ASHESI

ASHESI UNIVERSITY COLLEGE

DEVELOPING A MOBILE BASED APPLICATION FOR FEATURE PHONES TO TEACH CHILDREN IN RURAL AREA HOW TO CODE

APPLIED PROJECT

B.Sc. Computer Science

Michael Fiifi Quansah

2014

ASHESI UNIVERSITY COLLEGE

Developing a mobile based application for feature phones to teach children in rural area how to code

APPLIED PROJECT

Applied Project submitted to the Department of Computer Science, Ashesi University College in partial fulfilment of the requirements for the award of Bachelor of Science degree in Computer Science

Michael Fiifi Quansah

April 2014

DECLARATION

I hereby declare that this dissertation is the result of my own original work and that no part of it has been presented for another degree in this university or elsewhere.

Candidate's Signature: ………………………………………………………………………

Candidate's Name: ………………………………………………………………………………

Date: ……………………………………………………………………………………………………

I hereby declare that the preparation and presentation of the dissertation were supervised in accordance with the guidelines on supervision of thesis laid down by Ashesi University College.

Supervisor's Signature: ……………………………………………………………………

Supervisor's Name: ……………………………………………………………………………

Date: ……………………………………………………………………………………………………

ACKNOWLEDGEMENT

My deepest appreciation goes to the Almighty Lord for the gift of life and strength not only through the process of this dissertation, but also, through my stay at Ashesi University College.

I would like to thank my parents, Mr. and Mrs. Charles K.B. Quansah for the support fostered me through my stay at Ashesi. I am grateful to Dr. Awuah for pushing forward this Ashesi dream that for me, has opened my eyes to the need for my contribution to a developing Africa. It was at Ashesi that I developed a yearning responsibility to be a member of a burgeoning community of change-makers, for Ghana and Africa.

I am indebted to my supervisor, Dela Andy Kumahor, whose stern patience and coaching guided me through a worthwhile process and consequently, satisfactory outcome.

Finally, I would like to thank Randell Michelle of Ashesi University College, who helped coordinate the exchange between myself and Berekuso Basic School and the participants of the tests; Esther Ayi, Ibrahim Banafo, Caroline Oye, Daniel Kweku Ansah, Daniel Acquah and Cynthia Amadi.

ABSTRACT

It is important that while the world moves towards a digital age, people around the world build fluency in communicating with computers. In order to remain relevant, programming skills are highly necessary in the 21st century. In order to create such a culture, starting from the children is a step in the right direction.

Teaching programming to children in schools has become just as necessary as teaching them Mathematics, English or Science. In developing countries however, children are being left behind for two reasons; educational curriculums that do not support this new wave and systems that are not well structured or resourced to tackle this issue.

Nevertheless, it is necessary that children learn to program, however, it is even more important that children are taught to program in the way that they learn best, not the way that may be traditionally accepted.

The paper discusses a tool that has been developed on feature phones to teach children in rural areas, in the upper primary level how to code.

Key Words: Programming, computational programming, feature phones, maze, sequential processing

CONTENT

CHAPTER ONE
INTRODUCTION

"Programming skills are becoming ever more important, quickly turning into the core competency for all kinds of 21st Century workers. That inescapable fact is leading individuals to seek out new ways of learning code, start-ups and non-profits to find ways to help them and business to search for innovative approaches to finding the coders they so desperately need [1]."

Todays' society is largely digital; playing games, driving, browsing the Internet, sending emails, automated traffic lights, television remotes and toothbrushes. Essentially, all aspects of human life have been affected or are controlled by some form of computing. While being able to use these devices properly is necessary, it is imperative that we obtain fluency in computing. Fluency in computing or programming for that matter refers to the ability to understand and manipulate software or computer programs to fit one's preferences, or simply to understand how computer programming works and be able to manipulate it to one's advantage or preferences. The quote above highlights the importance of programming in our world today, but it merely casts a shadow on the importance of coding to the world today.

Programming is a process that involves detailed, step-by-step instructions for performing tasks, both simple and complex, on the computer. Programming involves the use of syntax (programming languages) to execute instructions in computer programs. Learning to program, goes beyond the possibility of building one’s capacity to become a good Computer Scientist, has a number of benefits;

- “It expands the range of what one can do with software, so that one is not limited to the features provided by standard application [2].” Effectively, one does not always have to depend on software or products from ‘experts’, one can also create (and customise) a product to one’s preferences.
- It supports the development of ‘computational thinking’, providing experience with important problem-solving and design strategies that extend into non-programming domain [2].
- It greatly expands the range of what one can create with the computer, while also expanding the range of what one can learn [3].

While programming has become a highly necessary skill for all, it has remained or has been reserved for just a small group of people. Historically, people have avoided Computer Science or programming for a number of reasons;

Firstly, the general perception is that individuals who learn how to code are usually bound for a profession in Computer Science. Following that trend, a huge percentage of people only take courses in Computer Science either because they want to pursue a profession in the field, or they want to carry out a project that may require some programming skill of some sort.

Secondly, Computer Science courses have been perceived to be difficult. Invariably, for most people it is a new language. Learning another language is difficult for most people. Computer programming languages are artificial languages; they deal with the unfamiliar world of data structures and algorithms. Thus, like trying to acquire a new language, people sometimes struggle with picking up programming languages. The fact that it is tends to be taught in later stages of childhood or teenage development, poses a problem. Not only do younger children miss out on an early start at programming, where children learn with curiosity and through gestures, but also they get used to concepts sooner than later.

As stated earlier, our world has become heavily dependent on computers, thus, it is highly necessary that there are effective and accessible ways to develop programming skills in individuals. Just like it is a necessity for people to be able to speak or at least understand English or certain languages to be able to communicate in our fast moving world, it is important to be able to communicate with computers in our increasingly digitized world. Learning to program should not be about becoming a Computer Scientist or programmer; it helps one, as mentioned earlier, obtain fluency in, and have the ability to be relevant in the computer generation. Learning to program is important and it is important that it starts early in life. Thus, providing an environment where children are taught how to program and understand computers is important.

Teaching children how to code is important;
"the child himself will learn to manipulate, to extend and to apply to projects, thereby gaining a greater and more articulate mastery of the world, a sense of the power of applied knowledge and a self-confidently

realistic image of himself as an intellectual agent [4].” The primary goal of teaching children programming is to nurture the development of a new generation of creative, systematic thinking individuals who are comfortable using programming to express their ideas.

Teaching children how to program, especially from a young age, has to be done properly. As stated earlier, children mostly learn through gestures and mimicking what they see. However, the manner in which programming has been taught historically does not favour this way of learning. Introductory programming lessons usually lunge into syntax and algorithms, with little emphasis on graphical output or feedback. It is important that rather than teaching children the way programming has normally been taught, it is important to teach them the way they learn. Some other factors that have inhibited the flow of effectively introducing programming to children include:

- Early programming languages were too difficult to use. Many children had difficulty mastering the syntax of programming.
- Programming was often introduced with activities (generating lists of prime numbers, or making simple line drawings) that were not connected to young people’s interests or experiences.
- Programming was often introduced in contexts where no one had the expertise needed to provide guidance when things went wrong – or encourage deeper explorations when things went right [3].

Programming languages should be basic enough for anyone at any appropriate age to start with and should be able to have opportunities for increasingly complex projects over time. An appropriate age in this context is the age at which a child can read and understand basic instruction. Also the teaching style should be flexible enough for people with different interest such that different learning styles can be accommodated.

In light of some of these concerns, a number of programming applications have been developed with children in mind. They focus on graphics and animation that are in the form of games that engage the child. Most of the applications that have been developed in the industry focus heavily on creative thinking and computational programming. Creative thinking in essence is being able to think about problems differently; to be able to think outside the box, to arrive at an optimum solution. Computational programming involves a step by step and sequential way of solving problems, where one progresses through a problem solving by dividing and solving.

An example is MIT's Scratch. Scratch is a lifelong project designed by the Lifelong Kindergarten Group at the MIT Media Lab. Scratch is an online platform designed for children between eight and sixteen to program interactive stories, games and animations. Scratch helps young people think creatively, reason systematically, and work collaboratively - essential skills for the 21st century [5]. Other applications that have developed models that work with this concept include Alice, Kodable, Hopsctoch, Tynker, Small Basic and Daisy the Dinosaur. Most of these applications or platforms are available either online as web services, or can be downloaded on mobile devices, to be run as applications. Most of the tools focus heavily on using interactive games and 'fun' projects that help to teach children logical and creative thinking, and computational programming. Others like Small Basic, is solely syntax-based. It was developed to help give children and novices an introduction to programming. While some of these tools provide effective ways to teach programming to children, they focus on only one side; either they use only graphical elements to try to teach programming, or they try to teach only syntax directly to children.

However, this project presents a tool that works with a combination of syntax and a graphical approach, which allows the children to appreciate the connection between the coding languages and their output, in this case, graphical. The project pushes the boundary of teaching programming with just either syntax or graphical outputs. Children while learning how to program should be able to see the results of the syntax that they apply. Thus it is imperative that a tool be developed to combine teaching children both visually and providing them with an introduction to the syntax.

The tool developed in this project is aimed at providing students in rural areas an opportunity to learn programming. In countries such as the United Kingdom [6] and Estonia, programming has been added to school curriculums, such that students start learning how to code by the age six [7]. In Ghana and other developing countries, programming is taught at a late stage. Most people are not introduced to programming till they get to university, and for those who may not take a Computer Science class, they may never get the chance to learn programming, let alone understand programming.

Additionally, access to opportunities to learn to program largely remains a privilege. The environments that harness this development are restricted to attending schools that can afford to buy computers or handheld devices, mostly smartphones or tablets to teach their students. Applications on mobile devices and computers are also limited to the part of the population that can afford them. Thus it is important that the opportunity is created for people, specifically children who may not have the opportunity or access to computer programming tutorials.

1.1 Problem Statement

While the world shifts into a digital age, the need for programming is increasing. For one to stay relevant in the fast changing world, it is important that one is able to understand computer programming. However, in the developing world, people are being left behind. There is the need for a system to be developed that helps reach and teach children how to program at an early stage.

1.2 Objective

The issue is multi-faceted thus the solution to this phenomenon must not only help to make programming accessible to children of all ages and in all socio-economic conditions, but also present a manner in which programming can be effectively taught to children (pedagogy). The objective of this project is to develop a tool that will help introduce programming to children in a fun, comprehensible and progressive manner. The tool that would be developed has two components. Firstly, a Java based mobile application on feature phones will be built that will help teach children in the Upper primary level (aged nine to eleven) how to program. The second component of the tool is to develop syntax that is understandable by the children to be used for the game, and also as an introductory language to programming.

1.3 Outline of Dissertation

The dissertation seeks to outline proposed solutions to the need for programming among children in rural areas. The dissertation is divided into five chapters.

Chapter One introduces the phenomenon of programming and its importance in the backdrop of a world which is rapidly entering the digital age. The introduction further outlines the necessity for children to be taught programming in a way that is both fun and productive. It also discusses the objectives of the project and problem statement.

Chapter Two discusses the design and design considerations of the tool that was developed to provide a solution to the problem. It also includes functional and non-functional requirements, and finally, use cases of the tool.

Chapter Three discusses the implementation of the tool and the platforms, constraints and concessions that were made in this regard.

Chapter Four outlines the tests performed and discusses the results obtained.

Chapter Five makes conclusions and recommendations for scaling of the tool. Discussed in this chapter are also questions posed about the way programming should be taught and how the tool can be extended to other age groups.

CHAPTER TWO

Solution

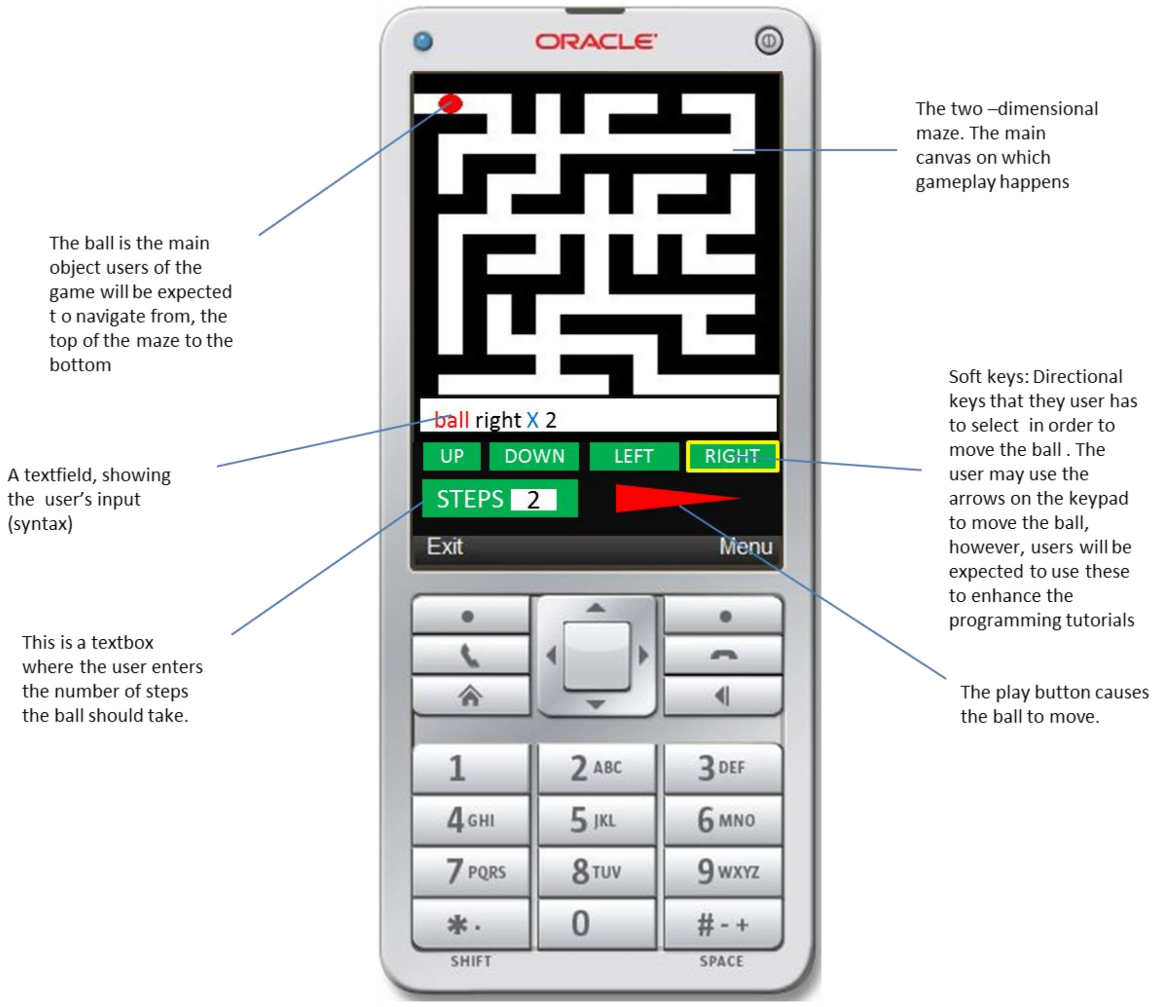
This chapter discusses the proposed solutions and the considerations made in coming up with the solution.

This project seeks to come up with a tool that effectively introduces children in the upper primary level, in rural areas, how to program. The tool has two components; a mobile game for students to learn while playing, and a syntax or language that not only can be used to play the game, but also, can be developed as an introductory syntax to programming at their level.

The mobile game-based application works on Java-enabled feature phones. The game is in the form of a two dimensional maze viewed from an aerial perspective. The aim of the game is to navigate a ball to the end of the maze. The ball's movement is controlled by the syntax that is developed alongside the game. The game further has levels or difficulty levels that enable one apply more syntax to solve similar but more complex problems.

Below is the prototype of the game. The screen is divided into a game canvas and a control pad; first the maze which is the game that is navigated by a ball. Then the soft keys which enable the user input the direction the ball takes. As the game is tailored to children, it is important that all the controls are visible and self-explanatory. The soft keys are left, right, up and down for the directions, and then a play button to enter an input. A text box is provided for specifying the number of steps a ball should move, as well as a text field, which shows the input made by the user.

For the syntax, it was important that the language used would closely resemble everyday language used by the children at school, or in regular conversations.



ORACLE
The two –dimensional maze. The main canvas on which gameplay happens
The ball is the main object users of the game will be expected t o navigate from, the top of the maze to the bottom
ball right X 2
UP
DOWN
LEFT
RIGHT
STEPS 2
Exit
Menu
Soft keys: Directional keys that they user has to select in order to move the ball . The user may use the arrows on the keypad to move the ball, however, users will be expected to use these to enhance the programming tutorials
A textfield, showing the user's input (syntax)
This is a textbox where the user enters the number of steps the ball should take.
The play button causes the ball to move.
SHIFT
SPACE

SCREEN SHOT OF THE MAZE AND IT'S GAME CONTROLS

2.1 Mobile Phones (Mobile Learning)

To enable widespread deployment of the tool to as many rural schools in Ghana, the game will be developed on a feature phone. The provision of quality education does come at a mean price. To teach programming effectively, one needs to have access to a computer. However, most of the schools in the rural areas in Ghana can barely afford textbooks, let alone regular computers. Access to computers in Ghana is limited. While we are on the edge of entering a digital age, most families, schools and societies are still left behind. The cost of a computer and maintaining one is more than a number of families and schools can afford. While the provision of quality education continues to suffer, mobile communication has grown rapidly over the past couple of years. As of 2011, there were over six hundred and twenty (620) million mobile connections in Africa alone [8]; and research in early 2014 showed that there are over twenty-eight million phone subscribers in Ghana [9].

These connections and the ease with which one can obtain a phone have led to the propagation of mobile learning, especially to the rural areas in parts of Africa, including Ghana. In Ghana, mobile phone penetration, (mobile penetration in a country indicates the number of active mobile phone numbers connections [10]) has reached over 100% according to the National Communication Agency [11]. Clearly, mobile phones are widespread through Ghana, thus harnessing their availability and accessibility to propagate education to the rural areas is an excellent model. Mobile learning is able to extend delivery of education in ways not seen before, thus the choice to use phones for deploying the tool.

Mobile phones can be grouped in three categories; basic, feature and smartphones. Basic phones allow users basic calling and messaging functionality and operate on very low end software. Feature phones are also low-end mobile phones. However, unlike basic phones, they incorporate features such as the ability to access Internet and play music, yet they lack they advanced functionality of a smartphone. Smartphones are phones that are able to perform many of the functions of a computer, typically having relatively large screens and an operating system capable of running multiple general-purpose applications simultaneously [12].

Feature phones are built to last. While smartphones support highly flexible features and platforms that allow users to easily run applications, feature phones are durable and will be 'safer' in the hands of children. Additionally, as far as cost, affordability and accessibility go, the choice of feature phones is best fitting for this project.

Smartphones are more fragile than feature phones, and thus more care has to be taken in handling them. In this vein, distributing them to children who will be using them as they use text books will be a costly venture. Additionally, not only are smartphones relatively more expensive than feature phones, repairing a smartphone also has higher costs than a feature phone. On average, smartphone prices range between 300usd to 750usd, while feature phones range between 20usd and 50usd [13]. Smartphones also have poor battery life, and usually require more than at least one full charge per day, whereby feature phones can be used over a period of days without the need of charging. In most rural areas, power supply is limited and unreliable, thus keeping the phones powered over a period of time, is essential. Admittedly, smartphones because of their advanced software provide better flexibility with the deployment of applications and games. However, the benefits of deploying the game on a feature phone, which include better accessibility, relatively higher durability and lower cost, far outweigh those of the smart phone.

The use of mobile phones perpetuates and takes advantage of the benefits of mobile learning. "Mobile learning offers modern ways to support learning process through mobile devices, such as handheld and tablet computers, MP3 players, smartphones and mobile phones [14]." Mobile learning presents unique attributes compared to conventional e-learning and learning methods, as instruction and learning can be done 'on the go.'

2.2 Demographic

Ghana's education curriculum requires that at the Upper primary school level, children (primary 4-6) aged nine to eleven, "are encouraged to reflect, think creatively, find out things for themselves to satisfy their curiosity, ask questions, criticize, solve problems, observe, view information critically, and assimilate new knowledge [15]." Thus, the choice to work with children at this level, is not only in line with their curriculum, but also, provides them with skills necessary for holistic academic development.

Berekuso is a small town of about two thousand people in the Eastern Region of Ghana. It is mostly rural and the locals by and large deal in farming. The town has one basic school where the quality of education leaves much to be desired. In 2012, the town recorded a zero percent pass rate in the Basic Education Certificate Examination, a standardised examination, taken by all students after junior high school (ninth grade) in Ghana. Basic school education and junior high school education in Ghana is designed to prepare students for this examination. Thus a zero percent score is a reflection of the gaping lack of quality in the system at Berekuso. Over the past years however, since Ashesi University College relocated to Berekuso, a number of community service initiatives have helped improve standards in the school. Students from Ashesi help teach the children in the poorly resourced school. Thus, testing the tool with children from Berekuso will help provide feedback to correctly gauge the application's effectiveness.

2.3 Using a Maze

The idea of the maze is to engage children in interactive games, while teaching them basic concept of programming. To navigate through a maze, one has to make step-by-step decisions. Based on the outcome of an input, the user either moves on or has to retract their movement. A maze closely simulates the sequential processing nature of a computer program. A maze's characteristics help users appreciate step-by-step processing, which is the basic way a computer handles instruction. Through step-by-step decision-making, users can navigate the maze. As the maze increases in difficulty, users will have to make more decisions and sometimes apply innovative methods in navigation.

The maze presents students with different possibilities of navigating their own route to reach the distance. The choice of the maze encourages the children to think computationally and creatively. Our current educational system is inflexible; where there is usually one way of doing things, where children are constricted to thinking and solving problems one way. However, programming involves making use of different algorithms that may give the same result but have different levels of efficiency. Thus, not only does a maze make for a fun game, but also for a productive learning experience.

2.4 Functional Requirements

- The game's sprite (a two dimensional image used to represent a moving object) should be navigable by syntax that is custom-created and also by the directional keys on the keypad.
- The game will feature music and sound effects, but will not be crucial for gameplay.
- The game must have an instruction scene, a 'change-level/difficulty level' screen. The user should be able to navigate from the any screen back to the gameplay screen.
- At the completion of each level, the user should receive a prompt that guides the user to a higher level.

- The sprite can move up, down, left and right, using the arrow keys. At a wrong execution, the game should send feedback to the user about an error.

2.5 Non – functional requirements

- Portability - The system should be deployable on Java-enabled phones. This should install fluidly via an online link or downloaded via USB. Finally, there is the option where the phones can have the games preinstalled on them. The phone should have at a minimum, 1MB worth of RAM space for installation.

- Flexibility - When the game is interrupted by a call, or other activity on the phone, the user should be able to 'resume and continue at the point that they left off.

- Scalability – Even though the game, and the concept at large, were largely designed to be on Java-enabled phones for the purposes of the initiative that is being served, it is possible to run the system on other platforms. Through a platform such as PhoneGap, the game can be deployed on Android and iOS platforms. The game should also be able to support online play and interaction. This will create the possibility for interaction with other users of the platform, including tutors.

- Security - Access permissions for systems data may only be changed by the system's data administrator i.e. the game creator.

- Usability – After every command is issued, to help teach the user programming, syntax will be shown in a text field before the sprite moves.

  The maze is black, over a white canvas, thus providing a very clean interface that helps the user easily navigate to the end of the game. The game should also incorporate sound effects to provide feedback to the user at various levels of game play.

2.6 Scenarios

This section outlines the major gameplay scenarios and how they contribute in reaching the goal of effectively engaging the user to both program and have fun.

1. Gameplay
2. Programming to play

2.6.1 Gameplay

The maze can be operated in a regular manner, as other mobile games; where the joystick of the phone's keypad can be used to navigate the ball. However, to help introduce the children to programming in an effective manner, the user will have to use commands to navigate the maze. Fig.2 is a screenshot of the game at start of the game. The user will be expected to navigate the sprite, which in this case is a ball, at the top left corner of the page to the bottom right. To navigate the ball, the user selects a combination of soft keys that will move the ball in the direction that he desires. In Fig 2, the selection has been made on the "Right" button and the number of steps to be taken is two, as shown in the text box, for "Steps". The user's input (combination of 'direction and steps') is shown in the text field above. Assuming the user wants to edit the input, he just needs to select a new direction and or change the number of steps. The number of buttons on the screen is kept to the minimum and most necessary so the user can focus on gameplay. When the user is prepared to move, he selects play, and the ball will move in the input specified.

Fig 3 shows a completed maze. Once the maze is completed, a message pops up on the screen. In this case, "Welcome to Berekuso!" The message can be changed by the administrator. Something as familiar and simple as "Welcome to Berekuso" was chosen to aid the users' familiarity with the game.

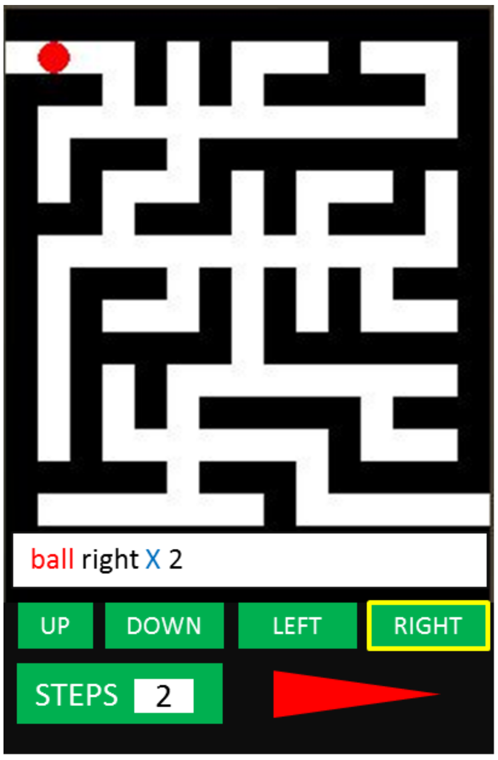


Fig 2. START OF GAME

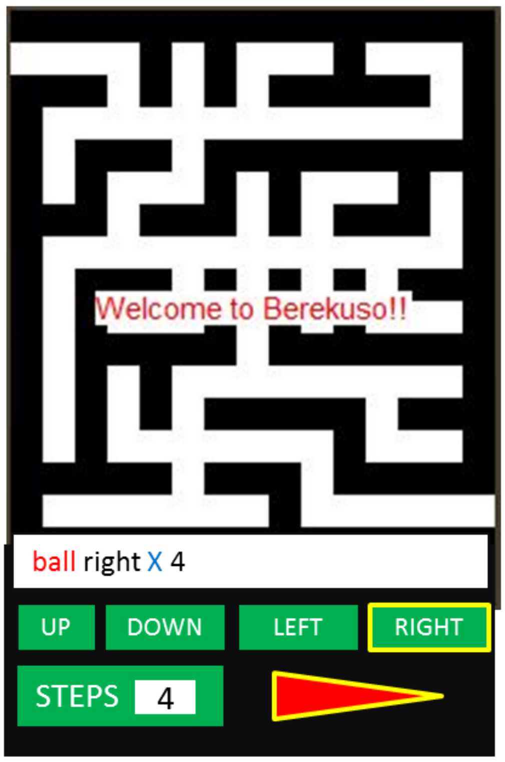


Fig 3. END OF GAME

In the case where the user enters an input, that will not be successful, the game responds with a pop up that alerts the user of the error. Thus at any given point, the game's algorithm is aware of the maximum number of steps that can be taken in every direction, and thus reacts to user input in that manner. An example is shown in Fig 4 below.

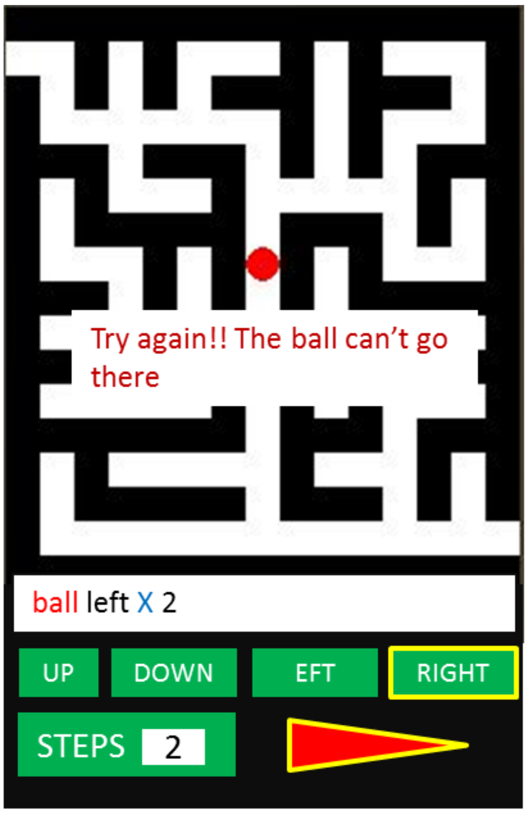


Fig. 4 HANDLING WRONG SYNTAX

2.6.2 Learning how to code

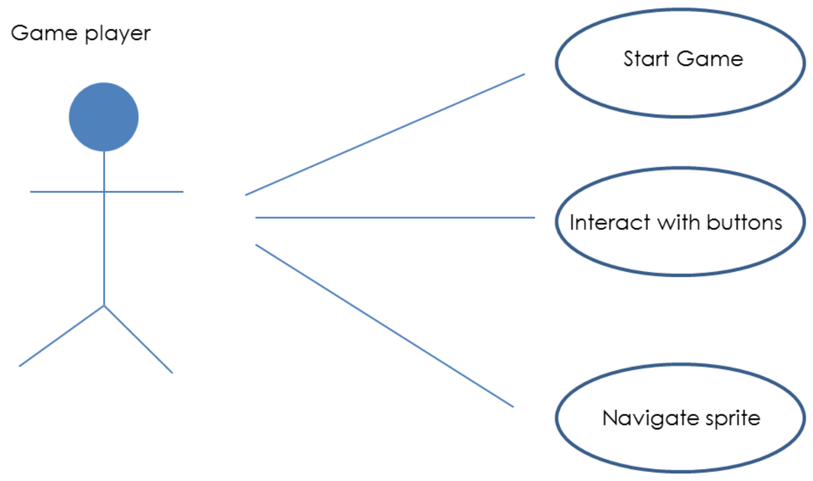
This is a basic introductory level. The aim is to give users a first hand overview that will encourage them to think of the required input and create instructions to successfully navigate the maze.

The text field below the maze shows the user what the input is, thus they can appreciate how the input or syntax translates into movement of the ball.

The input in the text field "ball right x 2" is formed by the user selecting the "Right" button and secifying the number of steps "2". The "ball" and "x" are already prefixed in the input.

2.7 Use Cases

The use case diagram below describes the interaction between the user and the game on the three different (main) levels; starting of the game, interacting with soft buttons on the game's canvas and also navigating through the maze with the ball. To start the game, the user merely has to start the application. Interacting with the buttons involves using the soft keys to navigate the direction the ball moves through the maze.



CHAPTER THREE

Implementation

The third chapter discusses how the product was developed, on what platforms and how it should be deployed onto the selected channels.

The maze was built using j2ME (Java Mobile Technology). This allows for the game to be deployed on Java-enabled handheld devices. The development was done on the Netbeans platform. Netbeans is an integrated development environment that allows users to create and run applications. The mobile development has an emulator thus enabling immediate testing as building is going on.

The game's maze was built on an algorithm that generates random mazes any time the game is started or reset. The algorithm works by setting up a maze with a different orientation any time the user requests for a new maze or finishes a level. This is done by making use of the standard Java classes, Vector and Random. The game has different difficulty levels. The higher the level, the smaller the column sizes of the maze walls. The application can be uploaded on the phone via USB connection, can be downloaded over the Internet and can also be sent from device to device via Bluetooth connection.

The development of the game followed an agile method; based on iterative and incremental developments where development and requirements evolve from collaboration with testers and end users. Modifications were made from feedback made by users before final deployment.

Finally, the game was deployed on the Nokia C2 phone. The Nokia C2 is a feature phone that supports Java-enabled applications. It was chosen due to its high availability, durability and good battery life. The game, as stated earlier, should be deployable on regular Java-enabled phones.

CHAPTER FOUR

Testing and Result

This chapter describes the process of testing the application and also the proposed pedagogy. Firstly, while the mock ups and concepts had been developed fully, the application was only developed to the level of the maze. Thus the testing had to involve simulations that would allow the participants to have a holistic experience of the game. The game was also not tested on the proposed phone, but was tested on the emulator in Netbeans. During the period, some experiments were carried out based on some of the developments from the tests.

The participants were six children, from class four to class six, two from each level, from the Berekuso Basic School. Only two of the children from class six had had any prior experience with computers in an organized class setting. Aside specifically choosing two from each class, the selection process was purely random. Academically, they all performed above average at school.

The testing period spanned a period of two days, for about 3 hours a day. The first day introduced the children to basic computer knowledge, allowed them to play with the maze, performed experiments (outlined below) that would test the application's efficacy and also together with myself developed a syntax that could be easily taught and understood by the children.

The second day of testing included quizzing the children on the concepts taught the day before. They were quizzed in two ways;

1. Using the language developed the previous day to navigate two mazes drawn on the whiteboard.
2. Using the language (verbally) to navigate the maze on the emulator

The final part of the testing was comparing the proposed solution to an already developed solution, Kodable, an application available on the Apple iTunes store.

4.1 Test Day One

Knowledge of the level of expertise the children had, helped provide a good angle as to how to introduce them to programming. Eventually, should this project be included in curriculums, it would be an introductory level course, with emphasis on teaching the children the basics of programming. However, it is important that the children have at least, a very basic introduction to using computers and cell phones. The children were introduced to basic concepts of computers. This included teaching them how to use a mouse and a keyboard to perform basic operations on the computer. At best, the children could type out their names, but were not necessarily acquainted with the arrangement of the alphabets on the keyboard and the special characters. They struggled finding the arrows, punctuation marks or even the space bar. Clearly, this was due to the fact that, their tutelage in Computer Science had been at a very basic level and had been limited to typing their names.

After the basic introduction to computers, the children were made to type out their names, classes and ages. Through this process, they had the chance to explore some of the keys they had never used before. These included the arrow keys, enter key and backspace key. This exercise helped prepare the children to use the computer and the also the emulator on which the maze was developed.

Secondly, the children played around and gained familiarity with the maze, which they all did successfully. Testing involved allowing children to use and critique the system; from look and feel of the application, to the functionality.
As stated initially, most of the applications designed to teach programming focus only on teaching children graphically. However, it is important that while children are being introduced to concepts such as sequential programming and using algorithms, they gain a good appreciation of how to program, using syntax. Thus a number of experiments were conducted to help the children grasp basic concepts of programming, and also to help them think creatively.

4.2 Blindfold experiment.

The blindfold experiment was performed to encourage the children to gain familiarity with the keywords to be used in the game. Firstly, the experiment involved me closing my eyes, and following the children's guiding my navigation through the maze. After performing this through several mazes, the children understood the model of how the ball moved in consonance with the directions that were provided. A few times when they gave the wrong direction, or when I moved wrongly, they would backtrack to get on line. Another aspect that was introduced here was specifying the number of steps in the direction that was chosen. An example of this was saying “right times two” to move the ball two steps in the right direction,

Afterwards, the children were put through a system of “paired programming” where one had to guide the partner through the maze, by using the terminology (syntax) which we had built (described later). Paired programming is a process whereby two people work on a programming problem together, where one, the ‘driver’ programs, while the ‘navigator’ watches over the work being done by the other, makes corrections or additions. In this case, the driver was ‘blindfolded’ and the navigator had to guide them through the maze by using the syntax. This exercise involved them making use of keywords “up”, “down”, “left” and “right” and also specifying the number of steps that had to be taken by the ball. The exercise happened to be relatively fluid, as all the groups managed to get to the end of the maze in less than two minutes.

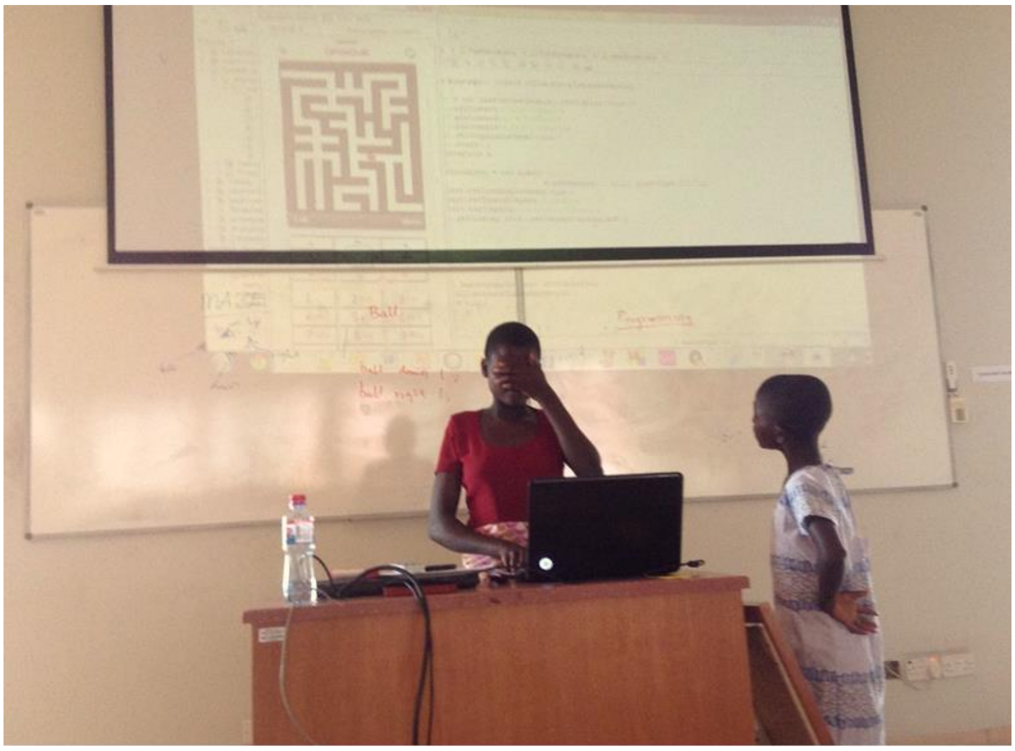


Fig.5 one participant guiding another (who is blindfolded) through the maze

The children showed an appreciable level of comfort using the directional keys. Additionally, anytime they reached a dead-end, all groups were able to navigate their way back to the correct route. This experiment showed that the children were very comfortable using the arrows and understanding the direction. This experiment was performed to simulate the step-by-step nature that computers work, or in programming, how a compiler works.

4.3 Forming Syntax

The blindfold experiment was the first step in teaching the children that the movement of the ball was dependent on the input made. After the blindfold experiment, the children had formed loose syntax to operate the ball. It is however important that the language be standardized, such that it can be replicated for other scenarios. By seeing the graphical output of the inputs they made, the children would be able to understand how programming at a basic level works. This contrasts some of the basic programming syntax taught at the most basic levels of programming. An example is how "System.out.print("Hello World");" in Java, may not be as intuitive as "ball right X 2" in the syntax the children used. We then proceeded to the next stage of forming the syntax. They easily identified the ball (sprite) and were able to navigate the ball relatively easily through the maze.

During the blindfold experiment, to move the ball in one direction a specific number of times, the child would repeat that specific direction the number of required times it would take to move the ball.

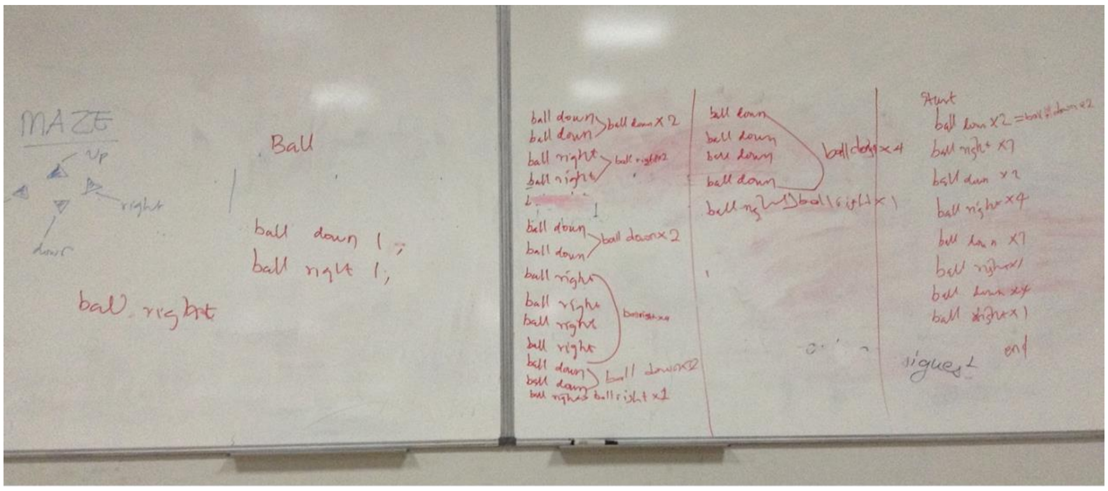


Fig.6developing the syntax through 'repeated' commands

To help condense this set of instructions into a more efficient system, we borrowed from normal conversation of speech. i.e. In an instance where someone needs to perform the action specific times, rather than repeating that command the required number of times, as in "Move forward, move forward, move forward”, a more efficient way is in saying “move forward three times.”

With this establishment, we were able to standardize the syntax. To move left four times, we would use the command “ball left X 4.” This system worked effectively for the children. Basic nuances such as using “X” instead of “*” which is accepted programming syntax for the product, were affected because it is important to make the syntax or pedagogy in general as familiar as possible to the everyday lives of the children. At school, they would use “X” instead of “*” for multiplication.

And thus, we were able to develop a syntax that is both easy to teach and easy to understand by the children.

4.4 Test Day Two

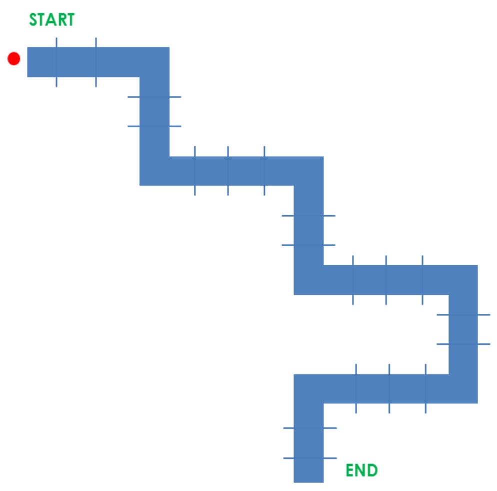
The second day of testing involved quizzing the children and observing how they interacted with different versions of the maze with the syntax we had developed the following day.

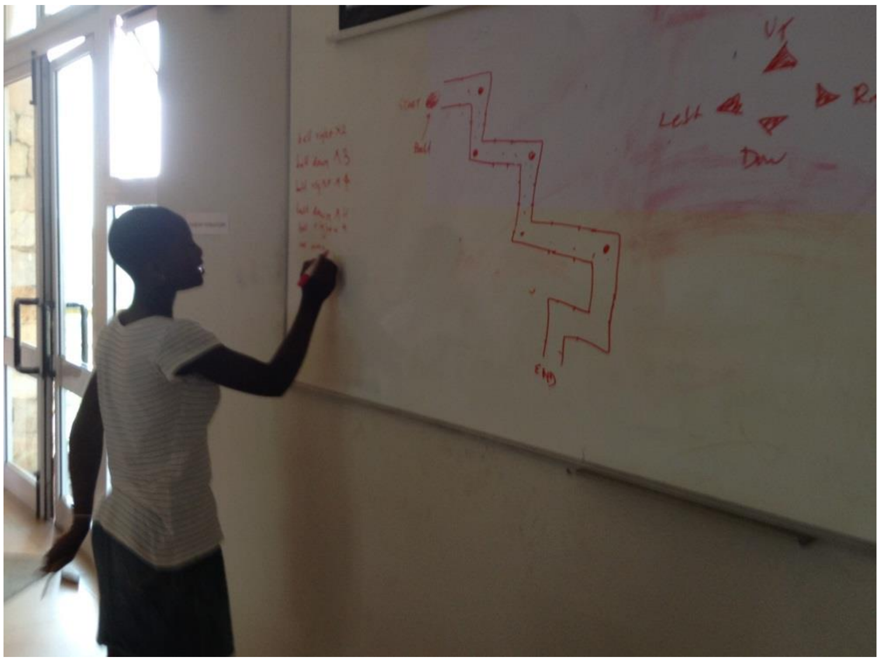
The first exercise involved the children working their way through two separate mazes, drawn on the whiteboard. They had to provide syntax for the code on paper. They did this with no assistance from anyone or one another.

The first maze followed a similar format that had been used the previous day. The children had to navigate the ball from the top left to the bottom right. In a maze like that, the dominant directions are “right” and “down”.

The maze was drawn on the white board by hand, and graduations were placed arbitrarily, as shown in the image below. This was to aid the children estimate the number of steps needed in a particular direction.

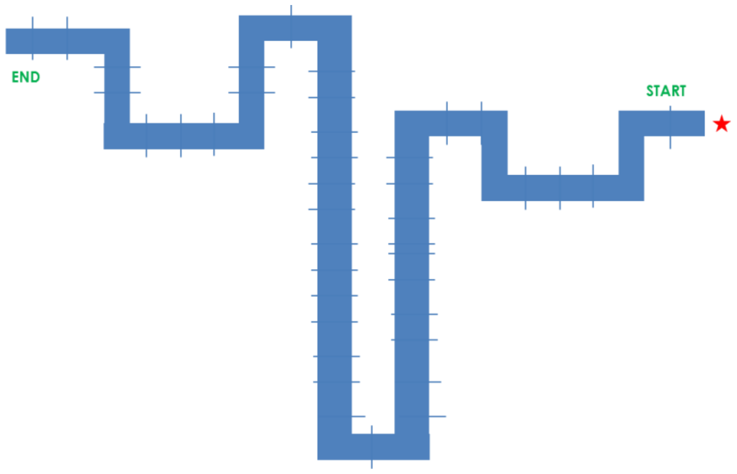
All the six children were able to correctly write the syntax for the direction. However, they struggled with the number of steps that had to be taken. Though they all realised the graduations guided the number of steps that had to be taken, they faltered with precisely how many steps had to be taken. Another reason they might have struggled initially was the fact that unlike the previous day of testing, where they worked on the laptop with the emulator, they had to work on paper. Working with the emulator, they were able to get immediate feedback of actions they took. However, on paper where there was no feedback, it seemed a bit difficult for them to make accurate estimates.





4.5 Flipping the maze

In the second exercise, the maze was inverted; rather than the conventional top left to bottom right, this maze went from bottom right to top left. This forced the children to think differently and creatively. Two of the participants, however, ended up using some of the instructions that would require moving the opposite way (as in the regularly oriented maze). This was indicative of the style of learning in Ghana, where most people depend on rote memorization. However, the four other children were able to recognize the difference and successfully navigate the maze.



The second maze drawn in the inverted way, (right to left, down to up) caused the children to think differently. The two children who failed to navigate the maze correctly showed dependence on rote memorization, rather than thinking creatively and differently about each problem. The rest of the group however, correctly applied the syntax in the right direction.

4.6 Orientation Lesson

The inverted maze quiz exposed gaping holes in the children's abilities to correctly differentiate between their left and right. Two of the children struggled with deciding whether the ball should go to the left or right, because they were unable to translate the word left into the direction or orientation. Thus, an exercise on coordinates was given. The children were each made to stand in a square tile on the floor and were given instructions to move left, right, up or down. To give the command, the syntax that had been developed for the maze was used. Thus for a child named Daniel to move three squares up, the command would be, "Daniel Up times three." This reflected the commands they had learned earlier in controlling the ball. Another aspect of this experiment was to simulate the ball's reaction to space constraints. In some cases, the child would be unable to fulfil the full length of a command because of space constraints. This scenario helped explain to them the results of giving wrong inputs to the sprite, or ball.

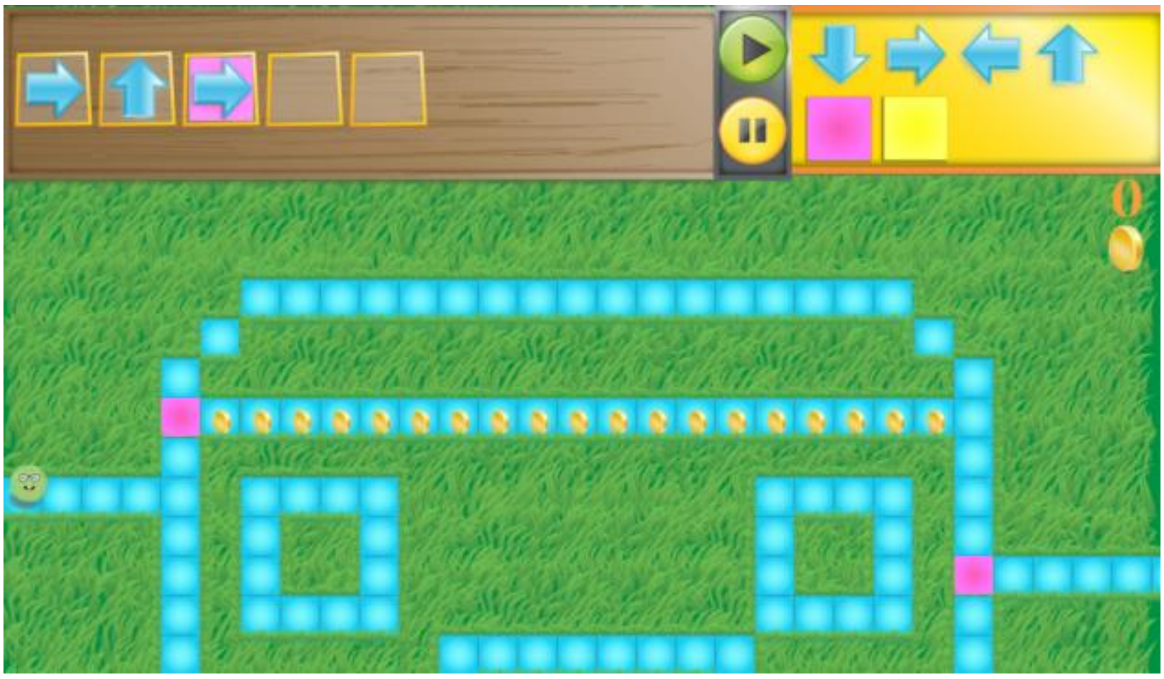
Fig 7. The participants learning about coordinates

4.7 Testing Kodable

The idea behind the development of Kodable, is that, while children are learning to speak or learning their first language, they should be given the chance to learn to code as well. Kodable seeks to develop the child's ability to code as early as possible. Once the child can drag and drop on an iPad, they would be able to play Kodable. The idea of the game is to move a fuzzy ball through a maze by dragging and dropping arrows. Kodable has no written instructions, but does have voice controlled instructions.

The participants were allowed to explore Kodable without any guide. Initially, the concept of dragging and dropping was foreign to them, as none of them had used an iPad before. However, once they mastered the ability to use the iPad, the children were able to work through the basic levels of the game.

The participants were also made to test the programming syntax they had learned with the maze, on Kodable. This they did by verbally forming out inputs before performing the Kodable levels. This they did successfully.



Kodable and the tool developed for this project are similar in using arrows or directions to teach programming as a process that is based on a step-by-step process. Kodable has a more lively and fun interface and also sound effects. However, Kodable does not introduce users to any kind of syntax whatsoever.

While it helps them think computationally and builds a certain level of computational analysis and sequential processing; the children do not precisely learn syntax that would help them code properly.

The combination of graphical programming and understandable syntax is a worthwhile introduction to programming.

4.8 Future Testing

Once the application is fully developed, testing should involve using more complex concepts of programming such as functions and 'if and else' statements.

4.9 Product Testing

Some of the participants requested that the maze be coloured differently from just black. The children also requested that sound effects be included in the game. One reason they were immediately drawn to Kodable was the more lively animation and sound effect it supports. Rather than a ball moving in response to a command, in Kodable, a fuzzy ball with a face rolls along the maze. Kodable also has sound effects for movement, errors and correct commands.

Testing has to be performed on more than one kind of Java-enabled phone.

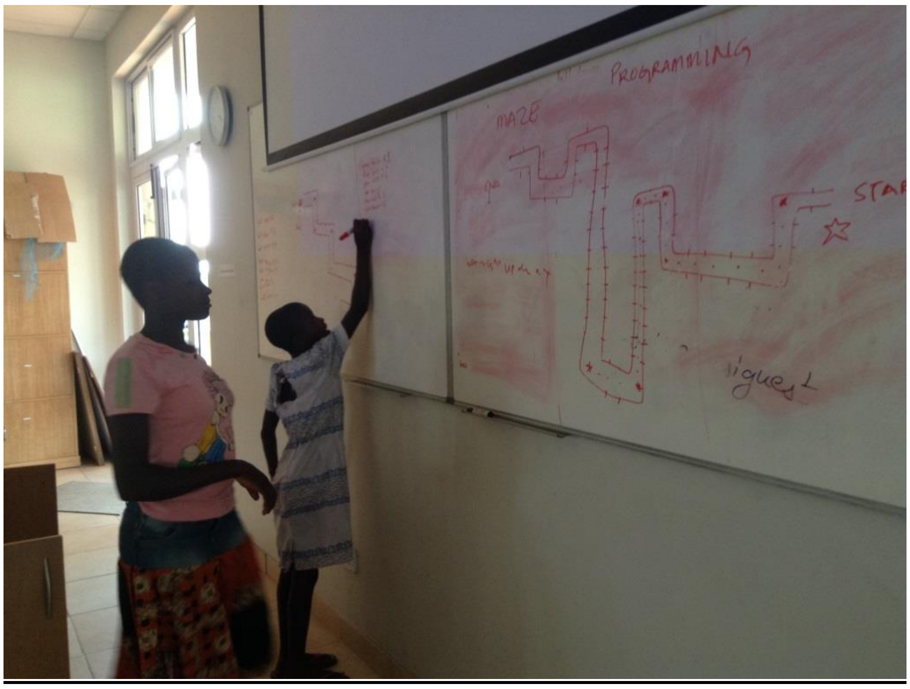
4.10 Observations and Results

Testing the maze and the programming language developed around it, helped come up with very important conclusions about teaching children how to program. For the first maze, the children were able to navigate the ball easily, from top to bottom, using the syntax that had been developed during the first testing session. During the process of the quiz though, all the children sat at a distance from the board and attempted to gauge the number of steps the ball would have to take through a given direction in the maze.

The second maze, which was inverted' however, tested their ability to apply the knowledge acquired from the previous day to a different kind of problem. As stated earlier, four out of the six children were able to navigate the maze correctly. Feedback from the two participants, who were unable to navigate the maze properly, showed that they had tried to remember the nature of the syntax from the previous maze, and also from the previous sessions. This is indicative of the style of education employed in Ghana, rote memorization. However, programming, or teaching programming should deal with teaching the student how to use algorithms for different problems, and also how to think creatively about solving problems.

Another issue that arose from the tests they wrote was the fact that they did them on paper, and had to work with the maze drawn on the board. The absence of feedback, and having to just think and gauge about the movement, rather than have a visual representation of their progress was an impediment. For the second test however, they were allowed to stand at the whiteboard to analyse the maze based on their 'predictions' or based on their syntax. They made markings on the board and tried to guess the various positions the ball had to be in. In this instance an increased level of interaction benefited them.

Thus being able to immediately visualise or receive feedback from an instruction they make or a command they write helps them better apply their knowledge on programming. Additionally, when the syntax involved resembles words they use in their everyday lives whose meanings or results are just the same as in everyday life, they are able to grasp the concept more easily.



CHAPTER FIVE

Conclusion and Recommendation

This chapter evaluates the results from the testing phase and outlines recommendations for further development and testing.

When basic syntax is combined effectively with graphical output, children are able to pick up programming relatively easily. Additionally, when the graphical representation is an immediate feedback of their input, it helps them immediately appreciate the results of their input.

The tool presented in this dissertation provides a foundation for children in programming. To navigate the maze, or solve a problem, for that matter, the application was one the participants could easily identify with. Using a maze, a concept that both challenges children and breeds healthy competition seemed welcoming to them. The ball and "Welcome to Berekuso" statement at the completion of a level, are concepts they can easily identify with. Additionally, the syntax was developed by observing their interaction and making it as close to their normal language as possible. Directions such as left, right, up and down are also basic everyday conversational words that the children are used to.

While this tool proves to be useful for upper primary school children, it is important to think about how tools can be scaled up for children of higher levels, and also scaled down for children at lower levels. At what age should children be able to start coding in regular programming languages such as Java, C++ and Python? Another school of thought that subscribes to the notion that "if a child cannot learn the way we teach, maybe we should teach the way learn” can be stretched into the concept of re-structuring programming such that it is able to accommodate different learning styles. Should programming also be about learning how computers work so we can communicate effectively with them, or should computers be made to learn how humans think such that they can understand us more effectively and perform instructions more intuitively?

For the higher levels, more advanced concepts of programming such as functions and loops can be implemented also with the maze. The mobile phone can be scaled up to allow for other subjects. Not only is the phone useful for programming, but also for other subjects that help provide access to learning for the rural children.

In conclusion, the tool, the interactive maze game and the syntax developed provided a good combination for teaching children how to program at a basic level.

Bibliography

[1] L. Orsini, "ReadWrite," 31 May 2013. [Online]. Available: http://readwrite.com/2013/05/31/programming-core-skill-21st-century#awesm=~onCjbLXrjlT3Nz. [Accessed 19 November 2013].

[2] M. Resnick, Y. Ohshima, M. Flanagan, K. Perlin, C. Kelleher, R. Torres and M. MacLaurin, "Growing Up Programming; Democratizing the Creation of Dynamic, Interactive Media," in *CHI*, Boston, 2005.

[3] M. M. J. H. A. M. Resnick, N. Ruck, E. Eastmond, K. Brennan, A. Millner, E. Rosebaum, J. Silver and B. K. Y. Silverman, "Scratch: Programming for Everyone".*Communications of the ACM.*

[4] S. Papert, "Teaching Children Thinking," in *The Computer in School: Tutor, Tool, Tutee*, New York, Teacher's College Press, 1980, pp. 353 - 365.

[5] Scratch, "About Scratch," [Online]. Available: http://scratch.mit.edu/about/. [Accessed 4 April 2014].

[6] S. Curtis, "Technology," Telegraph, 4 November 2013. [Online]. Available: http://www.telegraph.co.uk/technology/news/10410036/Teaching-our-children-to-code-a-quiet-revolution.html. [Accessed 4 April 2014].

[7] P. Olson, 6 September 2012. [Online]. Available: http://www.forbes.com/sites/parmyolson/2012/09/06/why-estonia-has-started-teaching-its-first-graders-to-code/.

[8] GSM Association, "Public Policy," December 2011. [Online]. Available: http://www.gsma.com/publicpolicy/wp-content/uploads/2012/04/africamobileobservatory2011-1.pdf.

[9] National Communications Authority , "News," February 2014. [Online]. Available: http://www.nca.org.gh/73/34/News.html?item=364. [Accessed 8 April 2014].

[10] Gyasi-Agyei, "ICT Talking Points with Prof. Gyasi," 6 May 2013. [Online]. Available: http://www.profgyasi.gictrac.org/documents/6May13_BFT_MobilePenetrationMNP.pdf. [Accessed 13 May 2013].

[11] Z. Issah, "Myjoyonline," 13 February 2013. [Online]. Available: http://myghanaonline.com/?id=1.1269900. [Accessed 19 November 2013].

[12] "Oxford Dictionary," [Online]. Available: http://www.oxforddictionaries.com/definition/english/smartphone?q=smartphone. [Accessed 8 April 2014].

[13] Y. Miyashita, "Evolution of Mobile Handsets and the Impact of Smartphones," February 2012.

[Online]. Available: http://www.icr.co.jp/docs/Evolution_of_Mobile_Handsets_and_the_Impact_of_Smartphones.pdf.

[14] UNESCO, "ICT in Education," [Online]. Available: http://www.unesco.org/new/en/unesco/themes/icts/m4ed/. [Accessed 7 April 2014].

[15] Ghana Education Services, "How do I find," 2012. [Online]. Available: http://www.ges.gov.gh/?q=content/basic-education-curriculum. [Accessed 1 April 2014].