces (physical, tangible, graphical). Children responses regarding the appeal of each interface, showed no difference among the three interfaces. As far as commitment is concerned, it was noted that in the case of the physical and tangible interfaces, a larger number of participants wished to solve another puzzle. Furthermore, Antle et al. (Antle, A. N., Droumeva, M., & Ha, D. 2009) using the same system, investigated the hypothesis that immediate physical interaction favors users in the case they have to deal with spatial problems. The results demonstrated that children were faster and more effective at puzzle solving and that can be attributed to the different actions and strategies the children adapted using the tangible system.

In the field of mathematics Manches et al. (Manches, A., O'Malley, C., & Benford, S. 2010), compared physical and virtual materials for solving arithmetic problems, to show how the limitations of the various interfaces, can influence users' actions. This particular research presented that the properties of an interface are important for finding possible solutions in arithmetic problems. On the contrary, Olkun (2003) did not find any difference between the graphical and physical interfaces when the task was to solve two-dimensional geometrical problems.

3.2 Comparisons in programming

Although a limited number of studies tried to compare tangible with isomorphic graphical systems at various fields of knowledge, contradictory results that occurred point-out in many occasions, the need to examine the circumstances under which these kind of interfaces offer more advantages at the environment of a real classroom. (Zuckerman & Gal-Oz, 2013). Specifically in the field of programming exist scarcely few such studies (Orit & Eva, 2009). These comparative studies deal with tangible and graphical systems, which have analogous characteristics and the focus of research are ease of use, enjoyment etc. In more detail Kwon et al. (2012) performed a study comparing the Algorithmic Bricks with Scratch system (Maloney 2010), but in that case the systems were not isomorphic.

A noteworthy study in tangible programming involves the work from Horn et al. (Horn et al., 2009) which compared a passive and a graphic programming language in the non-controllable environment of a museum. (Boston museum of Science). The research revealed advantages of the tangible programming language versus the graphic one. Specifically the passive tangible programming language was more attractive and more efficient for the users to get actively involved with. Furthermore this active involvement seemed to be more evident with girls. In parallel Horn et al. performed a study at a nursery school with children aged from 5 to 6. (Horn, Crouser, & Bers, 2012). By applying qualitative analysis, they came to the conclusion that if children were given appropriate technologies, they could better understand specific concepts from the fields of programming and robotics.

Even though it is estimated that tangible interfaces are more efficient than graphical ones, only a limited number of studies has addressed the issue of knowledge and social benefits from tangible interfaces in comparison to graphical interfaces. (Xie et al., 2008). In more detail, the impact of tangible interfaces and the circumstances under which tangible objects can become more effective for children to use in various fields such as programming, have not been extensively studied and remain unexplored. (Marshall, 2007; Kelleher & Pausch, 2005). Finally a recent study by (Sapounidis, T., & Demetriadis, S. & Stamelos I. 2014), carried out and presented a comparison study of children's performance using the two

isomorphic subsystems (tangible versus graphical). Data analysis upon task measurements showed that younger children needed less time to accomplish the programming tasks when using the tangible interface. On the contrary, elder children, who were more experienced computer users, needed almost the same time to accomplish the tasks with both interfaces. Furthermore, fewer programming errors occurred and better debugging was achieved in the tangible case.

3.3 Summary of limitations regarding tangible programming tools

Although there have been many attempts at constructing tangible systems, a lack of tangible programming tools is evident (Kwon et al., 2012). In more detail the limitations that seem to exist are the following: (a) several systems do not have a satisfactory number of commands and parameters (Cockburn & Bryant, 1997) and that seems to limit assimilation of programming concepts, (b) the lack of real time control hinders the smooth interaction of the programmer and the program itself (Gallardo, Julia, & Jorda, 2008), (c) some systems require special surfaces or rails and thus are difficult to relocate, making them difficult to use at real school classes, (d) some physical properties like shape, temperature etc. can offer advantages at tangible systems for programming, nevertheless such qualities have not been incorporated in existing systems, (Zuckerman et al., 2005; Manches et al., 2010; Xie et al., 2008), (e) and concepts such as storage and code reusability, are not supported by any system. It is clear that while tangibles appear more efficient than graphical user interfaces, more research is required to elucidate the circumstances under which the advantages are demonstrated in different domains. (Sapounidis, T., & Demetriadis, S. 2013).

Our effort will try to bridge the world of graphical and tangible programming by offering a mobile graphical isomorphic equivalent for tablets of an existing tangible programming language (T_ProRob, section 2.4.2). The tangible part of the system will mainly focus on the ability of the tablet screen to offer drag and drop capabilities very similar in nature, to having a physical object at hand. The graphical part of our system is of course the various icons that comprise the GUI (Graphical User Interface). The third factor-mobility- is

greatly enhanced by the combination of a tablet and a Lego EV3 Mindstorms™ robot connected together wirelessly via Bluetooth, making it an ideal learning aid inside any classroom or home.

4 DESIGN AND DEVELOPMENT

4.1 Interaction design for kids

Interaction design is about shaping digital things for people's use. It is about helping people to choose specific goals, through an interface. This could be withdrawing money from an ATM, taking a picture with a phone, or checking our emails. This is different from industrial design where the goal is to solve a problem by crafting a physical product that would be mass-produced, taking into account material and production line constraints. It's also different from graphic design, which is more meant to be looked at.

UX (User experience): deals with the overall experience associated with the use of a product or a service. It requires a good understanding of the user, and of the system of the product the user interacts with. UI (User Interface) is the specific interface of the product. It is the tool, the point of interaction between a human, and a system. Therefore, UI is a part of UX. Similar to designing for adults, designing for kids requires a strong understanding of what users need and want. But designing for kids differentiates from designing for adults. Young children except for the end goal they have in their mind while using the interface, also see the use of a tablet or game as a part of an adventure. Kids delight in challenge and conflict, regardless of their goals.

This generation of kids is digitally native, meaning that technology has been and always will be a part of their lives. As our target group starts from children up to 4 years old we need to be aware of the unique characteristics of this specific age. These are elements of programming that pre-reader children are capable to support already such as sequence, concept of code, cause and effect, counting, planning and problem solving. Kids are more sophisticated than they may appear initially and they're able to mentally categorize quite efficiently. (Design for Kids Digital Products for Playing and Learning 2014).

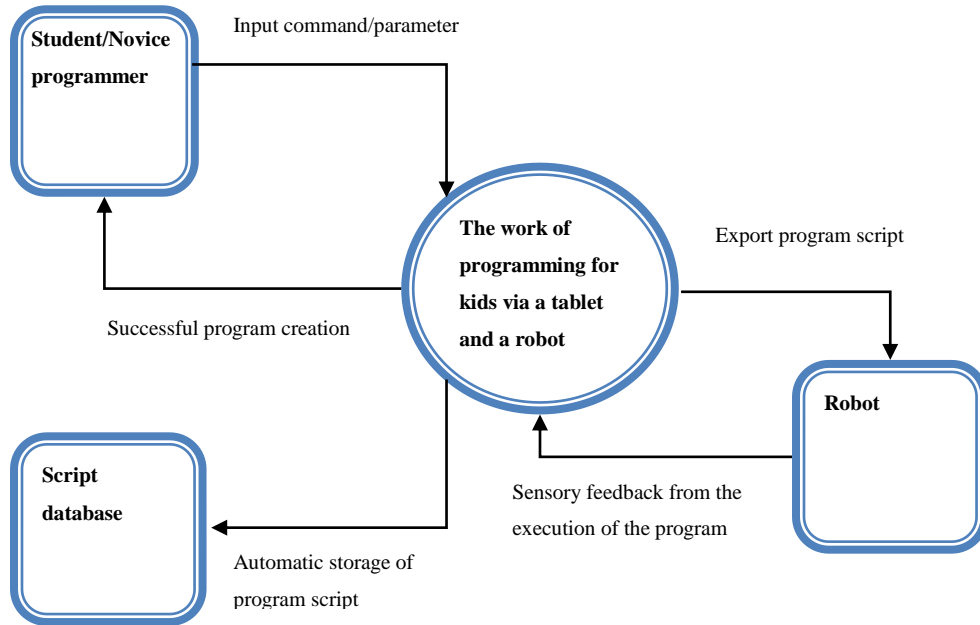








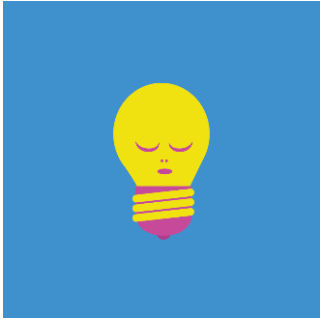
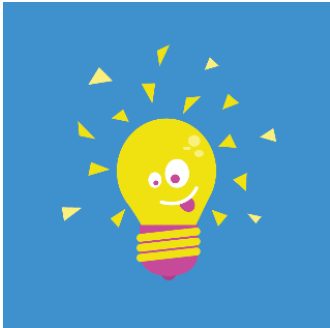
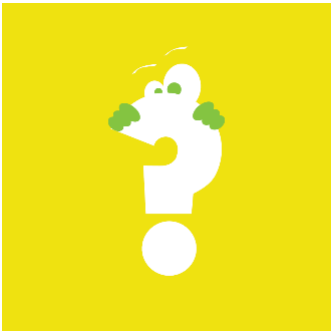
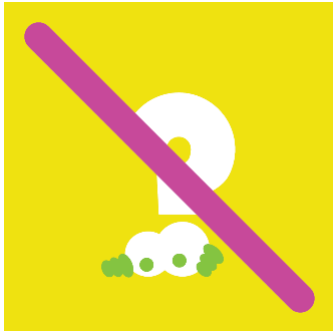
FIGURE 2. Workflow of our system

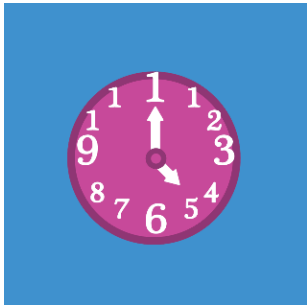

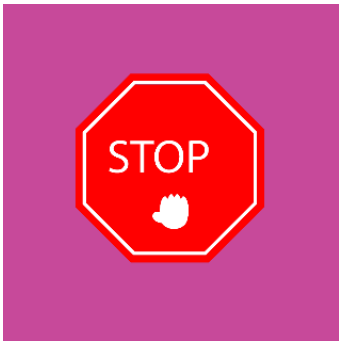
4.2 Icon Design





Colors should act as guides identifying all the interactions and specific contents. There should be a limited set of bright, bold colors, not too many colors that could overwhelm kids and make them lose interest. Navigation should be as simple as possible. Symbols should be basic shapes that mimic everyday items that children are familiar with (Design for Kids Digital Products for Playing and Learning 2014). Below follows a display of the various command and parameter icons present at our application.

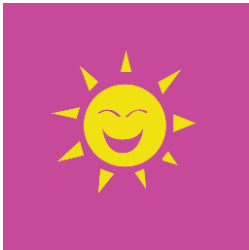
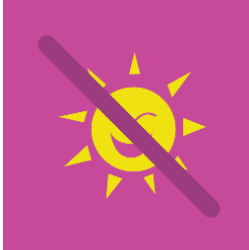

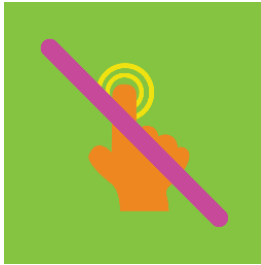


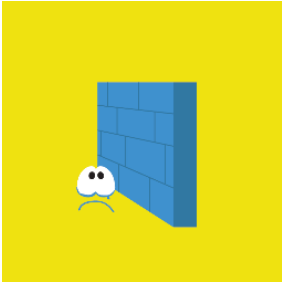
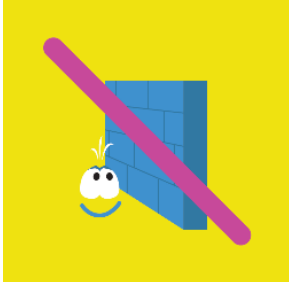
TABLE 1. Command and parameter icons

	<p>Turn Right (Turn right 90 degrees)</p>		<p>Turn Left (turn left 90 degrees)</p>
---	--	--	--

	<p>Move forward</p>		<p>Move backward</p>
	<p>FOR loop</p>		<p>End of FOR loop</p>
	<p>Turn light off</p>		<p>Turn light on</p>
	<p>If condition (start)</p>		<p>End of if statement body</p>

	<p>Delay for 5 seconds</p>		<p>Make a sound</p>
	<p>Forcibly terminate program execution</p>		

		<p>Numbered parameters (can be coupled with for loop commands and move/turn commands)</p>
		

	<p>Light sensor (Presence of light)</p>		<p>Light sensor (Absence of light)</p>
	<p>Touch sensor (Obstacle present)</p>		<p>Touch sensor (Obstacle not present)</p>
	<p>Sound sensor (Ambient sound)</p>		<p>Sound sensor (No Ambient sound)</p>
	<p>Ultrasonic sensor (Obstacle up ahead)</p>		<p>Ultrasonic sensor (Free space ahead)</p>

4.3 The Lego Robot

Lego robots that act upon programming, have already reached many schools. The Lego Mindstorms™ kits contain software and hardware to create customizable, programmable robots. In our case we have chosen the latest version of Mindstorms™ the EV3 which is the third generation Lego Mindstorms™ product released on September 2013 and fits in our case due to the advantage of connectivity with smart-devices as mobile phones and tablets. The EV3 programmable brick has 4 inputs (numbered using numbers from 1 to 4) and 4 outputs (numbered from A to D). The robot can power 4 motors and can gather information from the environment via various sensors that we will describe below.



PICTURE 23. Robots built using LEGO EV3 Mindstorms

4.3.1 Connection with the robot

The Lego EV3 robot can communicate through Bluetooth, USB (except for Windows Phone) or Wi-Fi connection. In order to send commands from a mobile device and control the Lego Mindstorms™ EV3 Robot, we send and receive messages to it using the LEGO MINDSTORMS EV3 API. We can connect, control and read sensor data from LEGO EV3 brick over Bluetooth, WiFi, or USB. LEGO MINDSTORMS EV3 API provides libraries

that are usable from the Windows desktop, Windows Phone 8, and WinRT (via .NET, WinJS and C++), along with full source code (Lego EV3 webpage 2014).

There are 3 types of commands that can be sent to the brick: `DirectCommand`, `BatchCommand` and `SystemCommand`. All of them are included inside the Brick object as *DirectCommand*, *BatchCommand* and *SystemCommand* properties, along with their corresponding methods. Also need to implement the *Icommunication* interface which will determine the way that library will connect to the brick. The library can only be used with a single brick at a time. Multiple brick communication is not supported yet (Lego EV3 webpage 2014).



PICTURE 24. LEGO EV3 brick

4.3.2 Motors and Sensors

Up to 4 motors can be hooked up to the ABCD ports on the EV3 brick. The EV3 Large Servo Motor is a powerful motor that uses tacho feedback for precise control to within one degree of accuracy. By using the built-in rotation sensor, the intelligent motor can be made

to align with other motors on the robot so that it can drive in a straight line at the same speed. The EV3 Medium Servo Motor is great for lower-load, higher speed applications and when faster response times and a smaller profile are needed in the robot's design. There are a variety of methods to interact with the motors which we can find in the API documentation. As an example, here is a *DirectCommand* which will turn the motor on Port A for 5 seconds at 50% power (Lego Mindstorms™ home page 2014):

```
await brick.DirectCommand.TurnMotorAtPowerAsync(OutputPort.A, 50, 5000);
```



PICTURE 25. The EV3 Large and Medium Servo Motors

Up to 4 sensors can be hooked up to the 1234 ports on the EV3 brick. There is number of sensors that come together with Lego EV3 kit. Infrared sensor detects proximity to the robot and reads signals emitted by the EV3 Infrared Beacon. Touch sensor detects when its front red button is pressed or released and has the capability to count single and multiple presses. Color sensor recognizes seven colors and also can detect the amount of reflected light and the intensity of ambient light. Ultrasonic sensor generates sound waves and reads their echoes to detect and measure distance from the objects. Gyro sensor measures the robot's rotational motion and changes in its orientation. Additionally, each motor also acts as a sensor and can return positional/rotational data. Each sensor/motor may also have the ability to return its data in a variety of different modes. As example, the Touch sensor can return whether the button is pressed, or it can return the number of times it has been pressed since it was last reset.

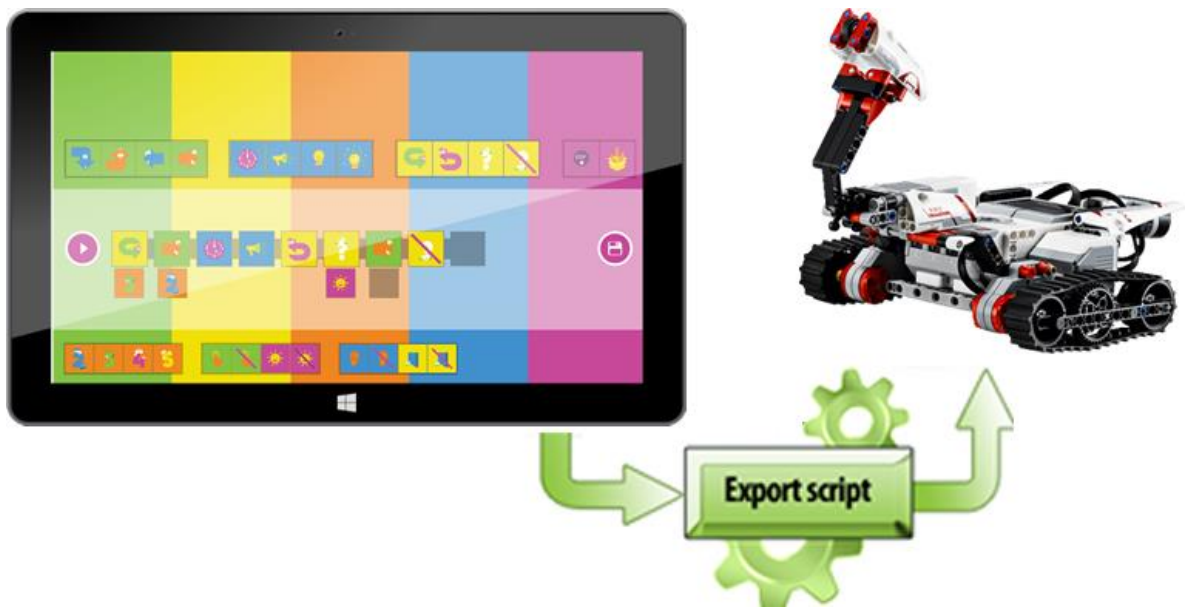


PICTURE 26. Sensors (Infrared – Touch – Color – Gyro – Ultrasonic)

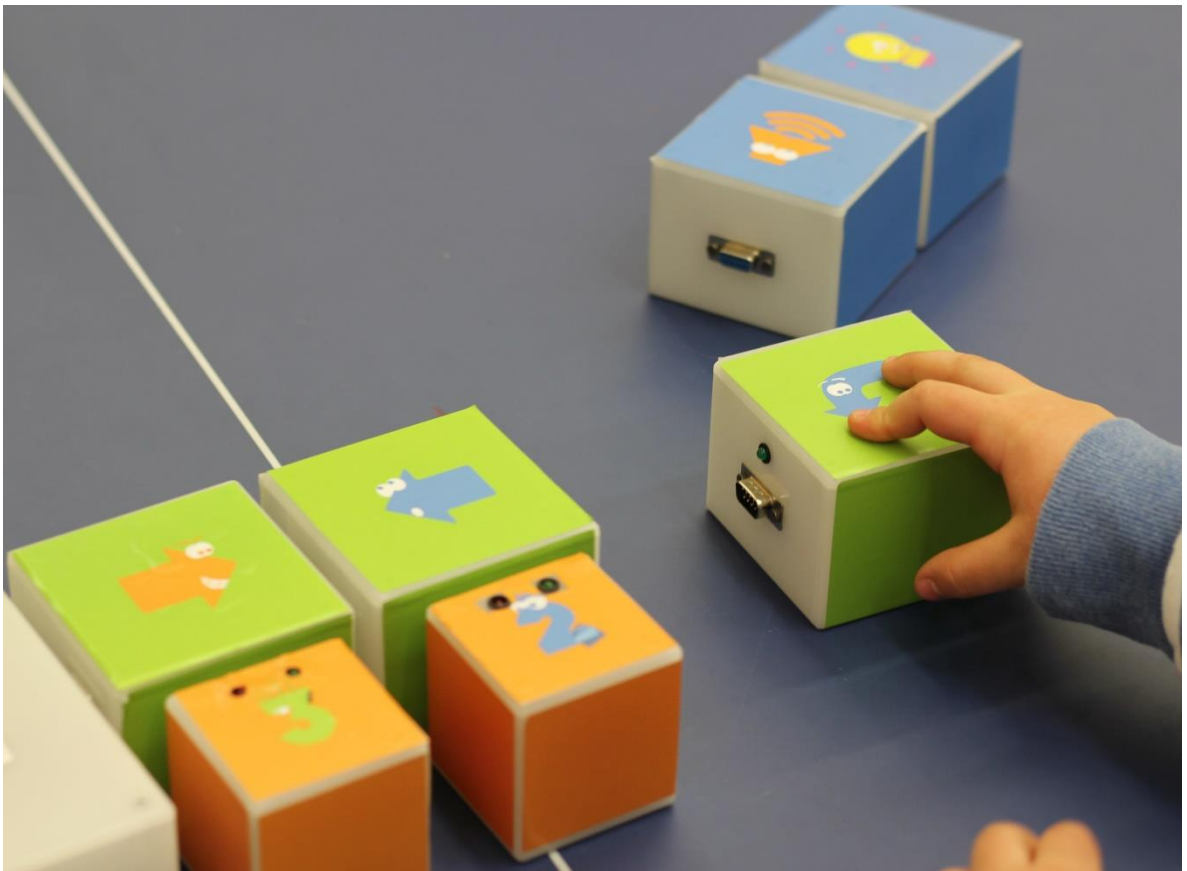
4.4 Mock ups



PICTURE 27. Mobile tablet system mock-up



PICTURE 28. Mobile ecosystem mock-up



PICTURE 29. Tangible programming system

4.5 Software Requirements

Software engineering is the process involving the creation of the software blueprint before the actual construction and release of a software product. It includes detailed specifications of the software's requirements both functional and non-functional. In order to provide a better understanding of the interaction our software has with the real world, business use cases (BUCs) are provided. Furthermore to define precisely all the interfaces between the product and other automated systems, organizations and users, product use cases (PUCs) are included. The above terminology is adopted from the Volere software requirements specification template (Volere requirements resources 2014). Also the goals of the project, the stakeholders of our product, a business data model and data dictionary are included which incorporate a specification of the essential subject matter, business objects, entities, and classes that are relevant to the product.

4.5.1 The Purpose of the Project

Programming has an enormous presence in everyday life of 21 century. New generation students are surrounded by computer technology and will possibly do in the future an occupation that hasn't been invented yet. Also the unprecedented growth rate of tablet computers or mobile devices in corporate and consumer markets is spreading steadily into schools. Future schools will most likely replace books with tablets. Researches show that students using tablets were *“more motivated, attentive and engaged”*.

A mobile application for tablets could introduce children to the basics of programming with an easy and interactive way. Children by playing with a friendly user interface will be able to understand better of what happens inside computers and also improve their math and logic skills. Using the tablet, children can develop their code by arranging different shapes-pieces-images-blocks that represent simple programming commands as part of a game. Additionally they could see the logic results and actions in reality as interaction with a *Lego Mindstorms™* robot. This will make them also more curious with the magical world of robotics.

4.5.2 Goals of the Project

We use Purpose, Advantage, Measurement (PAM) to structure our project's goal.

Purpose: one sentence to explain the organisation's reason for investing in the project.

Advantage: One sentence describing the benefit that the organization will realize if the project is successful.

Measurement: One sentence or a graph or diagram that quantifies how we will measure whether or not the benefit has been achieved.

TABLE 2. The Goals of our project

Goal #	Purpose	Advantage	Measure
1	We want to create a mobile application for tablets to introduce children to the basics of programming logic.	We want to be recognized as a leading software house for educational oriented software applications.	Graphs specifying numbers of software downloads for both the trial-period and full-featured versions of our software product.
2	We want to facilitate the understanding of fundamental programming principals through an intuitive and interactive manner.	We want to help children get accustomed to programming and elevate our corporate ethos profile.	Number of elementary schools enrolling in our "LEARN TO PROGRAM" campaign.
3	Transcend the programming experience from the boundaries of a tablet screen to the actual movement of	We want to forge a business partnership with <i>Lego</i> ®	Number of <i>Lego Mindstorms</i> ™ robot units sold, that include a free copy of our software.

	a <i>Lego Mindstorms</i> TM robot.		
4	We want to improve mathematical and algorithmic-problem solving skills-of children.	We want to provide the younger generations with the best educational tools available on the market.	Positive feedback from users of the software product.
5	We want to strive for the proliferation and establishment of our application as an invaluable and useful learning aid for children.	We want our firm to benefit from being popular amongst children.	Internet polls measuring user satisfaction. Social media references of our software.

4.5.3 The Stakeholders

Stakeholders form the basis on which our software product will operate and whose input is needed to build the product. The client has the final say on acceptance of the product, and thus must be satisfied with the product as delivered. We can think of the client as the person who makes the investment in the product. The person intended to buy/use the product is the customer. Below follows a table depicting the various stakeholders present at our product, along with their role in our product's culmination and operation. Furthermore we include for each stakeholder the degree of influence he/she has on our software product.

TABLE 3. Stakeholder definition for our software product

Stakeholder Class	Stakeholder Role	Stakeholder Rationale	Necessary Involvement	Stakeholder Influence
Interfacing Technology	Existing Hardware (<i>Lego</i>)	Necessary for our product to	Throughout the development	Big

	<i>Mindstorms</i> TM robot)	work	phase	
Interfacing Technology	Existing Hardware (Any tablet device)	Necessary for our product to work	Throughout the development phase	Big
Maintenance Operator	Software Maintainer	Keeps the software up-to- date	Throughout the lifetime-cycle of the product	Big
Operational support	Help Desk	To keep informed the customers of the product	Throughout the lifetime-cycle of the program	Medium
Client	Private investors- companies, Government ministries, Chief executive	To provide funding and market penetration for our product	Mainly throughout the development phase	Big
Core Team Members	Software Engineer	Responsible for putting all the software pieces together	Mainly throughout the development phase	Big
Core Team Members	Graphics Designer	Responsible for designing the icons of our application	Throughout the development phase and for future releases	Big
Core Team Members	Software Tester	Debugging the application	Throughout the development phase and for future releases	Big
Core Team Members	Programmers	Writing the actual code	Throughout the development	Big

			phase and for future releases	
Functional Beneficiary	Private corporations	Our product will bolster the sales of Lego Mindstorms™ robot	Throughout the lifetime-cycle of the program	Medium
Internal Consultant	Marketing specialist	To promote our product to the market	1~3 months before the official product release date	Big
Customer	Students, children, members of the public	To actually purchase the product	Throughout the lifetime-cycle of the product by providing feedback	Big

The hands-on users of the product are a list of a special type of stakeholder, the potential users of the product. For each category of user, we provide the following information:

- User name/category: Most likely the name of a user group, such as clerical users, schoolchildren, road engineers, or project managers.
- User role: Summarizes the users' responsibilities.
- Subject matter experience: Summarizes the users' knowledge of the subject matter/business. Rate as novice, journeyman, or master.
- Technological experience: Describes the users' experience with relevant technology. Rate as novice, journeyman, or master.
- Other user characteristics: Describe any characteristics of the users that have an effect on the requirements and eventual design of the product.

TABLE 4. The Hands-on users of our software product

User name	User role	Subject matter	Technological	Other user characteristics	User participation
-----------	-----------	----------------	---------------	----------------------------	--------------------

		experience	experience		
Elementary School Teachers	Demonstrator of our software	Novice and/or journeyman	Novice	Secondary user	Provide feedback for improvements in future releases
Pre-Elementary & Elementary School Students	First time user of our software and Lego Mindstorms™ robot	Novice	Novice	Age group 4~10, key user	Provide feedback for improvements in future releases
Parents	First time user of our software and Lego Mindstorms™ robot	Novice	Novice	Secondary user	Minimal
Owners of Lego Mindstorms™ robot	First time user of our software	Novice	Journeyman	Key user	Throughout the development phase by providing feedback
Beta testers	Development phase of our product	Expert	Expert	Experience in debugging software, key user	Throughout the development phase by providing feedback

Personas is a story about an invented person that will actually use our software product. By having one or more personas we can make the requirements specific to the people you are trying to satisfy. This is a particularly effective technique if we are specifying the requirements for a consumer product or a product that will be used by members of the public. Below follow two fictitious personas, the first being an elementary school student and the later being an elementary school teacher.

- Little 9 years old boy Mika Hakkinen, is thinking about what to ask Santa for Christmas. He doesn't want to be very optimistic for this year's present. He has heard that Lego is offering their Mindstorms™ robot with a special software that allows to control the robot via a tablet and make it perform lots of fun stuff! Also it would be a great chance for little Mika to get his hand on his father's iPad...
- Mrs. Helena is the tech-education teacher at Tampere's 2nd elementary school. She has heard about a new software company -Cubes Coding- that is launching a new promotional campaign called "LEARN TO PROGRAM". In this campaign they are offering a two month license for using their software for educational purposes. Also they are providing free of charge for the same time period, a Lego Mindstorms™ robot and a tablet, in order for pupils to experience programming to its fullest! Not to mention that now children will start to flock to the class by themselves and she will not have to yell so much...

4.5.4 Work Partitioning

A list showing all business events to which the work responds. Business events are happenings in the real world that affect the work. They also happen because it is time for the work to do something—for example, produce weekly reports, remind non-paying customers, check the status of a device, and so on. The response to each event is called a business use case (known as a BUC); it represents a discrete partition of work that contributes to the total functionality of the work (section 4.1, figure 2). The event list includes the following elements:

- Event name
- Input from adjacent systems
- Output to adjacent systems
- Brief summary of the business use case

TABLE 5. Business Event List

Event name	Input and Output	Summary
1. Student/ Novice programmer wants to write a program	New command and parameter(in)	The user inputs the command and parameters via the GUI of the application
2. Tablet wants to communicate with robot	Export script(out)	Upon completion of the program, the script is "fed" to the robot
3. Robot wants to communicate with tablet	Sensory feedback from the execution of the script (in)	When certain situations arise-mainly the fulfilment of a condition or not- , the robot communicates with the tablet and extra info is displayed
4. Automatic storage of program script	Program script output to script database(out)	Every script is ported to a database in a user friendly format.

BUC scenario for Business Event 1:

- A student starts writing a program.
- A command icon is selected and is placed on the right side of an existing one.
- A parameter icon is selected and is attached at the lower side of a command icon.
- The process is repeated until the end of the program

BUC scenario for Business Event 2:

- After the termination of the program.
- The tablet wants to communicate with robot via an already established Bluetooth connection.
- If the connection is still active the program script is transferred to the CPU of the robot.
- The robot starts executing the script.

BUC scenario for Business Event 3:

- While the robot is executing the script it is prompted to evaluate a condition that is related to the measurement of one of its sensors. Available sensors on-board the robot are light (present above a predefined brightness level or not), touch (come to an obstacle or not), ultrasonic (measure distance ahead from the robot) and sound (ambient sound present above a predefined threshold).
- After the sensor measurement is performed the outcome (true-false) is transferred back to the tablet and the user is informed of the outcome.
- The program continues with the next command.

BUC scenario for Business Event 4:

- Before a Student starts to write a program he/she is asked to enter a username.
- That username is used to automatically store the program script to the database in .csv format.
- If the student does not enter a username, the script is stored by using the current system time and date as its filename.

4.5.5 Data Model

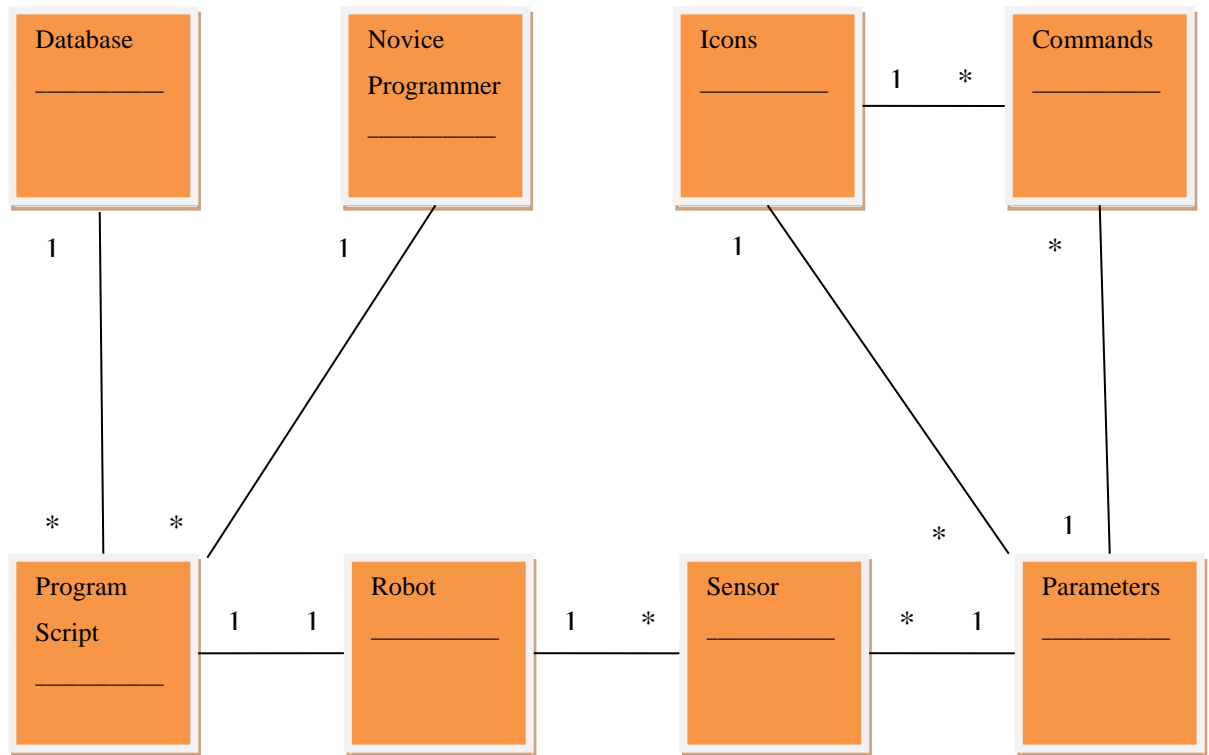


FIGURE 3. UML diagram of the data model, number (1) denotes has one, while the (*) denotes has many relationship.

TABLE 6. Data Dictionary

Data Name	Description	Definition	Data Type
Programmer	Novice Programmer/Student	Programmer's name	Class
Robot	Lego Mindstorms™ robot	Robot identifier	Class
Script	Program script	Script name + script creation date/time	Class
Database	Program script Database	Database name	Class
Icons	Images used for the GUI	Icon type + Icon colour + Error message	Class

Commands	Icons that execute a certain programming command	Command name + Max number of commands	Class
Parameters	Icons that attach to a command icon and alter it	Parameter name	Class
Sensor	A collection of different sensors (touch, light, ultrasonic, sound)	Sensor measurement	Class
Script Storage	Automatic script database storage	Script name + Script creation date/time + Programmer's name	Dataflow
Sensory feedback	Robot sensor measurement sent back to tablet	Sensor measurement + robot identifier	Dataflow
Update software	New software version	Software version number	Dataflow
Script name	Script name, also used for data storage		Attribute/Element
Robot identifier	Unique robot identifier		Attribute/Element
Sensor measurement	Various sensor readings		Attribute/Element
Command Name	Command type		Attribute/Element
Parameter Name	Parameter type		Attribute/Element
Theme Name	Theme Identifier		Attribute/Element
Script creation date/time	In HH:MM:SS and DD:MM:YYYY format		Attribute/Element
Error Message	In currently selected language		Attribute/Element

Programmer's Name	Programmer username		Attribute/Element
Server web address	ftp address		Attribute/Element
Icon type	Command or parameter		Attribute/Element
Icon colour	Colour of icon		Attribute/Element
Software version number	V1.1		Attribute/Element
Max number of commands	Up to 100		Attribute/Element
Database name	Default name		Attribute/Element

4.5.6 Product Boundary

A use case diagram identifies the boundaries between the users (actors) and the product. We arrive at the product boundary by inspecting each business use case and determining, in conjunction with the appropriate stakeholders, which part of the business use case should be automated (or satisfied by some sort of product) and what part should be done by the user or some other product. This task must take into account the abilities of the actors, the constraints, the goals of the project, and our knowledge of both the work and the technology that can make the best contribution to the work.

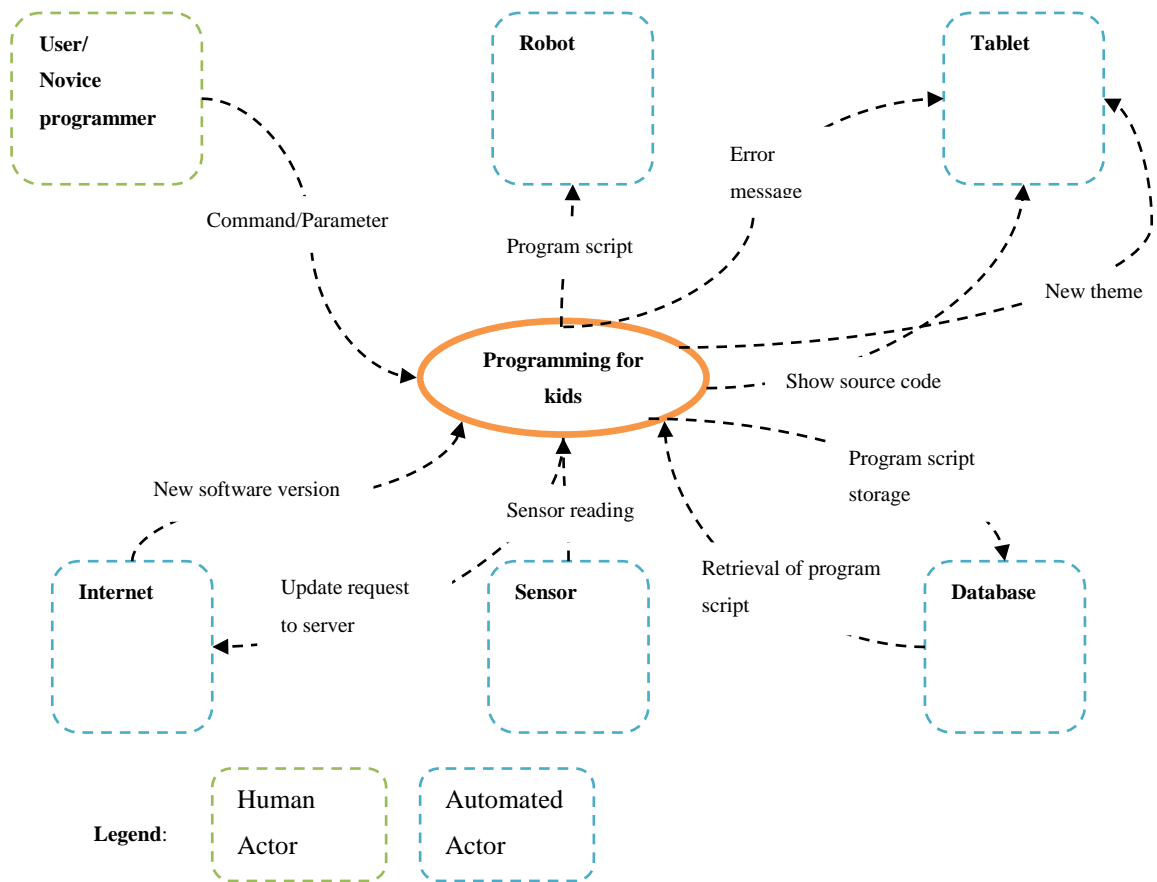


FIGURE 4. Product boundary diagram

TABLE 7. Product Use Case Table

PUC Number	PUC Name	Actor/s	Input & Output
1	User issues a parameter or command	User/programmer	Command/parameter (in)
2	Error message for inappropriate command / parameter	tablet	Error message (out)
3	Execution of script (script is transferred to robot via Bluetooth)	Robot/tablet	Program script (out)
4	Sensory feedback (measured from sensors on-board the	Sensor/robot	Sensor reading (in)

	robot/ultrasonic, touch, sound, light/		
5	Export script to database .csv format	database	Program script (out)
6	Update software via internet (to incorporate new Lego sensor e.g. heat (IR))	internet	Update request server (out) New software version (in)
7	Show source (high-level syntax representation)	tablet	Source (out)
8	Flashing icons (when a condition is met or not, green -red)	tablet	Flashing icon (out)
9	Look and feel button (bluish theme boys, pinkish girls)	tablet	New application theme (out)

PUC # 1

-User issues a command or parameter at the program timeline (commands on top,
parameters below of commands)

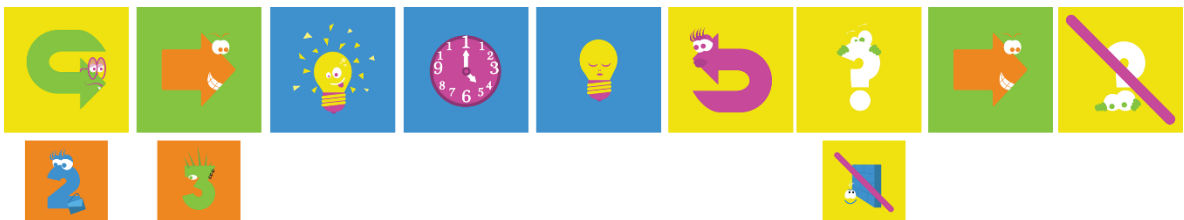


FIGURE 5. Snapshot of our application's GUI

The above snapshot will make the robot do the following two times:

- three steps forward
- turn on the light of the robot
- delay for a short time (5 seconds)
- turn off the light

- using the ultrasonic sensor, check for an obstacle in front of the robot
- if an obstacle does not exist move one step forward
- end the sensor check

PUC #2

-Error message with beeping sound for inappropriate command / parameter.

If a user tries to select a command or parameter that cannot be attached at the program timeline, an appropriate message will appear.

e.g. if for a conditional square icon the user tries to attach a numbered parameter. Only sensor parameters are attached to a conditional square icon the message could be "No numbers here".



FIGURE 6. Error message for inappropriate parameter icon

PUC #3

-As the program script is being interpreted, it transferred to the robot via Bluetooth. The Bluetooth connection is setup prior to the start of the program writing process.

PUC #4

-When a conditional icon is executed the robot uses the selected sensor. The sensor measurement is transferred back to the tablet via Bluetooth and the condition is evaluated. Depending on the evaluation of the condition (true-false) certain actions are performed by the robot.

PUC #5

-When the user enters our application he/she has the option of entering a username. That username together with a counter will form the filename for the .CSV file which will be copied to the database (e.g. John001.csv). If no user name is selected the filename comprises of the current system date and time (e.g. 10_10_2014_09h25m10s.csv). The whole process of script database storage is unobtrusive to the user.

PUC #6

-If the user wants to check for product updates, he selects the appropriate icon. If there is no active Internet connection an error message is displayed. Otherwise the tablet connects via ftp to the predefined address. The ftp address can be altered through the settings button of our application. When the downloading finishes the user is informed that the migration to the new software release will occur the next time he starts the application. The main use of the update process is to incorporate into the application new features such as new robot sensors (e.g. infrared-heat sensors) and new program commands and parameters.

PUC #7

-If the user wants to present the program flow in programming language-like syntax he can select the appropriate button and a pop up window will show the code. This product use case is intended for older children who can read and want to familiarize with basic programming constructs. An example of this kind of representation could be as follows:

Program script

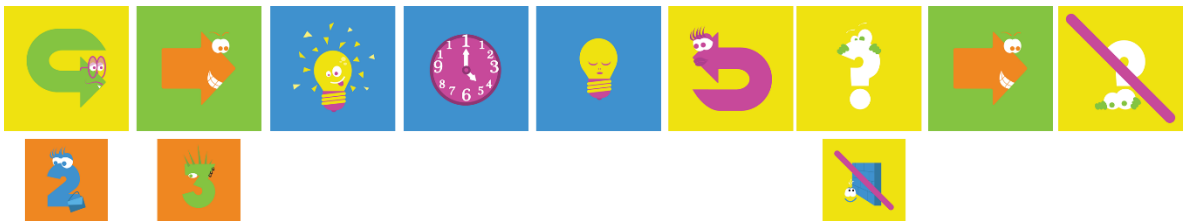


FIGURE 7. Snapshot of our application's GUI

```

FOR 2 TIMES DO
    3 STEPS FORWARD
    TURN ON LIGHT
    DELAY
    TURN LIGHT OFF
END FOR

```

FIGURE 8. Equivalent high-level syntax pop-up window of the above script (figure 7)

PUC #8

-When a condition is being evaluated -refer to PUC #4- the corresponding conditional icon will start flashing briefly (2 seconds) and its colour will change to green or red depending on the outcome of the condition (True or False respectively).

PUC #9

-Since the main audience of our product will be children of younger ages (4~10), one could select through the appropriate button a change of the overall theme of the application. It can include different coloured icons and background (e.g. pinkish tones for girls, bluish for boys) and differently stylized icons. Also as stated in PUC #6, one can update the software with the possibility to incorporate new themes in the future.

4.5.7 Functional & Non-functional software requirements

TABLE 8. List of functional and non-functional requirements

Requirement number	PUC Number	Requirement type	Description	Rationale	Fit Criterion
Unique identifier	Number of the related PUC	Functional, specific non-functional or	A one sentence text description of the	The reason why the requirement is important.	Measurement that makes the requirement testable.

	scenario	Constraint	requirement.		
1	#1	Functional	The product shall allow the user to select a command icon	For the robot to move commands are needed	The robot performs the appropriate command
2	#1	Functional	The product shall allow the user to select a parameter icon	A subset of the command icons can be coupled with a parameter icon to alter the command's effect	The parameter is attached to the lower side of a command icon at the program timeline
3	#2	Functional	The product shall inform the user for an inappropriate parameter	All parameters do not couple with all command icons	Flashing parameter icon and error message
4	#3	Performance	The product shall be able to export the script via Bluetooth to the robot	There is no physical attachment of the robot to the tablet	The transfer should not last more than 5 seconds if a Bluetooth connection is already established
5	#3	Functional	The product shall be able to establish in the background an active	An active Bluetooth connection is mandatory for bi-directional	Inform the user for successful connection or get error

			Bluetooth connection with the robot	communication of the tablet and robot	message
6	#4	Functional	The product shall be able to receive information (feedback) from the robot sensors	For the fulfilment or not of a condition	Flashing conditional icon (red-green)
7	#5	Functional	The product shall be able to export the program script to a database in .csv format	For future reference, open existing script	The transfer should occur automatically without user intervention
8	#6	Functional	Update product software version	If there is need to incorporate a new command-parameter icon (e.g. if there's a new sensor available example heat-sensor (IR))	The update process should first check a list of mirror servers and download and install the software. No user involvement required.
9	#7	Functional	The product shall be able to present the program in a high-level	Introduce older children to the syntax and structure of a real world	Code is presented in a pop-up window on the tablet's screen

			programming language style syntax	programming language	
10	#9	Look and Feel	The product shall be able to change the appearance of its graphical environment	Make it more appealing for girls and boys (e.g. bluish themes for boys, pinkish for girls)	Selection through an appropriate icon
11	#1	Usability	The product shall require no special training for a student to use it	The product's intended use is as a learning aid, not an integrated software development environment	Children shall be able to perform their tasks in a short period of time ≤ 1 hour
12	#1	Functional	The number of command icons at the program timeline must be restricted (Finite)	For the robot to move in confined spaces and not to lose its Bluetooth connection with the tablet	Default value of maximum number of command icons shall be 30
13	#1	Functional	Ability to change the maximum number of command icons present	Gradually as the user gains experience he/she may want to write more extensive	By accessing the settings icon of the application, a user can specify the

			at the program timeline	and complicated programs	maximum number of commands ranging 30~100.
14	#5	Performance	The Database shall be able to handle a lot of scripts	Automatic storage of scripts requires a lot of database records	Maximum number of scripts stored 10000
15	#6	Operational	Maintenance Releases	To incorporate new commands or Robot sensors or icon themes	Yearly releases
16	#2	Functional	The product shall be able to change the default language	For the various messages being displayed to be understood by the user	Selection via a submenu after selecting the settings icon
17	#1	Functional	The product shall prompt the user to enter a username at the start of the program	For easy reference to the database later	The file is stored in the with the appropriate filename
18	#2	Functional	The product shall be able to	To emphasize the event that	Error messages will

			produce beeping sounds	was triggered	be displayed while simultaneously a beeping sound is being heard
19	#1	Functional	The product shall offer the opportunity for a short video walk-around of the product	To inform the user	After the installation phase of the application
20	#1	Security	The product shall prompt the user for a password when selecting the settings button	To prevent unauthorized access, since children are going to use the product most of the time	Default value of password is TAPAC can be changed inside the settings menu
21	#1~9	Support	Our company shall provide a comprehensive means of supporting users of our product	To keep customers happy!	Help desk available (telephone), website, social media presence
22	#1	Standards	The product shall comply with the firmware	To successfully operate the robot	A command issues a response from the robot

			specifications of current Mindstorms™ NXT robot.		
23	#1~9	Operational	The software will run on Windows mobile tablet	To support Windows mobile platform	Seamless integration to the operating system

TABLE 9. Tasks roadmap-timeframe

Name of the phase	Required time to accomplish	Operating environment components included	Functional requirements included	Non-functional requirements included
Initial planning	2 month			
Software engine	3 months	Robot, tablet	1,2,3,5,6,7,8,9,12,13,16,17	4,20,22,23
Graphics design	2 month	Tablet	18	10
Pre-release phase, beta testing	1 month	Robot, tablet	19	11,14,15,21

5 SUMMARY AND FUTURE WORK

5.1 Issues that arose during problem analysis

5.1.1 Design new icons

Since our product leans mostly towards the graphical programming interface which was defined in section 2.3.1, it is imperative that the icons convey as accurately as possible the meaning of each command/parameter used in our mobile application. Furthermore the graphical language offered by the icons which, as we have demonstrated in table1 section 4.2, comprises of 13 command icons and 12 parameter icons, must take in account that the users of the application will be mainly children who might have no or little reading and writing skills. As stated before our target group begins with children as young as 4 years old which represent the majority of our hands-on users among others (section 4.5.3 table 4). Consequently it is hard to establish what type of design is appealing to children especially of younger age. As a future work one might suggest that there have to be questioners or videos of children's responses to the presentation of the system mock-up we have created (section 4.4, pictures 27 & 28). The questioners will try to grasp kids responses on the color, size and shape of each icon, also if they like the idea of offering differently colored themes for our application, a feature that is already included in our application in the form of a theme selection for differentiating boys and girls (section 4.5.6, table 7, product use case number 9). By evaluating the results from the above survey, we shall be able to better understand children's preferences and provide them with a better alternative.

5.1.2 Size of icons

The device which our application will operate on, is a tablet. The typical tablet screen size ranges from 7 to 10 inch diagonal screen length. This fact raises some constraints on how large the icons should appear on the screen (section 4.4, pictures 27 & 28). Our current

implementation supports 9 command icons present simultaneously at the program timeline with parameter icons attached at the lower side of a command icon (section 4.5.6, table7, product use case number 1). The rest of the program script shall be accessible by scrolling through the program timeline. Based on our previous experience and results, future work could deal with the following variations. The user could zoom in or zoom out in each icon thus making them appear smaller or bigger. Also the user could create a second or even third program timeline at the lower part of the screen. If this is the case, one could have present on screen at the same time from 9 to 27 command icons. Even more as is stated in section 4.5.6, table 7, product use case number 7 one can select to view the commands in a high-level text programming language style, if needed.

5.1.3 Length of program script

The Lego Mindstorms™ robot is an integral part of our system. Through the program script it can perform actions such as moving around, use its sensors etc. Sensors that are currently supported are Infrared, Touch, Color, Gyro, Ultrasonic (section 4.3.2, picture 26). Since our application is meant as a learning aid as is already stated in our project goals (section 4.5.2, table2, goal number 1), it will be mainly used into confined spaces like classrooms or homes. That is the reason why we have decided to limit the maximum number of command icons present at any program timeline to 30. As stated at the software requirements (4.5.7 Table 4, requirement number 13) table, the user can alter that setting to as much as 100 command icons (password is required). As a future work we could investigate-through surveys- how long will the average program script be and adjust that setting accordingly. The above measurement is possible since every program script is being saved as a .csv file in a database, as can be seen in section 4.5.6., table 7, product use case number 5. Furthermore we could attach a GPS sensor onboard the robot in order to be able to determine the tablet's distance from the robot. That could be useful for the robot no to stray too far away and also for the Bluetooth connection to work properly. As an example, our application would halt the program script execution if it detects that the robot is more than 10 meters away from the tablet.

5.1.4 Connection to robot

One might argue why we do not use the Wi-Fi connection option which is already supported in the LEGO MINDSTORMS EV3 API (see section 4.3.1) in order to achieve greater range of communication. Our response is that we chose only the option of Bluetooth connectivity as stated in the software requirements section (section 4.5.7, table 8, requirements 4 & 5) due below reasons. Wi-Fi and Bluetooth serve different purposes. Wi-Fi is for network communication on a wider range. Bluetooth is for close-range communication between two devices and this fits better with our purpose. Also because a Wi-Fi dongle must be connected to the USB host of the EV3 brick and since dongle is not included with the product we need to buy it to activate this feature (the only dongle known to work with the EV3 Brick is the NETGEAR WNA1100). Furthermore based on LEGO specifications, connection through Wi-Fi will consume more battery power than Bluetooth (Lego EV3 webpage 2014).

5.1.5 Operating system-Software distribution

The current implementation is for the Windows mobile operating system due to the fact that there are available software libraries for the connection with the robot in the Windows platform (section 4.3.1). As a future direction we could investigate if it is worthy to port our application to other tablet operating systems like IOS and Android. This decision is based on the fact that most lower specification -and therefore cheaper- tablets use the Android platform. As our application becomes popular and therefore fulfills one of our project's goals which is to become a useful learning software (section 4.5.2, table2, goal number 5) we could offer it to a larger audience. Another matter of consideration is if we are going to offer a full featured time trial version of our application or a downgraded version for free use. If the user is happy he could select to buy the application. Also special offers could be made to educational institutions.

5.1.6 Number of robots supported

The current version supports only one robot that can be programmed as can be seen in the UML diagram (section 4.5.5 Figure 3). We could examine if it is worthy for a user to be able to control a cluster of robots as in 2 or even 3. Variations to this approach could include the following. A single program script to control each robot independent of the other, each robot will have a unique identifier as parameter icon. A single program script which will run on all robots concurrently, like having three identical program scripts running at the same time. And finally different program scripts running on different robots at the same time. For the afore mentioned variations to materialize we shall overcome a significant obstacle we found during the development phase which is that the current LEGO MINDSTORMS EV3 API library can only be used with a single LEGO EV3 programmable brick -and therefore robot- at a time. Multiple brick communication is not supported yet (section 4.3.1, picture24).

5.1.7 Hardware constraints

Although the Lego Mindstorms™ firmware and hardware schematics are available free nowadays it may not be the case with future models. Also the current version of the robot has Bluetooth connectivity it may not be the case with future models (only USB connection). Having to communicate with the robot only via a USB cable will almost certainly prove to be a drawback for our product's appeal. Another issue if we are willing to offer backward compatibility with the firmware of older models (legacy support) of the Mindstorms™ robot.

5.1.8 Marketing strategy

We could approach the Lego Corporation to allow us to market our software as a combination package with the robot. If this is the case we could significantly increase our customer base. But we shall have an open mind and explore the possibility of contacting

other manufactures of programmable generic robots in order to establish a possible partnership.

5.2 Conclusion

Our proposal dealt with the problem of creating an application that could introduce children of a young age to the magical world of programming. The subject of educational software has many parameters and our approach to it was to try and merge the worlds of tangible and mobile graphical programming, creating an introductory programming ecosystem for children. We decided to create a graphical mobile isomorphic equivalent of an existing tangible programming system (T_ProRob system, section 2.4.2) that uses cubes as programming elements. Furthermore our application operates on a mobile tablet device thus making it more portable. In order to make the experience of programming more appealing and fun for children, we combined our application with a Lego Mindstorms™ robot (section 4.3) that serves as an actor for playing out the various programming scripts. The main features of our envisioned application include the following:

- Low power Bluetooth connection with the robot
- Ability to control a Lego Mindstorms™ robot
- Utilize robot onboard sensors
- Store each program script at a local database.
- Change the look -and-feel of our applications GUI.
- Update software to incorporate new commands or sensors for the robot
- Option to view program script in text form instead of icons

Through a careful selection of icons, intuitive design and inherent ease of use as specified in the software requirements section (section 4.5), we believe that our application can serve as an invaluable learning aid for children aged 4~10.

REFERENCES

Antle, A. N., Droumeva, M., & Ha, D. (2009). Hands on what?: Comparing children's mouse-based and tangible-based interaction. Paper presented at the Proceedings of the 8th International Conference on Interaction Design and Children, pp. 80-88.

Alice programming language. Read 23.11.14
<http://www.alice.org/index.php>

Baltie graphical programming tool. Read 23.11.14
<http://www.sgpsys.com/en/>

Blackwell, A., & Hague, R. (2001). AutoHAN: An architecture for programming the home. Paper presented at the Proceedings IEEE Symposia on Human-Centric Computing Languages and Environments, 2001, pp. 150-157.

Blackwell, A. (2003). Cognitive dimensions of tangible programming languages. Paper presented at the Proceedings of the First Joint Conference of the Empirical Assessment in Software Engineering and Psychology of Programming Interest Groups, Keele, UK. pp. 391-405.

CargoBot mobile application. Read 23.11.14
<http://twolivesleft.com/CargoBot/>

Cockburn, A., & Bryant, A. (1997). Leogo: An equal opportunity user interface for programming. *Journal of Visual Languages and Computing*, 8(5-6), 601-619.

Daisy the dinosaur mobile application. Read 23.11.14
<http://www.mindleaptech.com/apps/daisy-the-dinosaur/>

Design for Kids Digital Products for Playing and Learning Published: July 15, 2014
 Paperback: 248 pages, ISBN 1-933820-30-6 Digital: ISBN 1-933820-43-8

Guido van Robot. Read 23.11.14
http://en.wikipedia.org/wiki/Guido_van_Robot

Horn, M. S., & Jacob, R. J. K. (2007). Tangible programming in the classroom with tern. [CHI 2007 San Jose, CA, USA] , 1965-1970.

Horn, M. S., Solovey, E. T., Crouser, R. J., & Jacob, R. J. K. (2009). Comparing the use of tangible and graphical programming languages for informal science education. Paper presented at the Proceedings of the 27th International Conference on Human Factors in Computing Systems, Boston, MA, USA. pp. 975-984.

Horn, M., Crouser, R., & Bers, M. (2012). Tangible interaction and learning: The case for a hybrid approach. *Personal and Ubiquitous Computing*, 16(4), 379-389. Retrieved from <http://dx.doi.org/10.1007/s00779-011-0404-2>

Hopscotch mobile application. Read 23.11.14
<http://www.gethopscotch.com/>

Ichida, H., Itoh, Y., Kitamura, Y., & Kishino, F. (28 March 2004). ActiveCube and its 3D applications. Paper presented at the IEEE VR 2004 Workshop Beyond Wand and Glove Based Interaction, Chicago, IL USA.

Ishii, H., & Ullmer, B. (1997). Tangible bits: Towards seamless interfaces between people, bits and atoms. Paper presented at the Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, Atlanta, Georgia, United States. pp. 234-241

Itoh, Y., Akinobu, S., Ichida, H., Watanabe, R., Kitamura, Y., & Kishino, F. (2004). TSU. MI. KI: Stimulating children's creativity and imagination with interactive blocks. Paper presented at the Creating, Connecting and Collaborating through Computing, 2004. Proceedings. Second International Conference on, pp. 62-70.

Kelleher, C., & Pausch, R. (2005). Lowering the barriers to programming: A taxonomy of programming environments and languages for novice programmers. *ACM Computing Surveys*, 37(2), 83-137.

Kodu game lab. Read 23.11.14
http://en.wikipedia.org/wiki/Kodu_Game_Lab

Kwon, D. -, Kim, H. -, Shim, J. -, & Lee, W. -. (2012). Algorithmic bricks: A tangible robot programming tool for elementary school students. *IEEE Transactions on Education*, 55(4), 474-479.

Lego Mindstorms™ home page. Read 23.11.14
<http://www.lego.com/en-us/mindstorms/?domainredir=mindstorms.lego.com>

Lego Mindstorms™ EV3 API webpage. Read 23.11.14
<http://legoev3.codeplex.com/>

Lightbot visual programming game. Read 23.11.14
<http://lightbot.com/index.html>

Logo programming language. Read 23.11.14
[http://en.wikipedia.org/wiki/Logo_\(programming_language\)](http://en.wikipedia.org/wiki/Logo_(programming_language))

Maloney, J., Resnick, M., Rusk, N., Silverman, B., & Eastmond, E. (2010). The scratch programming language and environment. *Trans.Comput.Educ.*, 10(4), 16:1- 16:15. doi:10.1145/1868358.1868363

Manches, A., O'Malley, C., & Benford, S. (2010). The role of physical representations in solving number problems: A comparison of young children's use of physical and virtual materials. *Computers & Education*, 54(3), 622-640.

Marshall, P. (2007). Do tangible interfaces enhance learning? Paper presented at the Proceedings of the 1st International Conference on Tangible and Embedded Interaction, Baton Rouge, Louisiana, USA. pp. 163-170.

McNerney, T. (2001). Tangible computation bricks: Building-blocks for physical microworlds. Paper presented at the Proceedings of CHI 2001,

McNerney, T. S. (2004). From turtles to tangible programming bricks: Explorations in physical language design. *Personal and Ubiquitous Computing*, 8(5), 326-337.

Move the turtle mobile application. Read 23.11.14
<http://movetheturtle.com/>

Olkun, S. (2003). Comparing computer versus concrete manipulatives in learning 2D geometry. *Journal of Computers in Mathematics and Science Teaching*, 22(1), 43-56.

Orit, S., & Eva, H. (2009). Tangible user interfaces: Past, present, and future directions. *Foundations and Trends® in Human-Computer Interaction*, 3(1-2), 1-137.

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

Patten, J., Griffith, L., & Ishii, H. (2000). A tangible interface for controlling robotic toys. Paper presented at the CHI '00 Conference on Human Factors in Computing Systems, Hague, The Netherlands. pp. 277-278

Sapounidis, T., & Demetriadis, S. (2009). Tangible programming interfaces: A literature review. 4th Balkan Conference in Informatics, Thessaloniki, GREECE. pp. 70-75.

Sapounidis, T., & Demetriadis, S. (2011). Touch your program with hands: Qualities in tangible programming tools for novice. Paper presented at the 15th Panhellenic Conference on Informatics (IEEE/PCI), pp. 363-367.

Sapounidis, T., & Demetriadis, S. N. (2012). Exploring children preferences regarding tangible and graphical tools for introductory programming: Evaluating the PROTEAS kit. Paper presented at the Advanced Learning Technologies (ICALT), 2012 IEEE 12th International Conference on, pp. 316-320.

Sapounidis, T., & Demetriadis, S. (2013). Tangible versus graphical user interfaces for robot programming: Exploring cross-age children's preferences. *Personal and Ubiquitous Computing*, 17(8), 1775-1786. DOI 10.1007/s00779-013-0641-7

Sapounidis, T., & Demetriadis, S. & Stamelos I. (2014). "Evaluating children performance with graphical and tangible robot programming tools". *Personal and Ubiquitous Computing* DOI 10.1007/s00779-014-0774-3

Scratch junior mobile application. Read 23.11.14
<http://www.scratchjr.org/>

Scratch programming language. Read 23.11.14
[http://en.wikipedia.org/wiki/Scratch_\(programming_language\)](http://en.wikipedia.org/wiki/Scratch_(programming_language))

Small Basic programming language. Read 23.11.14
<http://smallbasic.com/>

Smith, A. C. (2007). Using magnets in physical blocks that behave as programming objects. Paper presented at the Proceedings of the 1st International Conference on Tangible and Embedded Interaction, pp. 147-150.

Tynker mobile application. Read 23.11.14
<http://www.tynker.com/>

Volere Requirements Resources. Read 23.11.14
www.volere.co.uk

Wyeth, P., & Purchase, H. C. (2000). Programming without a computer: A new interface for children under eight. Paper presented at the User Interface Conference, 2000. AUIC 2000. First Australasian, pp. 141-148.

Xie, L., Antle, A. N., & Motamedi, N. (2008). Are tangibles more fun?: Comparing children's enjoyment and engagement using physical, graphical and tangible user interfaces. Paper presented at the Proceedings of the 2nd International Conference on Tangible and Embedded Interaction (TEI '08), pp. 191-198.

Zuckerman, O., Arida, S., & Resnick, M. (2005). Extending tangible interfaces for education: Digital montessori-inspired manipulatives. Paper presented at the Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, Portland, Oregon, USA. pp. 859-868.

Zuckerman, O., & Gal-Oz, A. (2013). To TUI or not to TUI: Evaluating performance and preference in tangible vs. graphical user interfaces. *International Journal of Human-Computer Studies*, 71(7-8), 803-820. doi:10.1016/j.ijhcs.2013.04.003