Programming for Everyone: A Visual Programming Toolkit

By

Mufti Anees-Ur-Rahman

00000317890 NS

Supervisor

Brig (Retd) Associate Professor Dr. Fahim Arif

A thesis submitted to the faculty of Department of Computer Software Engineering, Military College of Signals, National University of Sciences and Technology (NUST), Rawalpindi in partial fulfillment of the requirements for the degree of MS in Computer Software Engineering

August 2022

Declaration

I, *Mufti Anees-Ur-Rahman* declare that this thesis titled “Programming for Everyone: A Visual Programming Toolkit” and the work presented in it are my own and has been generated by me as a result of my own original research.

I confirm that:

1. This work was done wholly or mainly while in candidature for a Master of Science degree at NUST
2. Where any part of this thesis has previously been submitted for a degree or any other qualification at NUST or any other institution, this has been clearly stated
3. Where I have consulted the published work of others, this is always clearly attributed
4. Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work
5. I have acknowledged all main sources of help
6. Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself

Mufti Anees-Ur-Rahman,
00000317890 NS

Copyright Notice

- Copyright in text of this thesis rests with the student author. Copies (by any process) either in full, or of extracts, may be made only in accordance with instructions given by the author and lodged in the Library of MCS, NUST. Details may be obtained by the Librarian. This page must form part of any such copies made. Further copies (by any process) may not be made without the permission (in writing) of the author.
- The ownership of any intellectual property rights which may be described in this thesis is vested in MCS, NUST, subject to any prior agreement to the contrary, and may not be made available for use by third parties without the written permission of MCS, which will prescribe the terms and conditions of any such agreement.
- Further information on the conditions under which disclosures and exploitation may take place is available from the Library of MCS, NUST, Islamabad.

This thesis is dedicated to *my beloved parents*

Abstract

Computer Programming is the backbone of the software industry. This industry is evolving and progressing minute by minute and is arguably one of the most important and impactful trends of the coming years. As the demand is growing day by day, the number of newcomers in this field are also growing rapidly. That is where the majority of the newcomers of the industry face their first problem, which is, that programming in the start is challenging to learn. The concepts, syntaxes and the general "flow" of the code are not easy to understand in the first few days. A lot of new students get discouraged from this thinking that this is too difficult of a subject or field for them to pursue while only a few stay consistent and come out as computer programmers. Even then there are a lot of students who are just average in computer programming while only a few are good. This led to the making of this framework which helps the "onboarding" process for the new commers and essentially aims to reduce the number of students who quit programming while also decreasing the time required by them to understanding the concepts of programming.

This Thesis presents a framework that by utilizing the ease of Visual Programming Languages and the fine control of Textual Programming Languages an idea that will not only help newcomers in programming by making their environment less intimidating but also help them learn quicker as concepts are understood a lot better while presented visually. Both blocked-based programming and traditional textual programming concepts are used in this to help students, in the beginning, understand the structuring of how code works by showing them a general overview of the code and its syntax. Then slowly we move the student into textual programming by replacing blocks one by one to minimize the difficulty of learning all code at once and maximizing their understandability of the syntax as they do not have to learn all the programming concepts at once, which is how traditional teaching methods usually approach this.

The results of the presented framework are very positive. The results are a combination of student feedback and student progression regarding computer programming. The sample space of the students consisted of 300 plus students in order to get conclusive results.

Keywords: *Visual Programming Languages, Block-Based Programming, Teaching Programming languages, Visual and Textual Based Programming languages*

Acknowledgments

All praises to Allah for the strengths and His blessing in completing this thesis. I would like to convey my gratitude to my supervisor Brig. Fahim Arif, PhD for his supervision and constant support. His invaluable help of constructive comments and suggestions throughout the experimental and thesis works are major contributions to the success of this research. Also, I would thank my committee members; Dr. Yawar Abbas and Brig Adnan Ahmad Khan,PhD for their support and knowledge regarding this topic.

Also last, but not the least, I am highly thankful to my parents and my Siblibgs. They have always stood by my dreams and aspirations and have been a great source of inspiration for me. I would like to thank them for all their care, love and support through my times of stress and excitement.

Contents

List of Figures

List of Tables

List of Abbreviations and Symbols

Abbreviations

VPL	Visual Programming Language.
TPL	Textual Programming Language.
IDE	Integrated Development Environment.
VI	Visual Instrument
UI	User Interface

Chapter 1

Introduction

1.1 Overview

Programming is an important and ever-growing field in modern times and estimates of its use in the future are only showing that it will be increasing in importance in the near and far future. With the growing importance of programming, the demand of Programmers and Application developers is increasing day by day. However, Programming and its concepts are a big hurdle in front of people who are just getting into computer programming or coding. Teaching computer programming to students is a challenge for a lot of teachers as most of them lack prior experience with this. Furthermore, the students also face a new problem that is understanding the concepts of computer programming or to understand how the computer “thinks”. Visual computer programming languages can help remove a lot of these difficulties as the students are not expected to start coding on day one rather, they are presented with a visually appealing and less intimidating environment. However, this does not completely negate the traditional text-based computer programming as ultimately that is what must be taught to the students and that is what the companies are hiring for.

1.2 Motivation

The motivation for this research to be carried out was the need felt to have a better and more efficient way of communicating with students or newcomers of how the computer programming languages function. It is very common to see students struggling in the beginning of their learning period of computer programming languages with the information, which is provided to them get very overwhelming, however at the same time the information is crucial to be provided as coding is interlinked with itself in such a way that it is not possible to skip some things to be taught later.

The main focus of this research is to evaluate the impact this has on the beginner students or newcomers in learning computer programming. As it can be deterred, this is a very vast field upon which the influence of this research can benefit users.

1.3 Scope

The scope of this research is very vast as it can be used to help any person who wants to start learning computer programming languages. Even though the main target of this research is students, any person who wishes to learn computer programming Languages can benefit from this.

However, our main focus for now are students who wish to learn computer programming Languages and this method can be applied at any level of academic studies to help students gain the knowledge required of computer programming languages easier.

1.4 Problem Statement

It is observed that to boost the speed, concepts, and understanding of computer programming, students need a visual representation of what they are working on, it can be either physical like circuits of a robot or an Arduino board, etc or it can be in the software which shows the overview of the code like scratch, Google Blockly, etc. However, the majority of companies where these students will join for jobs do not accept visual computer programming languages as computer programming skills like Scratch or Blockly. But by using this method to incorporate blocked-based computer programming to teach students text-based computer programming, the benefits of which are better understandability, ease of learning, etc. This in turn will not help students in learning faster but also prepare them for their careers as text-based programmers in companies, software houses, and firms.

1.5 Thesis Breakdown

The research has been done in a lot of different phases and steps. It is divided into the following chapters:

Chapter 1 Introduction: An Overview of Visual programming languages.

Chapter 2 Literature Review: Discussion and highlighting of work already carried out on this topic by other people.

Chapter 3 Proposed Framework: The explanation of the framework proposed to overcome the problems which are observed.

Chapter 4 Experimentation and Results: Testing the validity of the framework by creating a survey and viewing its outcomes.

Chapter 5 Analysis and Discussion: details of how the framework performs and the impact it had on the students.

Chapter 6 Conclusion: This section provides a recap of all the work done and also shows a direction for the future of this research.

1.6 Summary

In this chapter, the details of the motivation, scope of the project, introduction and the problem statement were presented. The main reason for the motivations was the difficulty faced by students whilst learning computer Programming languages. This is a very common concern for teachers and has been for quite a long time. The scope of this project is very vast as it encompasses not just students but anyone who is learning computer programming languages. However, we will focus on students in this paper and how to improve their knowledge grasping capabilities. This chapter provided the introduction to the research practices which were adopted in order to get informatics for the students and if the methodology applied to them was beneficial for them and the teachers.

CHAPTER 2

Literature Review

2.1 Introduction

A literature review is a comprehensive summary of previous research on a certain topic. A literature review looks at scholarly publications, books, and other sources that are relevant to a certain research topic. In the review, this previous work should be listed, detailed, summarised, objectively appraised, and clarified. It should serve as a theoretical framework for the study and aid the author in selecting the scope of the investigation. The literature review acknowledges the contributions of previous researchers, ensuring the reader that your work is well-considered. It is assumed that the author has read, assessed, and assimilated a previous work in the subject of study by mentioning it in the current work.
A literature review gives the reader a "map" of the field's progress, allowing them to fully appreciate it. The author has included all (or the vast majority) of earlier, noteworthy publications in the area into her or his research, as seen in this diagram.
Keeping this in mind, this chapter presents the work of other authors on this specific topic, as well as a conclusive and accurate conclusion based on their work.

A literature review provides the reader with a "landscape," allowing them to fully comprehend the field's advances. This diagram shows that the author has incorporated all (or the vast majority) of earlier, significant publications in the topic into her or his research.

Keeping that in mind this chapter provides the work carried out by other authors on this specific topic and by using their work a conclusive and accurate.

2.2 Related Work

A group of scientists developed a teaching method [1] that consisted of free software and low-cost electronic components to make a robot. Its main use was to increase the confidence and performance of students in engineering areas to improve circuitry design and programming. The results of this showed that this activity promoted an increase in the knowledge and helped students develop their skills with the help of visually looking at the code using the robot's circuitry which in term helped them understand programming concepts a lot quicker and easier than text-based programming.

A group of scientists carried out research in [2] which they took 105 students from the engineering department who study programming. This research showed that the students have the most difficulty understanding the abstract topics and understanding the basic concept of programming and designing programs. In the end, the authors proposed a visualization tool as an alternative to learning programming which was also agreed upon by the students.

A few scientists proposed a recommendation-based system [3]. The system consisted of Augmented Reality technology and learning theory.

Several students were taken and split into 2 groups. One group used Augmented Reality with deep learning recommendations while the other used Augmented Reality only. This showed that the students who were using Augmented Reality and the recommendation system performed a lot better regarding learning achievements while the other group performed better at computational thinking ability. Overall, in all aspects of programming, the students with both Augmented Reality and the recommendation system performed a lot better in comparison with the other group.

Multiple scientists took 347 students, and they were given an Arduino board with its custom application programming interface (API) [4]. Since the students had a visual way to "look" at their code and it does use the physical Arduino board they had a quicker grasp on programming and 75% of the students said that they want to continue to learn programming with the use of Arduino and its API.

Scientists found out that visual programming and game-based learning enhances the computational thinking and problem-solving skills [5].

2.3 Programming Languages

There are 2 types of programming languages:

- Textual Programming languages (TPLs)
- Visual Programming languages (VPLs)

2.3.1 Textual Programming languages (TPL)

The other type of computer programming languages are referred to as Textual Programming languages. These are the computer programming languages with most of the people in the world are familiar with as these are the ones usually studied in a school. They are further divided into two more categories namely:

- low level programming languages
- High level programming languages

2.3.1.1 Low Level Programamaing:

Low-level programming languages are computer programming languages that are written in a symbolic style. The Symbolic format is characterised by the use of hardware-specific mnemonics.
Mnemonics such as ADD, SUB, MOV, jmp, and others are used in languages such as Assembly. Low-Level Languages are the name given to these languages. Low-level languages are also difficult to learn and comprehend.
Hardware is also a factor in low-level languages. You can't use the code you wrote for one hardware chipset with another. As a result, you'll need to rewrite your application for the new hardware. Low-level programming languages are not portable.
The majority of modern computer languages, such as Python, Swift, and others, are classified as high level languages because they resemble the English language. These high-level languages are simple to learn and master, and they are also portable.
Key Characteristics:
Sub, add, mov, jmp, retq, popq, and other mnemonics are used to represent low-level programming languages. Maintaining, writing, and debugging low-level code is more

difficult than maintaining, writing, and debugging high-level languages code.
Because low-level languages are hardware-dependent, we can't reuse code written for one hardware platform on another. As a result, these dialects are not transferable. The best example of a low-level programming language is assembly.

2.3.1.2 High Level Programaming:

High level programming languages are usually more widely used and are the ones that are taught to the students more frequently than low level languages. High level languages are changed into low level languages in order to be understood by the computer however this process is very seamless and the user does not need to know low level languages at all while working with higher level languages. High level languages are easier to understand by the user compared to lower-level languages as they have a lot of words from the real-world languages that humans speak, for example English. A lot of people who have no prior experience with coding can understand what high level programming languages are doing if presented with code someone else wrote.

2.3.2 Visual Programming Languages (VPL):

In recent years, programming has become more accessible than ever. As the world awakens to the importance of computer programming, software engineers have developed several tool kits to make computer programming skill acquisition easier regardless of your background or environment. Visual computer programming languages (VPLs) can provide neophytes with a set of systems to learn how to build applications graphically.
A VPL (sometimes called a graphical computer programming language) lets you write computer and web applications using visual tools like boxes or images. Logical connections are usually displayed in the form of lines or wires. All of this allows you to conceptualize and map out a project's operations from data input to output in a way that is intuitive.
Scratch is arguably the best most recognizable VPL. It is a language primarily geared towards helping children. It helps them learn how to code in a gamified environment.
The simple graphical nature of tools like Scratch makes them ideal for building systems that teach people how to code. It primes novice software engineers and enthusiasts

to think like programmers. Additionally, they have unique characteristics that could minimize many of the common difficulties that new software developers encounter.

2.4 Translating Assembly language into Machine Language:

The low level language code is close to hardware mnemonics but still it need to convert into the Machine Level Code or Binary code so that computer processor (CPU) will understand it. So, We need to translate the assembly code to Machine code which is done by Assembler.

2.4.1 Example Program

This is an example of a simple addition program in C computer programming language to Add two numbers. It is presented in Both High level language and low level language to highlight the contrast in between these two different computer programming Language types.

2.4.1.1 High Level Programming Language

High Level Language Code. The Language used is C++.

```
#include<stdio.h>
int main()
{
int a = 10, b = 20;
int sum = a + b;
printf("Sum of %d and %d is %d \n", a, b, sum);
return 0;
}
```

2.4.1.2 Low Level Programming Language

Code that is the same as above but written in Assembly Language.

```
.section __TEXT,__text,regular,pure_instructions
```

```
.build_version macos, 10, 15 sdk_version 10, 15, 6
.globl _main                  ## -- Begin function main
.p2align 4, 0x90
_main:                                 ## @main
.cfi_startproc
## %bb.0:
pushq %rbp
.cfi_def_cfa_offset 16
.cfi_offset %rbp, -16
movq %rsp, %rbp
.cfi_def_cfa_register %rbp
subq $32, %rsp
movl $0, -4(%rbp)
movl $10, -8(%rbp)
movl $20, -12(%rbp)
movl -8(%rbp), %eax
addl -12(%rbp), %eax
movl %eax, -16(%rbp)
movl -8(%rbp), %esi
movl -12(%rbp), %edx
movl -16(%rbp), %ecx
leaq L_.str(%rip), %rdi
movb $0, %al
callq _printf
xorl %ecx, %ecx
movl %eax, -20(%rbp)          ## 4-byte Spill
movl %ecx, %eax
addq $32, %rsp
popq %rbp
retq
.cfi_endproc
## -- End function
.section __TEXT,__cstring,cstring_literals
```

```
L_.str:                                  ## @.str
.asciz "Sum of %d and %d is %d \n"
.subsections_via_symbols
```

2.5 Challenges of Learning computer programming

A list of commonly faced issues by people learning computer programming Languages are listed below:

Lack of Technical Background

For the budding developer and enthusiast, learning how to program may be the first experience with any technical subject. This could make getting started quite daunting. Although an individual's creativity can be repurposed to certain aspects of app building, the exacting and formal elements and logic of coding can be hard to grasp for beginners.

Difficulty

Today, if a person wants to learn how to code, they will have to wrap their heads around tough concepts, build critical thinking skills. In other words, learn to think in an unusual way like a software developer or how a computer does.

Programming requires a thought process that may feel quite unusual to new commers. this is something that becomes natural with practice. However, that does not make things any easier when a beginner is just starting out.

In an ideal world, students with diverse backgrounds would simply advance at their own pace. But in the classroom, it presents a significant challenge to the instructor, who must deliver the same course to all the students despite their differing needs.

2.6 How VPLs Help

Visual computer programming languages and tools provide easy-to-use editors that address most of the problems discussed above. Most people find concrete images easier to grasp than abstract concepts or formal syntax, and VPLs include a list of commonly used objects and actions that you can immediately insert into your program.

When working with text-based code, people start off with a blank code editor and must conceptualize how to represent the program starting with a line of code. This can be particularly intimidating for a novice user.

A visual-based computer programming language removes this barrier by presenting systems of objects that can be dragged and dropped to create working “code” right away. Choosing from a list of predefined elements is simpler than conceiving and writing an entire program from scratch. The graphical interfaces of visual computer programming tools make it easy to go from a general idea in someone’s head to a working program.

This ease of use is a defining feature of visual-based software development. It makes software development much more approachable for those who are anxious about learning, as well as beginners who have grown frustrated after grappling with a more standard language.

The simplicity of getting started means that a user can still focus on solving problems or building applications without the overhead of learning syntax and complicated computer software development constructs before they are fully able to understand them. Additionally, having learners on a more level playing field also benefits teachers, who can focus more on instruction and less on students’ diverse backgrounds.

While a VPL can still be great for teaching new software developers, they are useful for experienced developers and other professionals as well. They are commonly used by domain experts in various fields who need to get some computer programming work done as easily as possible.

Researchers and practitioners without any existing coding skills can benefit from the ability to create functional web and desktop programs for computers (and other devices/hardware) without a huge time investment upfront.

2.7 Popular Visual Programming Languages

Several visual computer programming tools are now widely used by people across several industries and for a variety of different purposes. Here is a sampling of some notable ones.

- As mentioned, the top-known example is MIT’s Scratch, which allows kids to create fun projects for computers and share them with the rest of the community. As of this writing, over 50 million projects have been made by Scratch users.

- App Inventor, also managed by MIT, makes it easy to create apps for Android (and soon iOS) devices through a simple drag-and-drop interface highly reminiscent of the one used by Scratch.

- miniBloq is utilized to write programs for Arduino microcontrollers through a set of blocks. It has been used to teach computer programming at the elementary level in Argentina, reaching over 60,000 students in one province alone. Projects made in miniBloq produce real C/C++ code used by Arduino.
- LabVIEW is a tool that is widely used by scientists and engineers for simulations, measurements, number crunching, and other common tasks. It utilizes a clean user interface that is powerful enough for experienced developers to create complex desktop and web programs for computers and other hardware. However, it is accessible to domain experts with no computer programming background.
- Bubble is a basic visual development environment that makes it possible to create web applications through a visual interface. Unlike some other drag-and-drop website creators, Bubble allows users to customize the behaviour of their apps in some complex ways while maintaining type and/or class control.

2.7.1 Scratch

Scratch is a free computer programming language that teaches newcomers to code in a fun and visual way. It's a fun-focused computer programming tool intended toward children as young as eight years old, but older people may benefit from it as well, and many people use it to create apps and games. It's a great way for teachers to get students interested in coding and computer programming.

Students can use block-based code to create animations and pictures, which can then be shared once the project is finished. This makes it ideal for distance learning, as teachers can give projects for students to do and communicate.

Scratch is named after DJs who mix records, and it allows users to use a block code-based interface to mix projects like animations, video games, and more with sounds and graphics.

The platform, which was developed by the MIT Media Lab, is available in more than 70 languages worldwide. At the time of publication, Scratch had roughly 67 million projects shared by over 64 million users. With 38 million monthly visitors, the website

is especially popular for learning to work with block-based coding.
It was first released in 2007 and has gone through two updates since then, transitioning from the Squeak coding language to ActionScript and then to JavaScript.
Scratch coding abilities may be valuable in future coding and computer programming courses and job opportunities. To be clear, this is a block-based system, which means it's easy to use and requires pupils to string together pre-written commands to do operations. It is, nevertheless, an excellent location to begin learning.

Functionality

Scratch 3.0 is separated into three sections: a stage, a block palette, and a code area, which is the most recent version at the time of writing.
On the stage, the outputs, such as an animated video, are presented. All of the commands that may be dragged and dropped into the project via the coding area are found in the block palette.
A sprite character can be chosen, and orders from the block palette area moved into the code area, allowing the sprite to do the activities. A cat cartoon, for example, may be made to advance 10 steps.
It's a very basic form of coding that emphasises the action event-based coding process rather than the complex language itself. Scratch, on the other hand, may be used with a broad range of real-world projects, such the LEGO Mindstorms EV3 and the BBC Micro:bit, expanding the coding platform's capabilities.

Appeals

Scratch's major selling point is how simple it is to use. Students may easily achieve a pleasant and engaging result, encouraging future use and deeper understanding of coding.
The online community is another valuable feature. Because Scratch is so widely used, there are several opportunities for interaction. Members of the site can write comments, tag, like, and share other people's creations. Challenges in the Scratch Design Studio are popular, and they encourage students to compete.
Educators have their own ScratchEd community where they can share stories and information, as well as exchange materials and ask questions. This is an excellent method for coming up with new project ideas. For easier control and direct commentary, a Scratch

Teacher Account can be used to create accounts for students. You must request one of these accounts directly from Scratch in order to get one.
You may use Scratch to control physical world devices like LEGO robots, as well as digital musical instruments, video motion detection with a camera, text to speech conversion, Google Translate translation, and much more.

2.7.2 Blockly

Understanding computer programming languages is required to create a website that responds to user inputs. And anyone working on an app will almost always need to use a higher-level programming language. Despite the fact that there are several resources for learning Python, JavaScript, and other similar languages, understanding the complicated commands and routines requires a significant amount of effort. Conventional tools are frequently too tough to comprehend fast for those who just wish to try their hand at computer programming or only need a short script.
This is where Blockly comes into play: Blockly is a Google project that uses graphic blocks to illustrate large text-based code sequences. These blocks can be placed together using drag and drop. This allows you to create a complex syntax in only a few minutes. Blockly is a code editor that features a graphical user interface. The syntax of the programmes created in this way is kept secret. Anyone who utilises Blockly can easily construct complex processes without having to learn each computer programming language's specialised commands.
There are numerous benefits to using Blockly. For example, with this library, constructing a responsive website without in-depth computer programming experience is easy. Professional software developers, on the other hand, can use the visual code editor to swiftly design small programs.
When computer programming, the light-hearted approach allows you to easily comprehend relationships. Even though they both express the same thing, "repeat 5 times" is easier to grasp than "for int I = 0, I 5, i++;." As a result, Blockly is frequently utilised in educational settings. If your youngster is interested in learning computer programming, Blockly can be a terrific method for them to take their initial steps in software development. The visual presentation of code sets makes achieving amazing results quick and

straightforward.

The code is clean and easy to grasp because to the visual interface. Blocks are arranged in a puzzle-like fashion and then converted into code. Despite the fact that Blockly is a JavaScript library, the source text can be directly translated into a variety of languages. JavaScript, Python, PHP, Lua, Dart, and XML are all supported by the code. In a browser, you may easily access the editor.

Visual computer programming isn't a new concept anymore. It's already being used by a lot of people to make websites. The intuitive website kits, like Blockly, thrive with their simple and intuitive UI. The written code runs in the background, allowing beginners and non-techies to get up and running quickly.

Scratch and Blockly are quite similar. Scratch is primarily meant as a teaching resource, whereas Blockly is geared more at professional developers in a corporate context. However the basic concept remains the same.Visual computer programming is no longer a novel concept. Many individuals are already using it to create websites. Blockly, for example, thrives because of its basic and intuitive user interface. Beginners and non-techies may get up and running quickly because the written code runs in the background.

Functions

The Blockly app – the visual code editor – offers eight categories with different functions:

- Actions are described by logic.
- Loops are control structures that repeat themselves until a specific action is taken.
- Math can do a variety of calculations and generate random numbers.
- Text can access inputs and generate unique outputs.
- Lists combine text and numeric elements to form lists.
- You can use colour to change the colour of the text or the background.
- Variables can be used in functions and calculations.
- When a specific input is recognised, functions explain how the website should behave.

Blockly Example

The blocks are really easy to utilise. Simply open a category and drag the necessary code block to the working area using the mouse. Individual blocks can be adjusted and combined with others to create new combinations. If no combination is possible, the block will not snap into place and will not be included in the source text. Individual puzzle pieces or groups of related blocks can be easily removed from the working area and re-added. The matching lines of code in the source text are removed in this case.

Where is Blockly used?

The library is intended for programmers, whereas the app is geared toward students and beginners. Developers can construct their own apps using the library by designing their own blocks with functions.

There are two ways to make your own blocks in Blockly. The JavaScript API, which is widely used in online applications, is the first. Second, an Android and iOS JSON interface is supplied. Only the most popular blocks are available in JSON format. Additionally, there is considerable documentation on GitHub. A full guide to building and utilising Blockly is also available from Google.

On YouTube, Stack Overflow, and GitHub, there are already a plethora of videos and articles explaining how to use the library efficiently. Programming various switches and receivers for smart home control is one of the many options. Because the programmes can be linked to a variety of languages, your creativity is virtually unbounded.

2.7.3 LabVIEW

The first graphical computer programming environment was LabVIEW (Laboratory Virtual Instrument Engineering Workbench), and it is still the most widely used graphical computer programming environment today. It provides a powerful and integrated environment for developing a variety of instrumental applications. Code, data, block diagrams, front panels, and GUI changes take up the great majority of time in an efficient LabVIEW program, which is free of extraneous tasks.

It lowers the frequency of data collection and processing errors made by humans. It reduces data entry errors, and having more reliable data allows for better product quality management and new discoveries. LabVIEW programmes are frequently referred to as virtual instruments because of their appearance and operation (VIs). It comes with a comprehensive set of VIs and functions for data collecting, analysis, presentation, and storing, as well as troubleshooting tools. It also includes features for using the LabVIEW

Web Server to link user applications to the Internet.
It is used to manage large and professional applications and includes integrated project management tools, integrated graphical debugging tools, and standardised source code control integration. LabVIEW is a free open-source development environment that comes with all of the tools you'll need to complete most projects.

Advantages of LabVIEW:
Some advantages of this technique over Text-based Programming are :

- When compared to text-based programming, graphical programming is far more interactive.
- In text-based programming, the syntax must be understood, while it is not essential in graphical programming.
- Text-based programming necessitates additional coding, but Graphical programming necessitates no further coding.
- In graphical programming, errors are shown by wire blocks, whereas in text-based programming, the program must be compiled to check for errors.

Features of LabVIEW:
Some other features of this are:

- User-friendly UI: It offers a drag-and-drop interactive User Interface that is easy to use.
- Built-in Functions: It has thousands of built-in functions, such as analysis and I/O. This is a part of the function palette.
- Scalable: LabVIEW's modular design allows for easy scaling and modulation of programmes.
- Professional Development Tools: It comes with several tools that assist in the integration and debugging of huge applications.
- Open environment: It contains the tools required for many open-source projects.
- Object–oriented design: It supports object–oriented programming features like as encapsulation and inheritance, allowing for modular and extendable programmes.

- Compiled language: It is faster since it is compiled.

Role of Components:

- When compared to a text-based programming environment, LabVIEW is a graphical programming environment that allows users to develop and evaluate any sophisticated system in less time.
- Virtual instruments are the graphical applications created with LabVIEW.
- The block or graphical component is performed when data is available at all inputs.
- The data is delivered to output terminals once the execution is completed, and then it is transmitted to the next block in the dataflow path.

Front Panel:
Users can interact with the VI through the front panel, which displays outputs and allows them to send inputs to the application.
Controls and Indicators:
The controls act as input devices for the VI's block diagram and deliver data to it. Controls such as knobs, pushbuttons, dials, and other input devices are commonplace. The indicators serve as output devices, presenting data acquired or generated by the block diagram. Popular indicators include graphs, Light Emitting Diodes (LEDs), metres, and other output devices.
Back Panel:

- The rear panel contains the VI's code for taking inputs from the front panel, processing them, and displaying the results.
- The back panel is sometimes known as a block diagram.
- To operate software, a block diagram employs graphical code.
- The code is added to the block diagram using a graphical representation of the functions that control the front panel items.

- The structures and functions that operate the controls and send data to the indicators are housed on the back panel.

The three different **palettes** available in LabVIEW are :

1. Tool Palette
2. Controls Palette
3. Function Palette

Tool Palette

- A mouse pointer operating mode is referred to as a tool. The cursor corresponds to the icon of the tool chosen in the Tools palette.
- Tool Palette allows users to build, modify, and debug Virtual Instruments.
- The tool palette is accessible from both the front panel and the Block Diagram.

The following are some of the tools available:

- Select text or change control settings with the operating tool button.
- Resize, select, and position objects with the Positioning tool.
- Labeling Tool allows users to customise text and create free labels.
- Quick access to an object Opens the shortcut menu for the selected object with the Menu Tool.

Controls Palette
Only the Front panel has access to the Controls Palette.
It has a variety of controls and indicators that the user will need when constructing the front panel.
Function Palette
It can only be found on the Block Diagram and is used to create Block Diagrams.
The following are examples of different function palettes:

- Numeric
- Array
- Time and Dialogue
- Waveform

LabVIEW includes libraries for connecting stand-alone instruments, data gathering devices, motion control, and vision systems.

2.8 Block-based Programming

Block-based computer programming systems are a useful tool because they present programmers with a colourful, easy-to-use drag and drop facility for coding. Users choose from color-coded sets to "snap" together and create a program. A lot of various kinds of code blocks exist to facilitate the user like Movement blocks, control or events blocks, blocks for adding loops (iteration), variables, and functions. These languages are so similar in structure that a student can jump from one block-based language to another effortlessly.

Block based Programming Languages

Table 2.1: Famous Block-based Programming Languages

Program	Description	Platform	Output
BYOB/Snap!	Snap! is a powerful Scratch modification that includes lambdas, first-class data, procedures, recursion, and a lot more. Snap! was renamed and redone in JavaScript,hence version 4.0 is no longer considered a Scratch adaptation.	Desktop, Mobile	Desktop, Mobile

Table 2.1: Famous Block-based Programming Languages

Program	**Description**	**Platform**	**Output**
Blockly	Google created a drag-and-drop language. It can be found on a lot of websites. It instantly translates a Scratch-like language to another text-based language. This could come in handy while studying more traditional languages like JavaScript or Python. This is how Scratch 3.0 works.	Desktop, Mobile	Desktop, Mobile
Android App Inventor	Google's Android App Inventor lets you design Android apps using a basic, Scratch-like interface. In fact, it was created by an MIT team using Scratch. MIT.	Desktop	Android
Stencyl	Stencyl has a similar interface to Scratch, although it offers a few more complex editing options.	Desktop, Mobile	Desktop, Mobile
Gamefroot	A tool for designing side-scrolling games that may be played online. For complex scripting, it offers a drag-and-drop block editor.	Desktop, Mobile	Desktop, Mobile

Table 2.1: Famous Block-based Programming Languages

Program	**Description**	**Platform**	**Output**
Pocket Code	A visual programming language and app for smartphones, tablets, and HTML5-enabled mobile browsers on Android, iOS, and Windows Phone. It is based on Scratch and produced as free open-source software by the Catrobat team.	Android	Android
Hopscotch	An iOS app that is similar to Scratch but is much simpler and easier to use. It's in between Scratch and ScratchJr in terms of structure. Hopscotch is largely an iPad application, with only the player available on other devices.	iOS	iOS, Desktop
GameSalad	A drag-and-drop programming tool for unskilled coders that allows anyone to simply develop games.	Desktop	Desktop, Mobile

Table 2.1: Famous Block-based Programming Languages

Program	**Description**	**Platform**	**Output**
GameMaker Studio	Inexperienced developers can use this drag-and-drop game creation software to build video games in a variety of genres. It also employs "Game Maker Language," a graphical user interface programming language for incorporating more advanced features into a game. This software has been used to produce several successful games, including Undertale. Although the software is not free, a trial version is available.	Microsoft Windows	Desktop, mobile, Xbox One, PlayStation 4, Nintendo Switch
Tynker	A kid-friendly drag-and-drop programming application that enables anyone to simply develop games. It includes programming tasks as well as the ability to program external devices and mod Minecraft. The service also teaches Python and HTML5, however it is not entirely free.	Desktop, Mobile	Desktop, Mobile

Table 2.1: Famous Block-based Programming Languages

Program	Description	Platform	Output
CODE.GAME	In China, it was first known as Codemao, an educational visual drag-and-drop programming software. The Kitten Editor is similar to Scratch, except it includes physics, artificial intelligence, video, and augmented reality video sensing. In Python, the Turtle Editor provides for visual drag-and-drop programming.	Desktop, Mobile	Desktop, Mobile

2.9 VPLs Vs. TPLs

As we know, visual programming requires the use of graphical elements to build programs and most regular languages and tools are text-based. This leads to several differences between the two that affect the process of creating computer code and the characteristics of the projects you can create. Below is a comparison on several key points, revealing some of the main pros and cons of visual programming compared to text-based programming.

Simplicity is the bread-and-butter of visual programming. Most are designed to make the initial learning process as easy as possible with drag-and-drop interfaces and predefined actions to choose from.

Conversely, traditional languages could be more difficult for would-be developers to get started with. People must master several important concepts and processes before they can be truly productive.

Power and Flexibility

Designing a computer programming language still involves several trade-offs. Particularly, there is a well-known trade-off between ease of use and expressive power.

While visual programming can be much easier to use for straightforward tasks, they can be cumbersome for more complex projects. Employing visual objects to describe complicated logic and control flow can result in a jumble of nodes and connections that is incredibly difficult to read and manage, defeating the original purpose of a VPL.

Traditional text-based code, on the other hand, allows for much greater flexibility in describing computations. To put it simply, it is usually easier to express certain concepts through visual programming than it is in text-based programming. However, it is certainly easier to code more advanced operations through an old-fashioned textual computer programming language.

Debugging

Finding and fixing errors in your projects, also known as debugging, is an important part of computer programming that is often underappreciated by beginners. Bugs will creep into code no matter how careful a developer is, and it can be surprisingly difficult to locate and resolve them.

Debugging can be either easier or harder in a VPL. Since VPLs provide immediate visual feedback for everything in the program, it can potentially be easier to figure out what went wrong. Conversely, a traditional language typically has extensive debugging support built into their development environments. As the projects become larger and more complex, the benefits of these tools become obvious.

Performance

By simplifying the practical details of a low-level language, visual programming becomes easier to work with. Unfortunately, this also makes it much harder to create programs that perform well. Thus, when performance is a concern, a traditional language would have a large advantage over a visual one.

Predefined Tasks

While traditional text-based code has the edge when it comes to raw power, VPLs can shine when they are designed to solve specific problems in a particular field. For instance, LabVIEW contains many tools for accomplishing common tasks needed by scientists and engineers. But when you need custom functionality not provided by the language, it is not always easy to build within the VPL itself. In this case, there is no substitute for the flexibility of general-purpose textual computer programming.

Learning Potential

Since many VPLs are specifically designed to aid learning, they make for excellent steppingstones early on. However, that is not the whole story. At some point, the user may hit a wall due to the inflexibility of visual languages and tools.

If a person is interested in eventually developing serious computer programming skills, there will come a time when a transition to a conventional language like Python or Java needs to occur so they can start to get some experience writing and debugging more complex code. Much of your computer programming experience in a VPL transfers to other languages and tools since the process of creating software projects in VPLs teaches general concepts that apply across different languages.

Trends in Visual Programming

Today, visual languages are increasingly employed both in education and in domain-specific contexts like add-ons to established software. While visual programming may not be suitable for every application, their growing adoption suggests that they are not going anywhere.

Combining VPLs with Textual Programming

A growing trend sees some development platforms combining visual programming with traditional languages in integrated environments. A standout example is the Blueprint scripting system in the Unreal Engine 4 game development engine. Blueprints allows new programmers to manipulate game objects as visual nodes.

Users of Unreal Engine are given the choice to work in pure C++, Blueprints, or a combination of the two. A developer can use Blueprints to add functionality to their game or quickly prototype a new game. In fact, it is possible to produce full games entirely using a VPL.

Responses to a StackOverflow question indicate that even large game companies producing AAA games use these visual scripting capabilities as part of their workflows. With Blueprint, non-programmers can make substantial contributions to game development without having to learn the intricacies of even the most basic of traditional languages. For large, demanding games made in Unreal Engine 4, a common approach is to use C++ for core elements of the game that are especially complex or require maximum performance. Blueprints can then be utilized to extend functionality in a straightforward way.

2.10 Summary

In this chapter it has been observed that according to the research work previously carried out on similar topics like:

- Teaching computer programming language
- Methods used to teach computer programming language
- Using of VPLs in teaching computer programming languages.
- Using Robotics to teach computer programming languages.

Are not only regarded as excellent substitutes to teaching computer programming languages normally but in a lot of cases yield better results in not only the teaching aspect but also help students to keep interest in the subject at hand for a longer period of time maximizing the learning capability as compared to traditional teaching approaches.

Chapter 3

Proposed Framework

3.1 Introduction

This chapter focuses on the framework which has been suggested. It consists of merging both the TPLs and VPLs in order to create an easy to understand and attention-grabbing environment for the students so their productivity for learning the coding languages is maximized.
Diagrams representing the overview of the framework and multiple examples of how the framework functions are presented in this chapter. Details of the framework are also provided with the diagrams in order to clearly show the working of this framework.

3.2 Proposed Framework

The concept in this paper is Blocked based programming, but unlike other visual programming languages this method has a differentiating feature, that is after a student learns how to structure the program visually, one block component of the code will be replaced by a text-based programming language syntax. This will help the student to learn the syntax of just one block rather than the whole code from start to finish. After multiple times writing the code when the software thinks that the student has achieved a good enough grip on the syntax of the block, another block will be replaced by the text-based programming code. This process will be repeated until the student has learned how to code the complete syntax of all the blocks of code and hence, they will be fluent in typing code and have a great understanding of the concepts and structure as they were initially presented by visual blocks. However, if the software detects that the student is having trouble coding certain code block syntax, the software will revert that piece of code back to its block state until a future time when it thinks it will benefit the student. This framework is further explained diagrammatically in Figure 1. The main ideas of this framework are:

- Multiple Languages support
  The blocks of this framework will be made publicly available to the developers of all languages which in turn will allow the blocks to work in most of the languages and help students to learn the language they desire.

- Accessible Front-end
  The front end will be made of HTML,CSS and other Libraries supporting these, as the advantage of using these languages is that instead of the user having to install the software the user can simply use it on any browser on all the operating systems that support web browsing.

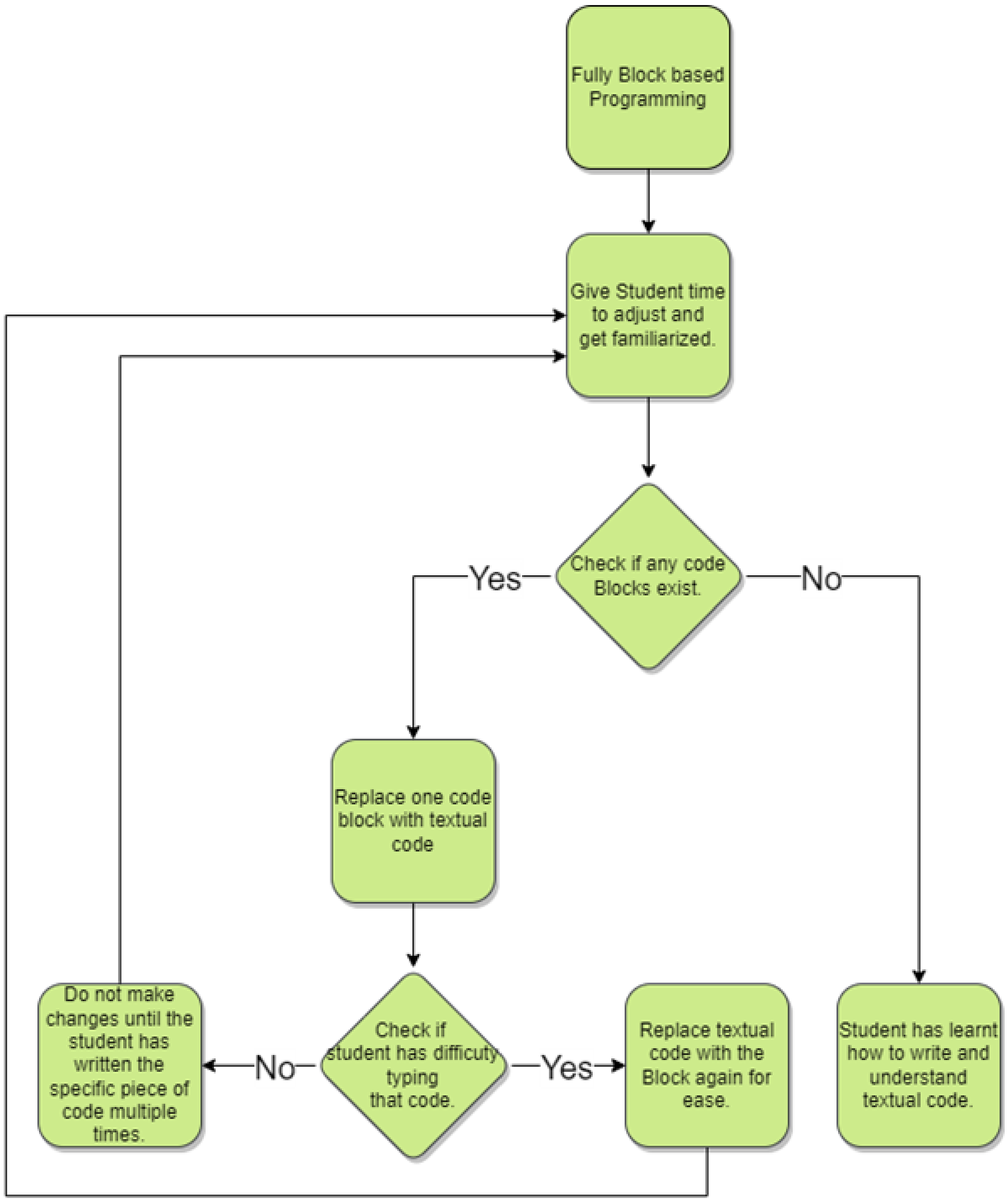

Fully Block based Programming
Give Student time to adjust and get familiarized.
Check if any code Blocks exist.
Yes
No
Replace one code block with textual code
Check if student has difficulty typing that code.
No
Yes
Do not make changes until the student has written the specific piece of code multiple times.
Replace textual code with the Block again for ease.
Student has learnt how to write and understand textual code.

Figure 3.1: Block-based programming Proposed Framework

3.3 Research Methodology

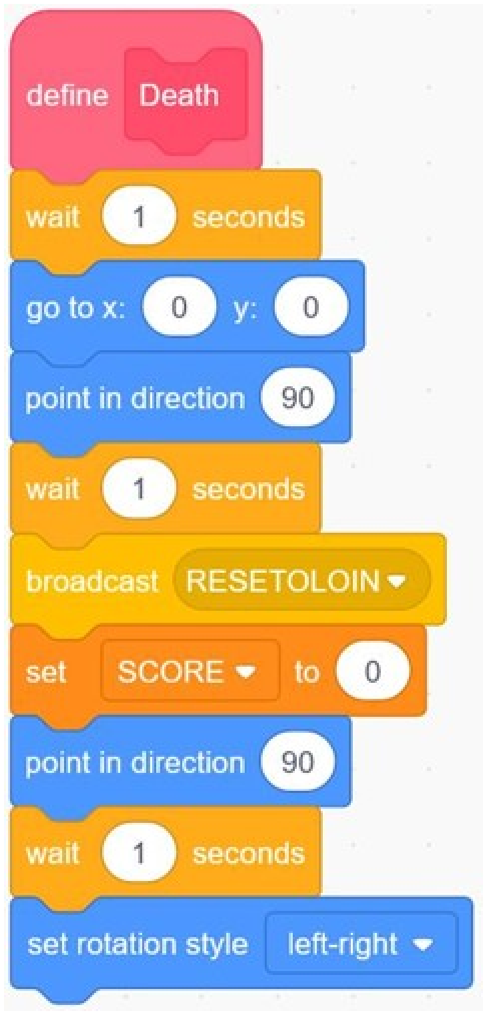
The primary method used to carry out the research was a Survey based approach. A questionnaire with some specific tasks was created for the students. A few tasks were also made for the students in order for them to get a hands-on approach to computer programming and Visual Programming tasks. After the tasks were concluded the students were presented with a questionnaire to provide their feedback and opinions on the experiment carried out on them. There were a total of 304 students, and their main subjects were Electrical Engineering and Mechanical Engineering. After the results have been collected from the students, we made graphical images to visualize the results, these images are provided in the paper later.

3.4 Explanation with Example

Once a beginner wants to start learning the concepts of coding and the language hierarchy instead of code, they are presented with a block-based programming environment as shown in figure 2. This will help the beginner get an “overview” of the layout of how computer programming languages work and the hierarchy in which they work. The user will be using this block-based environment for some time until they have understood how the structure of coding works.

3.5 Description of Code

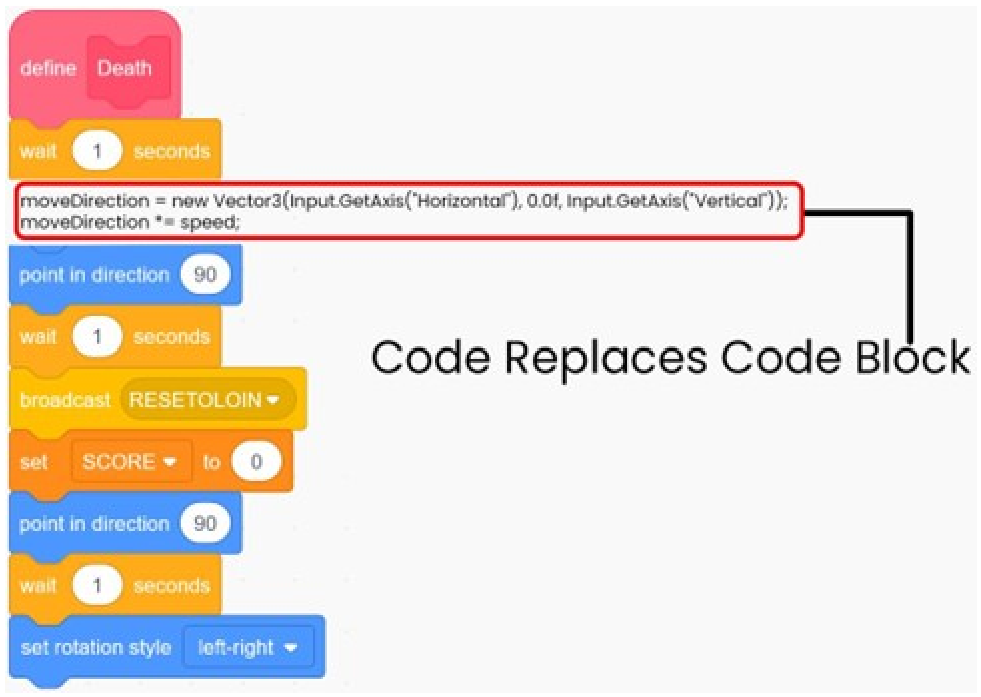
The code blocks in this image display a character losing or “death” animation. The first block represents the name, 2nd,5th, and 9th display the time before animation plays, 3rd, 4thand 8thtell the positioning of the character,6th resets the position of the object, 7th block resets the score, and 10 sets the rotation style (animation type).



```
define Death
wait 1 seconds
go to x: 0 y: 0
point in direction 90
wait 1 seconds
broadcast RESETOLOIN
set SCORE to 0
point in direction 90
wait 1 seconds
set rotation style left-right
```

Figure 3.2: Block-based programming Proposed Framework

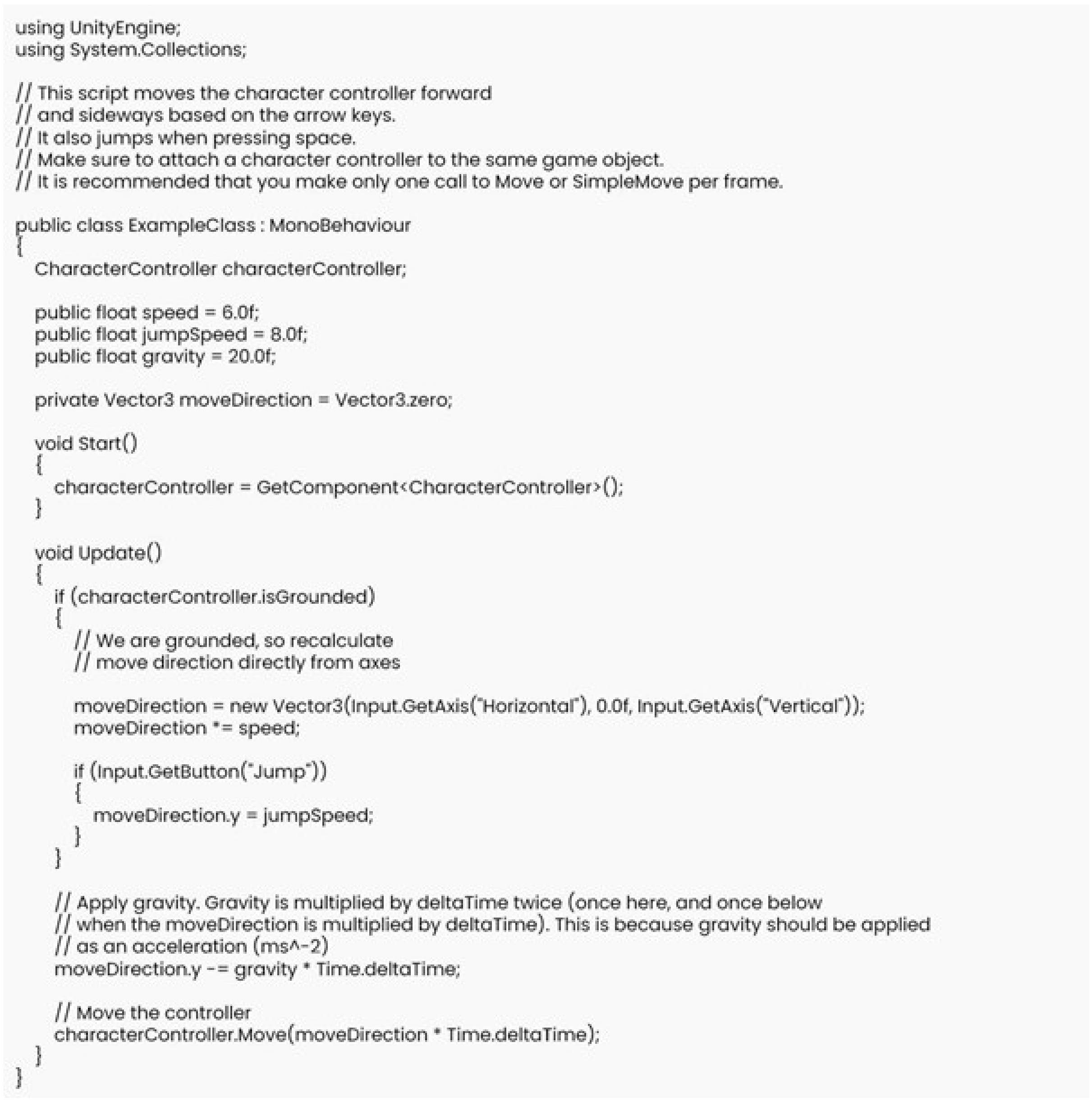
After the user is comfortable with the basics of computer programming one of the blocks will change into code with complete syntax. This is good for the beginner as instead of having to understand the whole code from beginning to end, they would just have to understand one snippet of code. This not only will reduce the learning curve drastically but will also help the user in learning and understanding that piece of code. This is shown in figure 3.3.



```
moveDirection = new Vector3(Input.GetAxis("Horizontal"), 0.0f, Input.GetAxis("Vertical"));
moveDirection *= speed;
```

Figure 3.3: Representation of replacing a block of code with actual code

When the user is comfortable in coding that string of code, we will replace the 2nd block with code. So, the user must learn that while still writing that and the old code they just learned. If this proves too much for the user, the code will revert to its block state however if the user gets a grip on this piece of code the block will remain code until the user has mastered it. This process keeps on repeating until there are no blocks left which means that the user has learnt to write and understand code and its syntax, as shown in figure 3.4.



```
using UnityEngine;
using System.Collections;

// This script moves the character controller forward
// and sideways based on the arrow keys.
// It also jumps when pressing space.
// Make sure to attach a character controller to the same game object.
// It is recommended that you make only one call to Move or SimpleMove per frame.

public class ExampleClass : MonoBehaviour
{
  CharacterController characterController;

  public float speed = 6.0f;
  public float jumpSpeed = 8.0f;
  public float gravity = 20.0f;

  private Vector3 moveDirection = Vector3.zero;

  void Start()
  {
    characterController = GetComponent<CharacterController>();
  }

  void Update()
  {
    if (characterController.isGrounded)
    {
      // We are grounded, so recalculate
      // move direction directly from axes

      moveDirection = new Vector3(Input.GetAxis("Horizontal"), 0.0f, Input.GetAxis("Vertical"));
      moveDirection *= speed;

      if (Input.GetButton("Jump"))
      {
        moveDirection.y = jumpSpeed;
      }
    }

    // Apply gravity. Gravity is multiplied by deltaTime twice (once here, and once below
    // when the moveDirection is multiplied by deltaTime). This is because gravity should be applied
    // as an acceleration (ms^-2)
    moveDirection.y -= gravity * Time.deltaTime;

    // Move the controller
    characterController.Move(moveDirection * Time.deltaTime);
  }
}
```

Figure 3.4: Full code when the student has learnt all the necessary programming

3.6 Visual Interface

This section presents a detailed overview of the UI(User Interface) of what a VPL environment is comprised of and the different tools, features, functions and components which are presented to the user. Using these Visual elements the user can edit, modify and join blocks of code in such a way that they make up a functioning

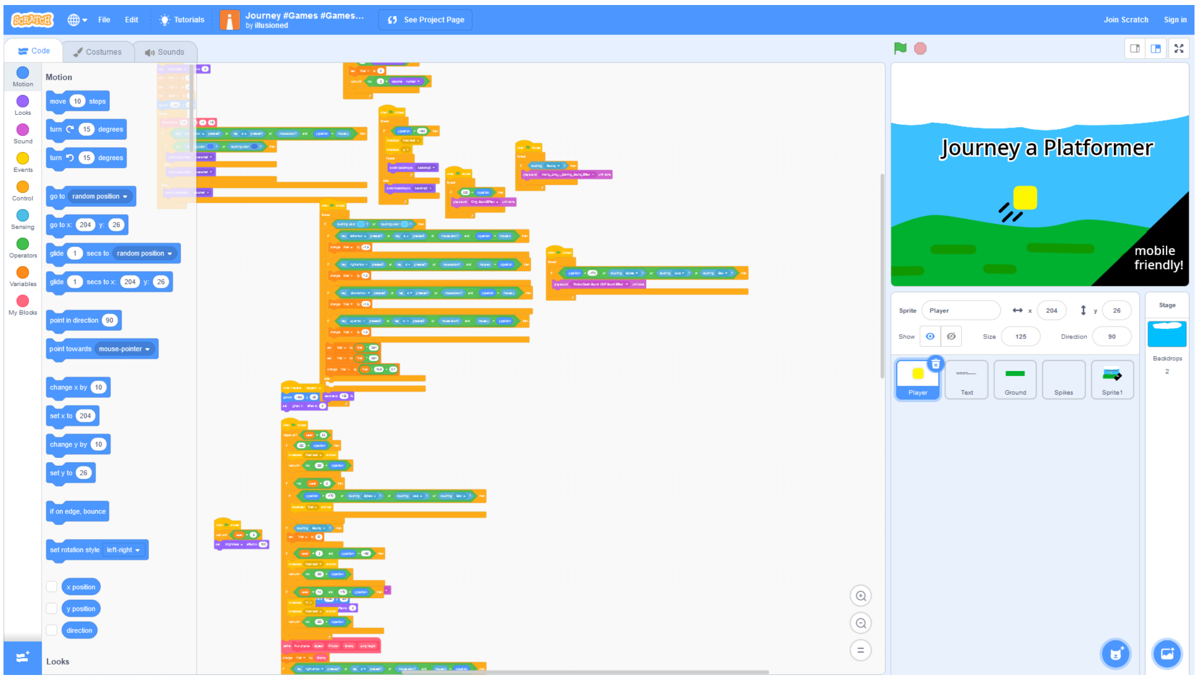
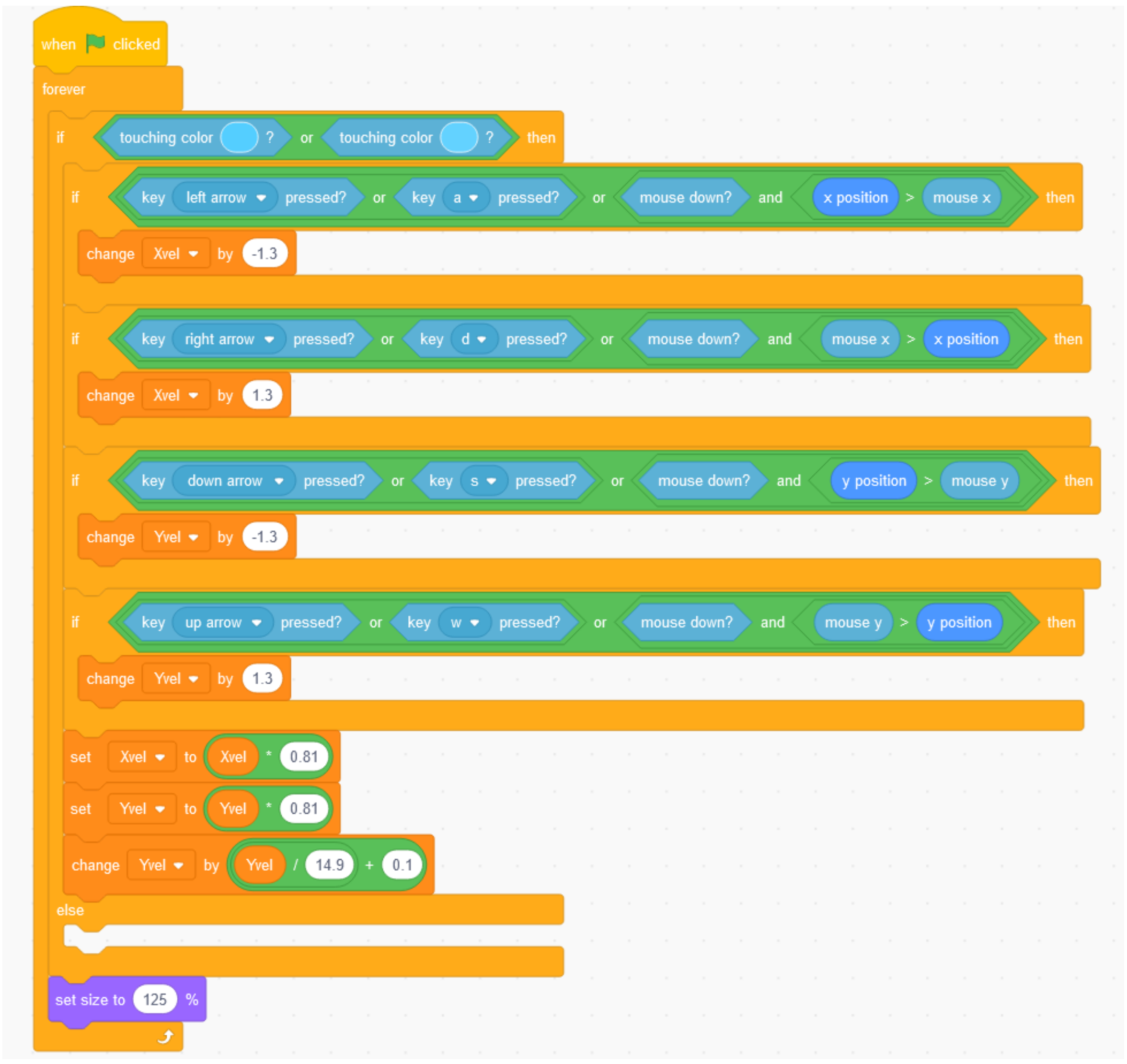


Figure 3.5: Overview of the complete Scratch environment

The image 3.5 demonstrated above is what the user is presented with when they open up a scratch project. The left window is called the "Component Window" and holds the code blocks which can be dragged and dropped into the middle window in order to function. The window in the middle is where the blocks can be merged, edited and placed in order to tell the compiler how to run the code. Lastly the window on the right contains the details of a project and the "Sprites" used in it. It also provides the option to add more sprites either from the users computer or from Scratch’s library which consists of thousands of sprites, sound files, images and videos.



```
when ⚑ clicked
forever
  if touching color ( ) ? or touching color ( ) ? then
    if key left arrow ▾ pressed? or key a ▾ pressed? or mouse down? and x position > mouse x then
      change Xvel ▾ by -1.3
    if key right arrow ▾ pressed? or key d ▾ pressed? or mouse down? and mouse x > x position then
      change Xvel ▾ by 1.3
    if key down arrow ▾ pressed? or key s ▾ pressed? or mouse down? and y position > mouse y then
      change Yvel ▾ by -1.3
    if key up arrow ▾ pressed? or key w ▾ pressed? or mouse down? and mouse y > y position then
      change Yvel ▾ by 1.3
    set Xvel ▾ to Xvel * 0.81
    set Yvel ▾ to Yvel * 0.81
    change Yvel ▾ by Yvel / 14.9 + 0.1
  else
  set size to 125 %
```

Figure 3.6: Code Blocks Example

In image 3.6, we focus on the main or the most used component of VPLs, the Code Blocks. This code block represents a script for character movement. The total amount of code blocks this has is 10 (excluding the topmost flag code block). As it can be observed the code blocks are editable. Hence, we can type in the exact amount of numbers we wish in order to translate the movement speed of the character. It should also be noted that the blocks can be rearranged in any order that the user finds feasable so their software works. For example, we can put the last code block in the middle if we need it to be there and the other code blocks will automatically adjust their size and space in order to incorporate the newly added code block.

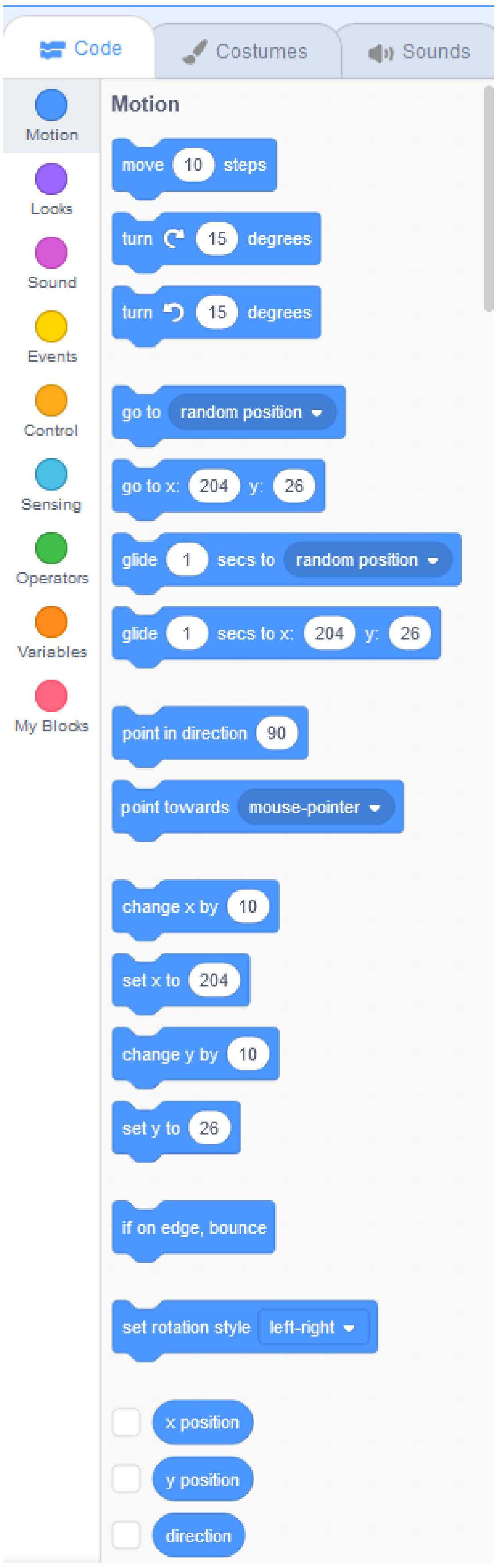


Figure 3.7: Components

The components are the Codeblocks which can be dragged and dropped into the main window in order to make functioning code. The blocks can be edited as well to provide fine control over the computer program. The components are divided into other cartegories. the categories consist of:

- Motion.
- Looks
- Sound
- Events
- Control
- Sensing
- Operators
- Variables
- My Blocks

3.7 Summary

The chapter above demonstrates the working of the framework with multiple examples and diagrams. An overview of the framework is also provided, for ease of understanding. Using the methods shown in this chapter the experimentation and results were formed of the students. The results obtained by using the above-mentioned methods are provided in the next chapter.

Chapter 4

Experimentation and Results

4.1 Introduction

Here the survey details are provided. The questions which were asked in the survey as well as the answers to the questions within the survey are provided. In addition to the survey details, the equipment used to carry out this experiment, the environment and the computers the students interacted with have their details shown below.

4.2 Lab Set Up

The lab setup consisted of 56 computers with internet availability and the IDEs for textual based programming languages already installed as this computer lab is used to teach programming languages to students. The Operating System used on the computers is Windows 10 as it is the most commonly used Operating System and the most familiar to the students. The computer systems available in the lab are also equipped with modest hardware specifications which are:

- Intel core i-5 6th gen
- 8 Gigabytes of RAM
- 500 Gigabytes of Storage (HDD)

Which is plenty for compiling code and coding related tasks.

4.3 Survey of Students

The survey which was conducted afterwards for conclusive results, and it consisted of the following questions:

- Have you ever used a programming language before?
- Have you ever used Visual Programming languages before?
- Did the VPL help you in understanding the concept of how code is structured?
- Was the VPL easier to understand than the textual programming language?
- Did this method of incorporating both visual and textual based programming-based languages help you in understanding the fundamental programming concepts?
- Do you prefer Visual Programming languages over Textual Programming languages?

4.4 Survey Results

The acquired results from the Survey are shown below in diagrammatical form:

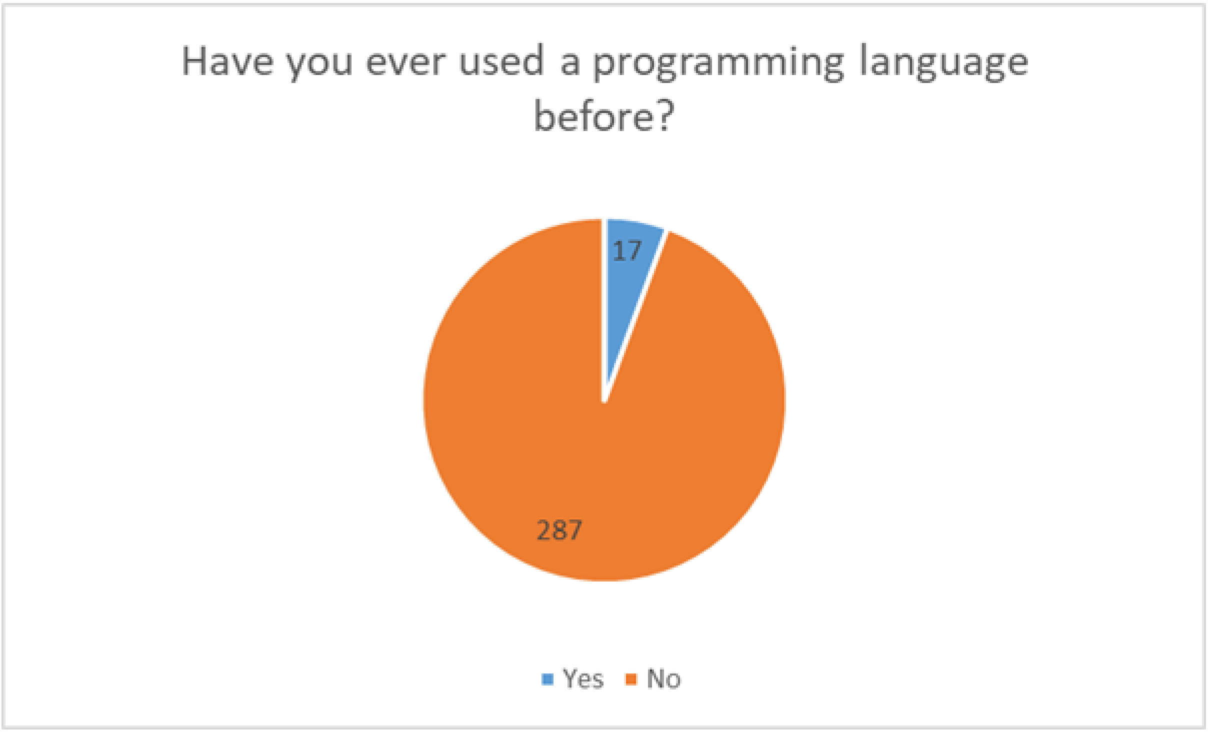


Figure 4.1: Previously used Programming languages

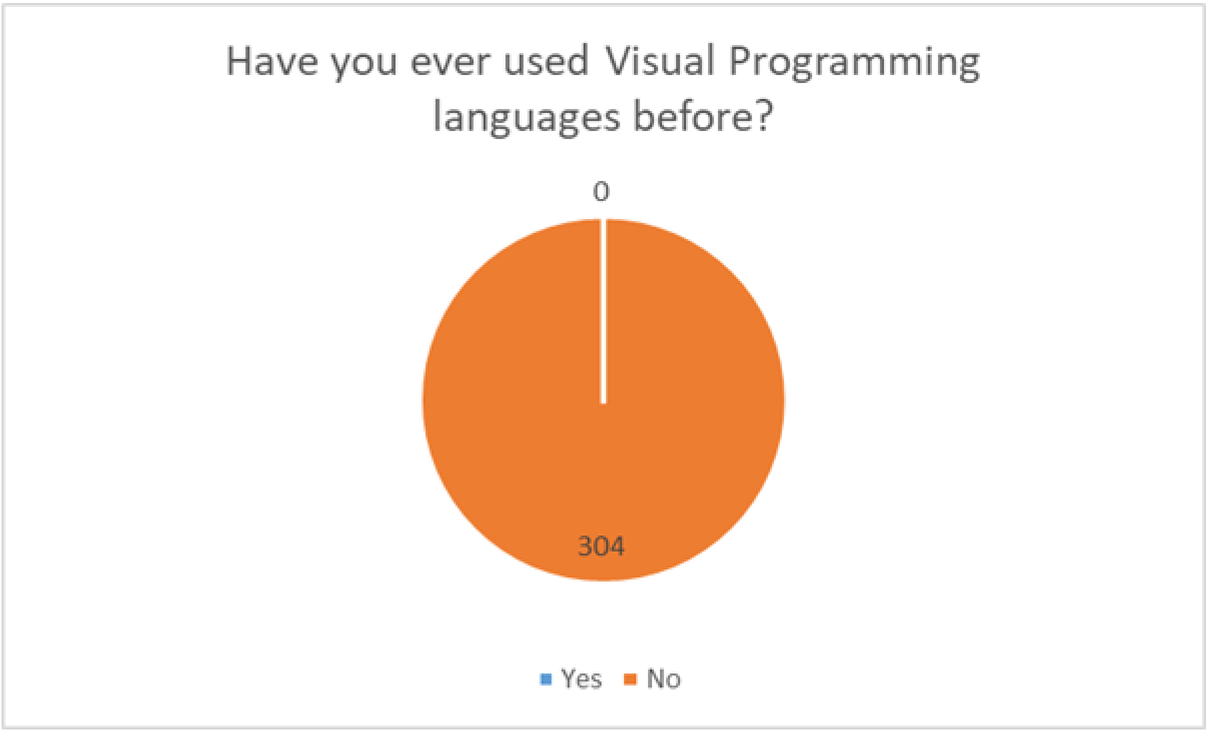


Figure 4.2: Previously used Visual Programming languages

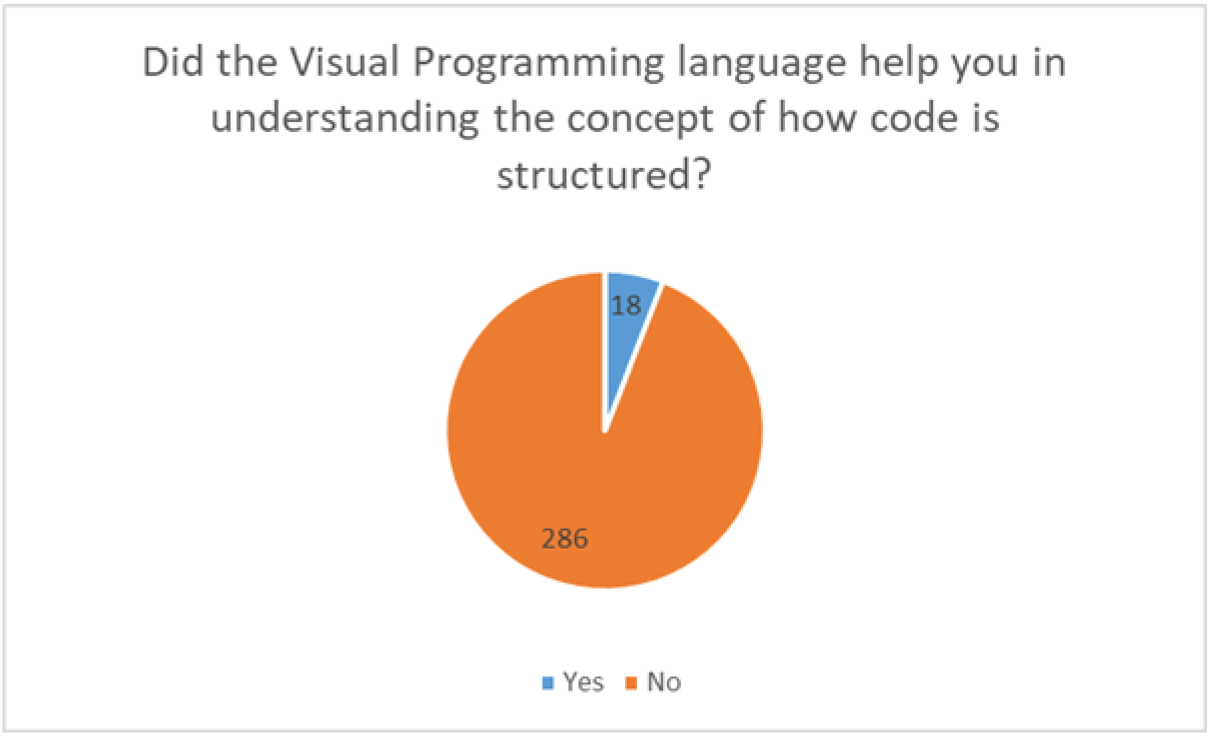


Figure 4.3: If VPL Helpful or not

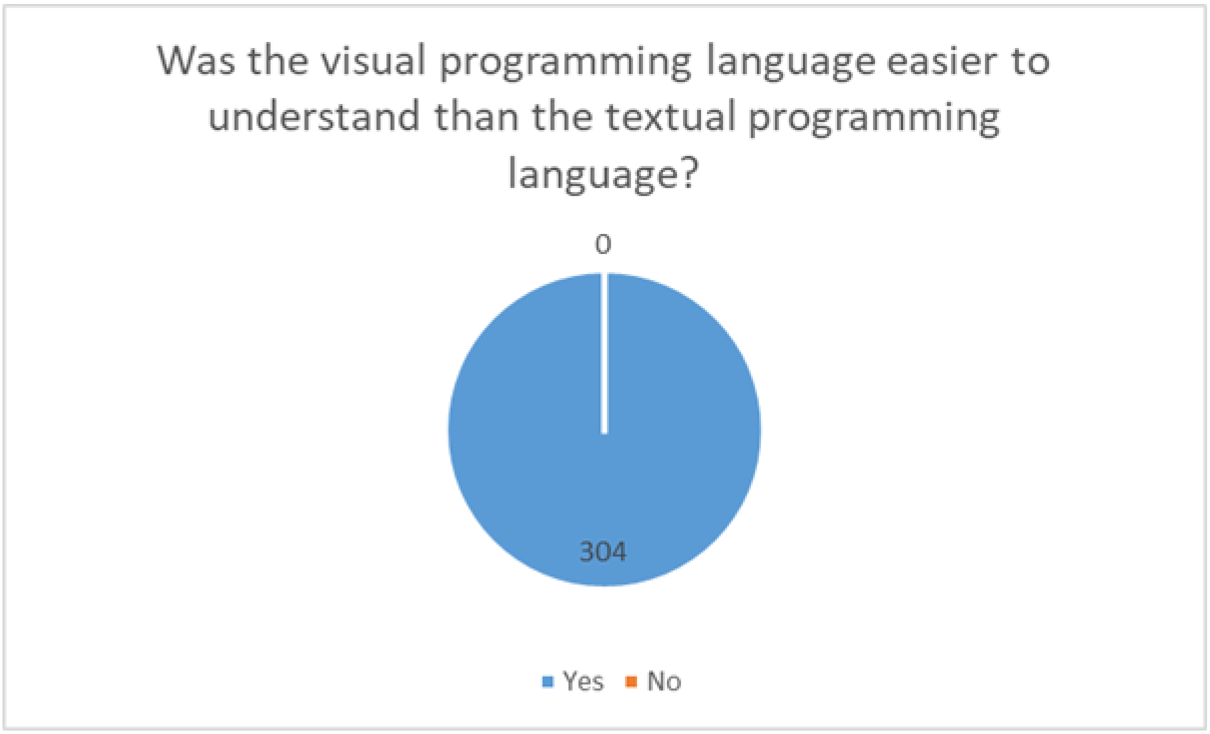


Figure 4.4: If it was easier than TPL or not

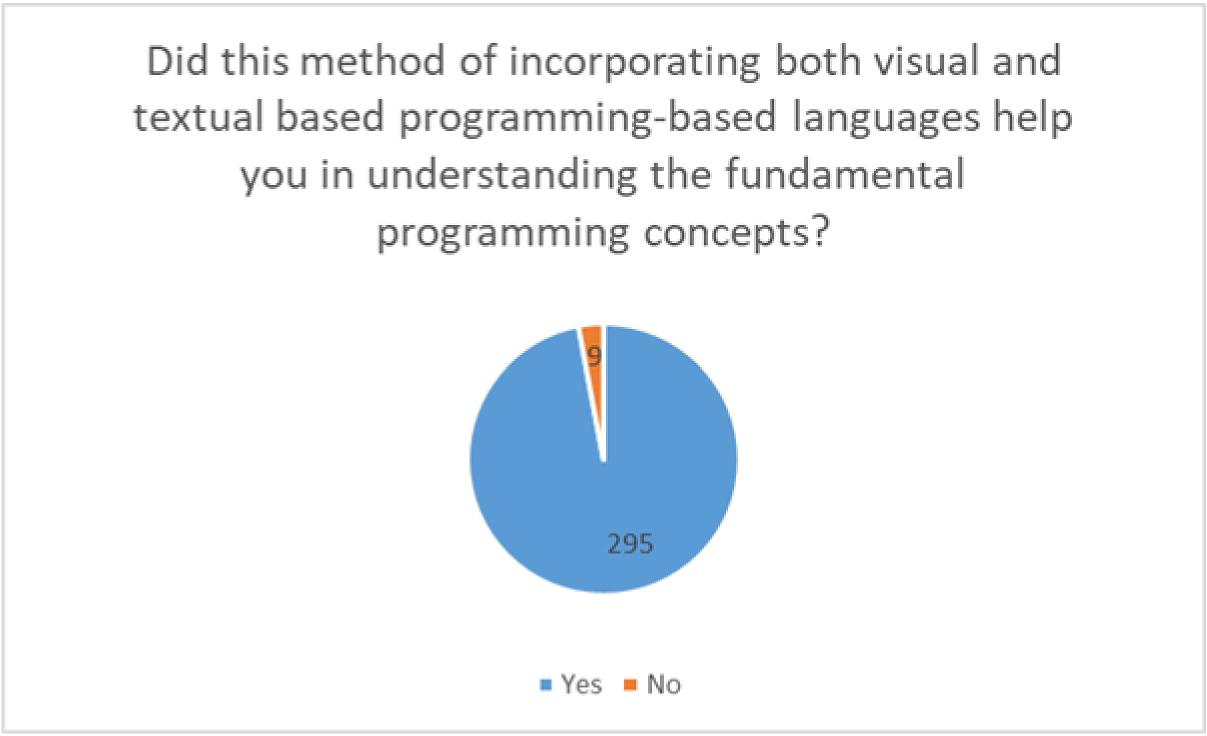


Figure 4.5: Helpful in teaching students

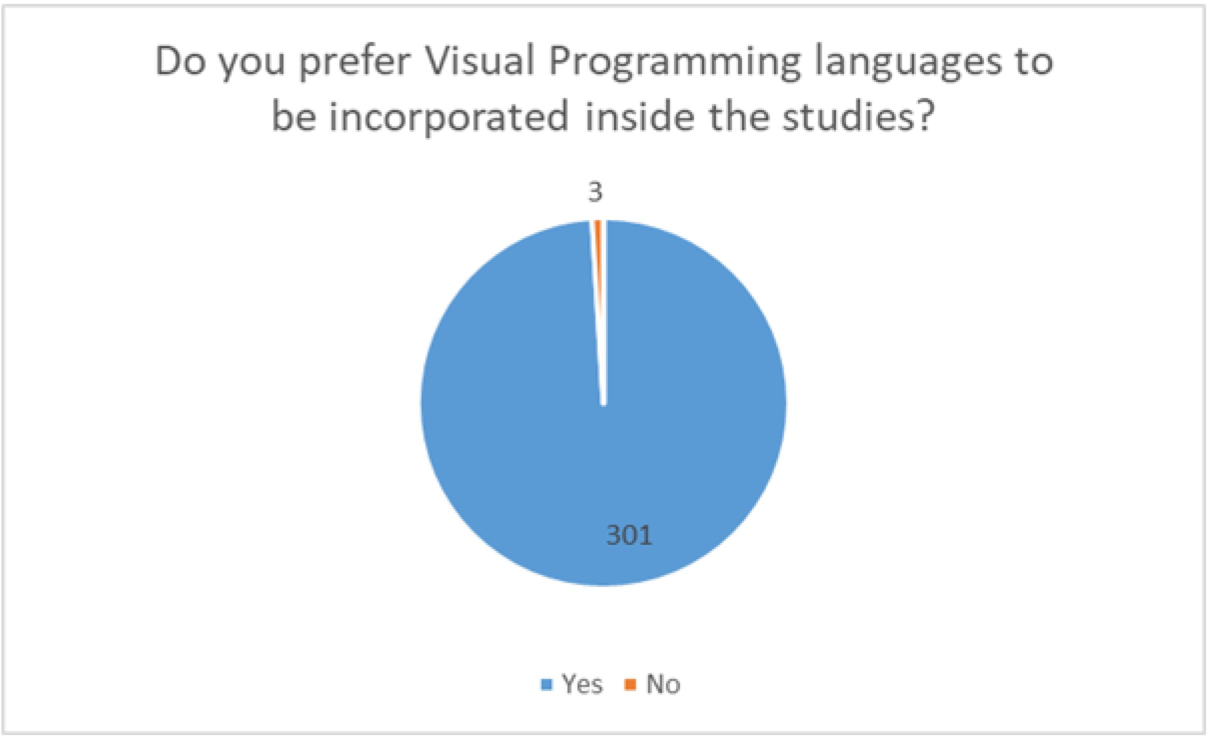


Figure 4.6: Preference about VPL or TPL

4.5 Summary

A general overview of the environment, the computers and survey were provided in this chapter. The computers are good for computer programming hence no issues are faced from the hardware perspective of the computers. The survey questions are also mostly positive which support the experimentation and the method used on the students to encompass the new computer programming language.

Chapter 5

Analysis and Discussion

5.1 Introduction

This chapter discusses the results obtained by the experimentation of teaching computer programming languages using the technique suggested prior. The details of the number of students, their educational background, and the details of how the experiment was carried out in steps is described ahead. The results obtained by this experiment is also shown after that both diagrammatically and explained in text form as well. Finally, the impact on students and the verdict of the experiment is provided.

5.2 Results and Discussion

A total of 304 students were taken from a private Pakistani university in the general area of Islamabad. The subjects they were studying were Mechanical and Electrical Engineering. Upon asking all the students belonging in the second year of undergraduate courses, 94The students were given a very basic piece of code for the first time and a general explanation of how it works, and then they were asked to reproduce it themselves. After this, they were presented with the VPL "Scratch" to make the same code with the visual interface. After the 2nd phase of the experiment was conducted, we finally asked the students to replace random visual code blocks with textual-based code to see if they understood the flow and syntax of the code.
Presented above is the general overview of how the experiment was conducted; now looking at the results obtained from this experiment.

When the students were presented with a regular text-based programming language, we surveyed them if they understood the syntax or not. 97% of the students did not understand the syntax at all while the other 3% said that they understand a little bit of code at random. Furthermore, the code they wrote was just copied from the introductory code we provided them and just a few values were changed since we did not stop any student from copying but no one wrote the full code themselves. We asked the students if they understood the syntax or the structuring of the code to which most of them replied with no. The student's response is graphically shown in figure 5.1.

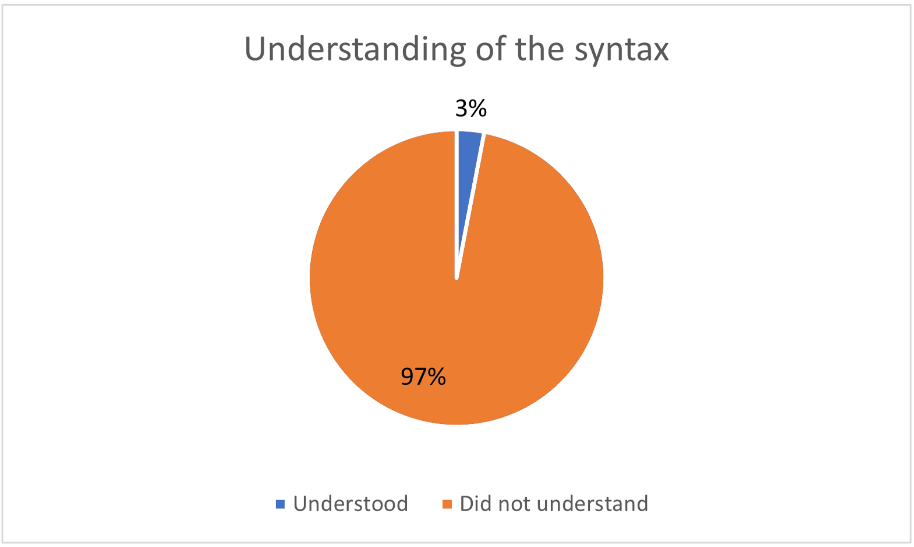


Figure 5.1: Understanding of students after 1st experiment

After the Textual-based programming test, we provided students links to "Scratch" and showed them the same piece of code but in a visual environment. After explaining how it works, we asked the students to make the same code in "Scratch". In this test, 88% of the students understood how to make the same program while the other 12% made minor mistakes which were rectified by the teaching staff along with clarifying their queries about block-based programming tools. After this experimentation, students were asked to provide feedback. The response was encouraging. The aim of teaching the beginners was achieved, as they acknowledged the understanding of the structuring of code, also the order of variables to be declared and used. This is shown in figure 5.2.

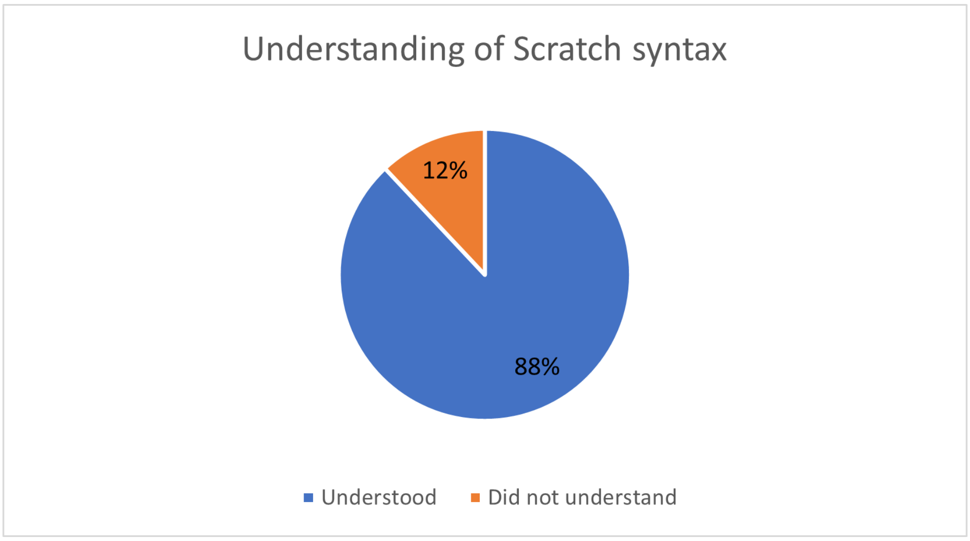


Figure 5.2: Understanding of students after 2nd experiment

Finally, we asked the students to replace a random visual code block with actual code which they had attempted in the first experiment. The experimental results show that 78% of all the students wrote the code correctly in their first attempt. 12% had minor issues while 10% still had difficulty in coding. Diagrammatically these results are illustrated in figure 5.3.

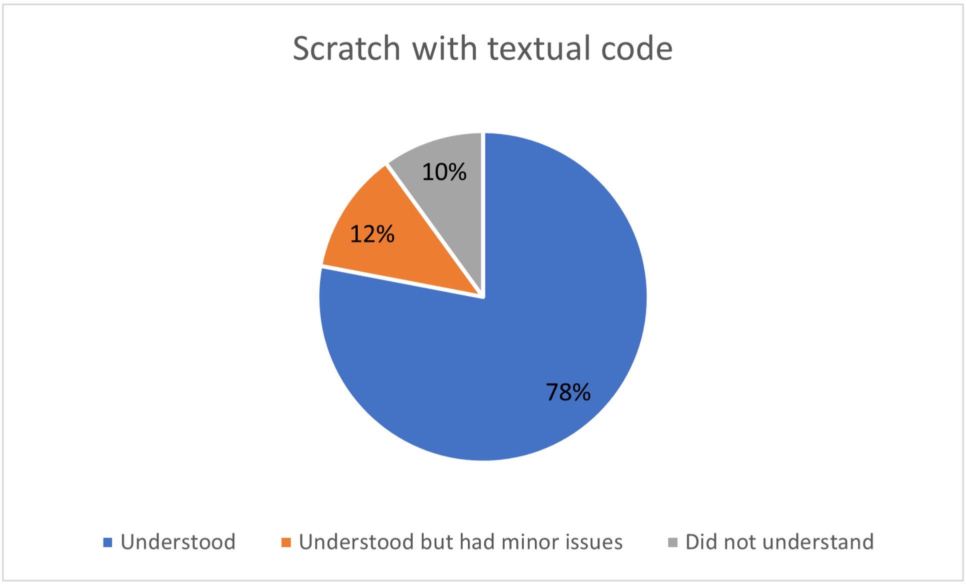


Figure 5.3: After using Both Textual and Visual Programming languages

5.3 Comparison of Understandability of Syntax

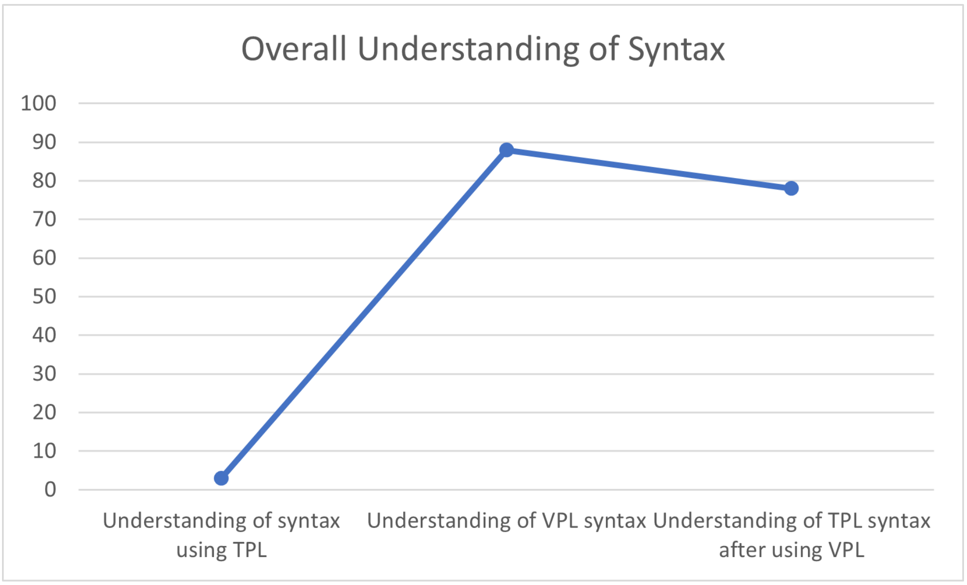


Figure 5.4: 8 Final Results of the progress of learning

The graph in figure 5.4 consists of all the results that had been obtained so far from the students. The graph line is divided into 3 main categories representing the steps of the experiments conducted. The first part of the graph line in figure 8, illustrates that in the beginning understanding of the syntax was minimum. After the 2nd experiment, the results improved drastically, and they rose from 3% to 88%. Finally, in the end, when we combined both the syntax and VPL we came to know that the results have improved from 3% to 78% which is a very positive and reassuring result for teaching using VPLs.

5.4 Impact on Students Learnability

The experimentation comprised of more than 300 students from undergraduate level of engineering disciplines. The outcome of these experiments have depicted that the students learnt a lot of positive abilities such as:

- An ability to identify, understand, and solve complex syntax of computer programming languages with ease.
- An ability to communicate effectively with coding syntax.

- An ability to integrate the use of modern computer-based VPL tools into engineering applications.
- An ability to grasp knowledge quicker and with less effort yet have great understanding regarding the subject at hand.

5.5 Summary

As it has been observed above this experiment has provided very positive and overall good results. The students have shown both interest and willingness to learn using the proposed methodology and have also found it easier to understand in a shorter amount of time than traditional methods which have been used for teaching. Some students had difficulties, but they were very easily rectified with the help of their teachers. In the end it has been overall very positive and a good alternative to the teaching methods which are currently being used by the educational institutes.

Chapter 6

Conclusion

6.1 Conclusion

Programming is a very important field in current and future times as the industry is moving towards software more and more every day. This method helps to ease the journey of students and newcomers to computer programming by firstly focusing on the concepts and then slowly focusing on syntax rather than other traditional methods. In the future, we hope that this software idea gets developed and improves further by adding features like social interactions so students' progress is directly shared with the teachers so they can help them when they need it and keep an eye on the progress of the students. As we have seen students yield better results in understanding the structuring of computer programming using this method which helps eliminate long and lengthy learning procedures.

6.2 Future Work

This is a very ambitious concept proposed for teaching purposes as a teaching tool for students in the initial stage of learning computer programming languages. Indeed, it has quite a room for improvement. Therefore, in the future, the proposed methodology can be made more efficient and more visual elements for easy understanding can be added. Besides, an application can be developed to support both TPL and VPL in the same environment for the ease of the user. We can also add recommendation-based tasks which will help each user learn at their own pace and any idea which helps students to grasp the knowledge quickly and efficiently.

6.3 Limitations

Looking at the immense positive response to the research which has been carried out on this teaching method, there are shortcomings to it as well. The most obvious one is that as of right now no official application of this method exists. To carry out the research we had to use both scratch and any IDE for textual code. This on its own is not a huge problem but it would be very convenient to make an application which supports both features.
Another limitation is that this is internet based, which means if by any chance the internet connection is disturbed or not available, this would not work. So, a development of an offline localized software which does not require internet would make this accessible to students with slow or no internet connections available.

References

[1] Edgar Serrano Pérez and Fernando Juárez López. An ultra-low cost line follower robot as educational tool for teaching programming and circuit's foundations. *Computer Applications in Engineering Education*, 27(2):288–302, 2019. doi: https://doi.org/10.1002/cae.22074. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/cae.22074.

[2] Siti Rosminah, siti rosminah md derus, Ahmad Zamzuri Mohamad Ali, and Mohamad Ali. Difficulties in learning programming: Views of students. 09 2012. doi: 10.13140/2.1.1055.7441.

[3] Pei-Hsuan Lin and Shih-Yeh Chen. Design and evaluation of a deep learning recommendation based augmented reality system for teaching programming and computational thinking. *IEEE Access*, PP:1–1, 03 2020. doi: 10.1109/ACCESS.2020.2977679.

[4] Izabela Perenc, Tomasz Jaworski, and Piotr Duch. Teaching programming using dedicated arduino educational board. *Computer Applications in Engineering Education*, 27(4):943–954, 2019. doi: https://doi.org/10.1002/cae.22134. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/cae.22134.

[5] Kanika Kanika, Shampa Chakraverty, and Pinaki Chakraborty. Tools and techniques for teaching computer programming: A review. *Journal of Educational Technology Systems*, 49:170–198, 05 2020. doi: 10.1177/0047239520926971.

[7] C. C. Stephanidis and G. Salvendy, "Seven HCI Grand Challenges", International Journal of Human-Computer Interaction, vol. 35, Number 14, 2019 pp. 1229-1269.

[8] H. Tsukamoto, Y. Oomori, H. Nagumo, Y. Takemura, A. Monden and K. Matsumoto, "Evaluating algorithmic thinking ability of primary schoolchildren who learn computer

programming," 2017 IEEE Frontiers in Education Conference (FIE), Indianapolis, IN, 2017, pp. 1-8.

[9] “Statutory guidance National curriculum in England: computing programmes of study,” [Available] https://www.gov.uk/government/publications/national-curriculum-in- england-computing-programmes-of-study/n.

[10] J. E. Sammet, "Programming Languages: History and Future" , Communications of the ACM, vol. 15, Number 7, 1972, pp. 601-610.

[11] N. Bak, B. Chang and K. Choi, "Smart Block: A Visual Programming Environment for SmartThings," 2018 IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC), Tokyo, 2018, pp. 32-37.

[12] H. Tsukamoto et al., "Textual vs. visual programming languages in programming education for primary schoolchildren," 2016 IEEE Frontiers in Education Conference (FIE), Erie, PA, USA, 2016, pp. 1-7.

[13] T. Karvounidis, I. Argyriou, A. Ladias and C. Douligeris, "A design and evaluation framework for visual programming codes," 2017 IEEE Global Engineering Education Conference (EDUCON), Athens, 2017, pp. 999-1007.

[14] Chen and Wang "VIPLE: Visual IoT/Robotics Programming Language Environment for Computer Science Education",2019.

[15] B. Frey, Moving from the Known to the Unknown to Measure the Initial Learnability of Programming Languages,IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC) 2017.

[16] J. María Rodríguez Corral, "A Study on the Suitability of Visual Languages for Non-Expert Robot Programmers",Received November 25, 2018, accepted January 23, 2019, date of publication January 29, 2019, date of current version February 14, 2019.

[17] P. Gao, "A New Teaching Pattern Based on PBL and Visual Programming in Computational Thinking Course", The 14th International Conference on Computer Science and Education (ICCSE 2019) August 19-21, 2019. Toronto, Canada,2019.

[18] C. Kyfonidis, "Block C: A block based programming teaching tool to facilitate introductory C programming courses", 2017 IEEE Global Engineering Education Conference (EDUCON).

[19]A. Rao, "Milo: A visual programming environment for Data Science Education", 2018 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC).

[20] B.J. Smith, Harry S. Delugach, "Work In Progress - Using a Visual Programming Language to Bridge the Cognitive Gap Between a Novice's Mental Model and Program Code", 2010, Washington, DC 40 th ASEE/IEEE Frontiers in Education Conference.

[21]Abdul Samad ,Arsalan, Muhammad ,Mufti, Anees ,Hameed, Mazhar. (2019). Comparative Study of Conventional Methods and Augmented Reality: Effects on Class Performance. 7. 7-29.

[22] C. C. Stephanidis and G. Salvendy, "Seven HCI Grand Challenges", International Journal of Human-Computer Interaction, vol. 35, Number 14, 2019 pp. 1229-1269.

[23] H. Tsukamoto, Y. Oomori, H. Nagumo, Y. Takemura, A. Monden and K. Matsumoto, Evaluating algorithmic thinking ability of primary schoolchildren who learn computer rogramming," 2017 IEEE Frontiers in Education Conference (FIE), Indianapolis, IN, 2017, p. 1-8.

[24] "Statutory guidance National curriculum in England: computing programmes of study," [Available] https://www.gov.uk/government/publications/national-curriculum-in-england-computing-programmes-of-study/n.

[25] J. E. Sammet, "Programming Languages: History and Future" , Communications of the CM, vol. 15, Number 7, 1972, pp. 601-610.

[26] N. Bak, B. Chang and K. Choi, "Smart Block: A Visual Programming Environment for SmartThings," 2018 IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC), Tokyo, 2018, pp.32-37.

[27] H. Tsukamoto et al., "Textual vs. visual programming languages in programming education for primary schoolchildren," 2016 IEEE Frontiers in Education Conference (FIE), Erie, PA, USA, 2016, pp. 1-7.

[28] T. Karvounidis, I. Argyriou, A. Ladias and C. Douligeris, "A design and evaluation framework for visual programming codes," 2017 IEEE Global Engineering Education Conference (EDUCON), Athens, 2017, pp. 999-1007.

[29] Chen and Wang "VIPLE: Visual IoT/Robotics Programming Language Environment for Computer Science Education",2019.

[30] B. Frey, Moving from the Known to the Unknown to Measure the Initial Learnability of Programming Languages,IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC) 2017.

[31] J. María Rodríguez Corral, "A Study on the Suitability of Visual Languages for Non-Expert Robot Programmers",Received November 25, 2018, accepted January 23, 2019, date of publication January 29,2019, date of current version February 14, 2019.

[32] P. Gao, "A New Teaching Pattern Based on PBL and Visual Programming in Computational Thinking Course", The 14th International Conference on Computer Science and Education (ICCSE 2019) August 19-21, 2019. Toronto, Canada,2019.

[33] C. Kyfonidis, "Block C: A block based programming teaching tool to facilitate introductory C programming courses", 2017 IEEE Global Engineering Education Conference (EDUCON).

[34] A. Rao, "Milo: A visual programming environment for Data Science Education",

2018 IEEE Symposium on Visual Languages and Human Centric Computing (VL/HCC).

[35] B.J. Smith, Harry S. Delugach, "Work In Progress - Using a Visual Programming Language to Bridge the Cognitive Gap Between a Novice’s Mental Model and Program Code", 2010, Washington, DC 40th ASEE/IEEE Frontiers in Education Conference

[36] B. Frey, Moving from the Known to the Unknown to Measure the Initial Learnability of Programming Languages,IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC) 2017.

[37] J. María Rodríguez Corral, "A Study on the Suitability of Visual Languages for Non-Expert Robot Programmers”, Received November 25, 2018, accepted January 23, 2019, date of publication January 29, 2019, date of current version February 14, 2019.

[38] P. Gao, "A New Teaching Pattern Based on PBL and Visual Programming in Computational Thinking Course", The 14th International Conference on Computer Science and Education (ICCSE 2019) August 19-21, 2019. Toronto, Canada,2019

[39] C. Kyfonidis, "Block C: A block based programming teaching tool to facilitate introductory C programming courses", 2017 IEEE Global Engineering Education Conference (EDUCON).

[40] A. Rao, "Milo: A visual programming environment for Data Science Education", 2018 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)

[41] M. Anees-Ur-Rahman, “PFE: A Visual Programming Frame Work for Teaching Programming to Dummies or beginners.” 2020 University of Sindh Journal of Information and Communication Technology, 4(3), 194 - 198.

[42] B. Myers, "Taxonomies of visual programming and program visualization",Journal of Visual Languages and Computing, vol. 1, no. 1, pp. 97-123, 1990.

[43] Carl O’Brien, Irish Times newspaper in 2016.

[44] Quille, K., Bergin, S., and Mooney, A. (2015). Programming: Factors that influence success revisited and expanded. In International Conference on Enguaging Pedagogy (ICEP), 3rd and 4th December, College of Computing Technology, Dublin, Ireland (Vol. 10, pp. 1047344-1047480).

[45] C. Chang, Y. Yang, and Y. Tsai, "Exploring the engagement effects of visual programming language for data structure courses", Education for Information, vol. 33, no. 3, pp. 187-200, 2017.

[46]. R. Hijon-Neira, L. Santacruz-Valencia, D. Perez-Marin, and M. Gomez-Gomez, "An analysis of the current situation of teaching programming in Primary Education", 2017 International Symposium on Computers in Education (SIIE), 2017.

[47]. Nielsen, J.: Usability Engineering. AP Professional (1993)

[48] R. Holwerda and F. Hermans, "A Usability Analysis of Blocks-based Programming Editors using Cognitive Dimensions," 2018 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC), 2018.

[49] Mihci, Can and ÖZDENER, Nesrin. Programming education with a blocks-based visual language for mobile application development, 2014.

[50] Mota, José Miguel, et al. "Visual Environment for Designing Interactive Learning Scenarios with Augmented Reality." International Association for Development of the Information Society, 2016.

[51] https://en.wikibooks.org/wiki/A-level_Computing/CIE/Computer_systems,_communications_a

[52] Serrano Pérez, E, Juárez López, F. An ultra-low cost line follower robot as educational tool for teaching programming and circuit's foundations. ComputApplEng Educ. 2019; 27: 288– 302. https://doi.org/10.1002/cae.22074

[53] Rosminah, Siti and md derus, sitirosminah and Mohamad Ali, Ahmad Zamzuri

and Ali, Mohamad. (2012). Difficulties in learning programming: Views of students. 10.13140/2.1.1055.7441.

[54] P. Lin and S. Chen, "Design and Evaluation of a Deep Learning Recommendation Based Augmented Reality System for Teaching Programming and Computational Thinking," in IEEE Access, vol. 8, pp. 45689-45699, 2020, doi: 10.1109/ACCESS.2020.2977679.

[55] Perenc, I, Jaworski, T, Duch, P. Teaching programming using dedicated Arduino Educational Board. ComputApplEng Educ. 2019; 27: 943– 954. https://doi.org/10.1002/cae.22134

[56] Chakraverty S, Chakraborty P. Tools and Techniques for Teaching Computer Programming: A Review. Journal of Educational Technology Systems. 2020;49(2):170-198. doi:10.1177/0047239520926971

[57] ISO 9241-210: Ergonomics of human-system interaction- Part 11: Usability: Definitions and concepts, International Organization for Standardization, Geneva (2018)

[58] Chou, Chientzu Candace, Lanise Block, and Renee Jesness. "A case study of mobile learning pilo project in K-12 school " Journal of Educational Technology Development and Exchange 5.2, 2012, 11-26.

[59] EyalEshed, On Designing Mobile Education Apps,citizentekk, January 2014

[60] Seffah, Ahmed, Mohammad Donyaee, Rex B. Kline,andHarkirat K. Padda. "Usability measurement and metrics: A consolidated model." Software Quality Journal 14.2, 159-178, 2006.

[61] Kunjachan, Mary Ann Chiramattel. "Evaluation of Usability on Mobile User Interface." University of Washington, Bothell, 2011.

[62] Bevan, Nigel. "International standards for HCI and usability." International journal of human-computer studies 55.4, 533-552, 2001

[63] Hussain, Azham, and Maria Kutar. "Usability metric framework for the mobile phone application." PGNet,ISBN, 2009, 978-1

[64] Caldiera, V. R. B. G., and H. Dieter Rombach. "The goal question metric approach." Encyclopedia of software engineering 2.1994, 528-532

[65] J. W. CRESWELL, Research design: Qualitative, quantitative, and mixed methods approaches Sage publications, 2013.

[66] Shapiro, R. B., and Ahrens, M. (2016). Beyond blocks: Syntax and semantics. Communications of the ACM, 59(5), 39-41.

[67] P. DOURISH and G. BUTTON, ”On" ethnomethodology": Foundational relationships between ethnomethodology and system design.,” Human-computer interaction, vol. 13, nr 4, pp. 395- 432, 1998.

[68] C. HEATH and P. LUFF, ”Collaboration and control: Crisis management and multimedia technology in London Underground Line Control Rooms.,” Computer Supported Cooperative Work (CSCW), vol. 1, nr 1-2, pp. 69-94, 1992.

[69] W. E. MACKAY, ”Is paper safer? The role of paper flight strips in air traffic control.,” ACM Transactions on Computer-Human Interaction (TOCHI), vol. 6, nr 4, pp. 311-340, 1999.

[70] B. B. KAWULICH, ” Participant observation as a data collection method,” Forum Qualitative Sozialforschung/Forum: Qualitative Social Research, vol. 6, 2005.

[71] C. D and C. B, ”Qualitative Research Guidelines Project,” 07 2006. [Online]. Available: http://www.qualres.org/HomeSemi-3629.html. [Använd 17 03 2017].

[72] J. SMITH and J. FIRTH, ”Qualitative data analysis: the framework approach,” Nurse researcher, vol. 18, nr 2, pp. 52-62., 2011.

[73]. Scratch.: https://scratch.mit.edu/. Accessed July 26, 2020

[74]. Hora del Código Chile.: http://www.horadelcodigo.cl/. Accessed July 25, 2020

[75]. Brooke, J.: SUS-A quick and dirty usability scale. Usability Eval. Ind. 189(194), (1996)

[76]. Morales J., Botella F., Rusu C. Quiñones D.: How “Friendly” Integrated Development Environments Are?. In: Meiselwitz G. (eds) Social Computing and Social Media. Design, Human Behavior and Analytics. HCII 2019. Lecture Notes in Computer Science, Vol. 11578. Springer, Cham. https://doi.org/10.1007/978-3-030-21902-4_7 (2019)

[77] M. B. KINZIE and D. R. JOSEPH, ”Gender differences in game activity preferences of middle school children: implications for educational game design.,” Educational Technology Research and Development, vol. 56, nr 5-6, pp. 643-663, 2008.

[78] S. IMEL, ”Teaching Adults: Is It Different?,” ERIC Digest No. 82, 1989.

[89] L. Cardellini, ”An Interview with Richard M. Felder Entrevista Con Richard M. Felder,” Journal of Science Education, vol. 3, nr 2, pp. 62-65, 2002.

[80] D. TRAYNOR and P. GIBSON, ”Towards the development of a cognitive model of programming: a software engineering approach,” i Proceedings of the 16th Workshop of Psychology of Programming Interest Group, 2004.

[81] S. M. M. RUBIANO, O. LÓPEZ-CRUZ and E. G. SOTO,” Teaching computer programming: Practices, difficulties and opportunities,” i Frontiers in Education Conference (FIE), 2015.

[82] C. C. Stephanidis and G. Salvendy, "Seven HCI Grand Challenges", International Journal of Human-Computer Interaction, vol. 35, Number 14, 2019 pp. 1229-1269.

[83] H. Tsukamoto, Y. Oomori, H. Nagumo, Y. Takemura, A. Monden and K. Matsumoto, "Evaluating algorithmic thinking ability of primary schoolchildren who learn computer programming," 2017 IEEE Frontiers in Education Conference (FIE), Indianapolis, IN, 2017, pp. 1-8.

[84] “Statutory guidance National curriculum in England: computing programmes of study,” [Available] https://www.gov.uk/government/publications/national-curriculum-inengland-computing-programmes-of-study/n.

[85] J. E. Sammet, "Programming Languages: History and Future" , Communications of the ACM, vol. 15, Number 7, 1972, pp. 601-610.

[86] N. Bak, B. Chang and K. Choi, "Smart Block: A Visual Programming Environment for SmartThings," 2018 IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC), Tokyo, 2018, pp. 32-37.

[87] H. Tsukamoto et al., "Textual vs. visual programming languages in programming education for primary schoolchildren," 2016 IEEE Frontiers in Education Conference (FIE), Erie, PA, USA, 2016, pp. 1-7.

[88] T. Karvounidis, I. Argyriou, A. Ladias and C. Douligeris, "A design and evaluation framework for visual programming codes," 2017 IEEE Global Engineering Education Conference (EDUCON), Athens, 2017, pp. 999-1007.

[89] B.J. Smith, Harry S. Delugach, "Work In Progress - Using a Visual Programming Language to Bridge the Cognitive Gap Between a Novice’s Mental Model and Program Code", 2010, Washington, DC 40 th ASEE/IEEE Frontiers in Education Conference.