**University of Derby**

**Department of Electronics, Computing & Mathematics**

A project completed as part of the requirements for

BSc (Hons) Computer Games Programming

entitled

**Games – Effectiveness of Teaching Entry Level Programming**

by

**Adam Ballinger**

a.ballinger1@unimail.derby.ac.uk

**April 2017**

# Abstract

Programming is an intimidating subject to break into, and existing teaching techniques don’t always consider different learning styles. There is an increase in the number of drop outs for computer science related courses, primarily because of students lacking either motivation or the ability to understand the complex and difficult material involved in learning programming.

Games have shown their ability to teach programming and engage students in ways that typical teaching techniques can’t. This project produces a new game that is designed to teach the basics of programming to complete beginners, and to determine if games could be used as an additional teaching technique in schools.

Once the experiment concluded, this project determined that games do have the potential to become a supplementary component in teaching environments for teaching programming.

Table of Contents

[Abstract 1](#_Toc481170744)

[1. Introduction 4](#_Toc481170745)

[1.1 Project Aim 4](#_Toc481170746)

[1.2 Objectives 4](#_Toc481170747)

[1.3 Rationale 4](#_Toc481170748)

[1.4 Hypothesis 4](#_Toc481170749)

[2. Literature Review 6](#_Toc481170750)

[2.1 Introduction 6](#_Toc481170751)

[2.2 Teaching Techniques 6](#_Toc481170752)

[2.2.1 Why Technique Matters 6](#_Toc481170753)

[2.2.1.1 Lectures 6](#_Toc481170754)

[2.2.1.2 Scaffolding Technique 6](#_Toc481170755)

[2.2.1.3 Concept Mapping 7](#_Toc481170756)

[2.2.1.4 Constructivism Technique 7](#_Toc481170757)

[2.3 Learning Styles 7](#_Toc481170758)

[2.3.1 Why Learning Style Matters 7](#_Toc481170759)

[2.3.1.1 Active Learners 8](#_Toc481170760)

[2.3.1.2 Sensing Learners 8](#_Toc481170761)

[2.3.1.3 Visual and Verbal Learners 8](#_Toc481170762)

[2.3.1.4 Sequential Learners 8](#_Toc481170763)

[2.4 Existing Software for Teaching Programming 9](#_Toc481170764)

[2.4.1 Scratch (scratch.mit.edu, n.d.) 9](#_Toc481170765)

[2.4.2 Pex4Fun (pexforfun.com, n.d.) 10](#_Toc481170766)

[2.4.3 Code Combat (codecombat.com, n.d.) 10](#_Toc481170767)

[2.5 Conclusions and Summary 11](#_Toc481170768)

[2.5.1 Issues 11](#_Toc481170769)

[2.5.2 Existing Software 11](#_Toc481170770)

[3. Methodology 12](#_Toc481170771)

[3.1 Introduction 12](#_Toc481170772)

[3.2 Project Strategy 12](#_Toc481170773)

[3.2.1 Unity 12](#_Toc481170774)

[3.2.2 Moon-Sharp 12](#_Toc481170775)

[3.2.3 Lua 13](#_Toc481170776)

[3.3 Project Design and Development 13](#_Toc481170777)

[3.3.1 Design 13](#_Toc481170778)

[3.3.2 Development 14](#_Toc481170779)

[3.4 Data Gathering 14](#_Toc481170780)

[3.5 Data Analysis 15](#_Toc481170781)

[3.6 Conclusions 15](#_Toc481170782)

[4. Findings and Analysis 16](#_Toc481170783)

[4.1 Introduction 16](#_Toc481170784)

[4.2 Findings 16](#_Toc481170785)

[4.2.1 Results and Basic Analysis 16](#_Toc481170786)

[4.2.2 Further Analysis 19](#_Toc481170787)

[5. Discussion 20](#_Toc481170788)

[5.1 Introduction 20](#_Toc481170789)

[5.2 Questionnaire Results 20](#_Toc481170790)

[6. Conclusions and Recommendations 23](#_Toc481170791)

[6.1 Introduction 23](#_Toc481170792)

[6.2 Conclusion 23](#_Toc481170793)

[6.3 Recommendations 23](#_Toc481170794)

[6.4 Future Work 23](#_Toc481170795)

[6.4.1 Automatic Code Formatting 23](#_Toc481170796)

[6.4.2 Syntax Highlighting 24](#_Toc481170797)

[6.4.3 Integrated Level Editor 24](#_Toc481170798)

[7. References 25](#_Toc481170799)

[8. Appendices 27](#_Toc481170800)

[8.1 Appendix 1 27](#_Toc481170801)

[8.2 Appendix 2 28](#_Toc481170802)

[8.3 Appendix 3 29](#_Toc481170803)

# Introduction

## Project Aim

The aim for this project is to create a game that serves the purpose of teaching somebody with little, to no prior experience in programming, the very basics of Lua which the user will then use to complete several tasks. Feedback from participants will then be used to evaluate how effective they found the game in teaching the fundamentals of programming, and whether these kinds of games could potentially serve as an alternative to teaching people to write code. Even though the project aims at teaching somebody a programming language (Lua), the concepts that this project aims to teach can be carried over to other programming languages such as C#, C++, Python etc.

## Objectives

1. A literature review will be done to find out about existing research in teaching beginners how to program, how effective current methods are the different styles of learners there are. The review will also cover existing software used for teaching programming and how they work.
2. Research will be carried out on existing solutions to find effective ways of approaching the implementation of the project.
3. A game will be designed and developed using the Unity game engine and existing C# Lua interpreter “Moon-sharp”, which will then be given to a group of volunteers that will play through the game, and provide feedback on how effective / useful they found the game.
4. The game will feature a set number of levels that each teach the user a new concept of programming which they must then use to complete a task. Each level may or may not require the user to utilize something they learnt from a previous level.
5. Using feedback from participants, a discussion on the findings will be carried out to determine if the project serves as an effective solution to teaching, and compare these results to existing research to determine if the concept of using games for teaching somebody how to program is a viable solution.

## Rationale

For someone that has never done programming before, it can be overwhelming when trying to understand the basics of programming through resources such as books or online tutorials. A game can potentially make that experience more enjoyable and be potentially more effective at teaching those basics to somebody over conventional methods. So, the reason for this project is to attempt at making that initial experience of beginning to program easier and more enjoyable for the user.

## Hypothesis

Learning to program can be an intimidating process for a beginner. People may be put off by the overwhelming amount of work and reading that needs to go into learning how to write code. Using games to provide an interactive learning experience to somebody starting to learn how to program can help with introducing someone to the basic concepts of writing code, and improve the learning experience of the user. This paper will look to prove whether using games are an effective tool for teaching beginners how to code, and whether they can be used as a supplementary component in classrooms.

# Literature Review

## Introduction

This section of the paper will cover current techniques used for teaching entry level programming their effectiveness. It will also cover different learning styles, and how each style influences someone’s ability to learn the entry level material for programming, and how the different teaching techniques supplement these different styles. It will then review existing software used for teaching programming and how effective they are in teaching programming to beginners.

## Teaching Techniques

### Why Technique Matters

Teaching technique influences how effectively a student can take in and understand new material. The authors of the paper, *The Impact of Learning Styles in Introductory Programming Learning*, identify that there is a global issue with beginners learning the introductory levels of programming, and that teachers from around the world have attempted many different strategies to try to reduce the difficulty students face, but with no success (Carmo, et al., 2007). The paper also identifies that it is an important to look for new methods of aiding the teaching of introductory programming to new students, which is what this project aims to achieve by finding out if games could be an effective tool in supporting teaching.

### Lectures

Lectures are the traditional technique for teaching most subjects (Duffany, 2017). A lecture usually consists of a single lecturer who will speak to a room of people typically as a one-way conversation about specific subject material prepared by that lecturer (Duffany, 2017). Although lectures are the most commonly used teaching technique, they are often criticised for their lack of audience participation, and a lack of motivation for students to expand on the material which they have learnt outside of the lecture (Duffany, 2017). Even though lectures are criticised, no effective alternative techniques have been found that could replace them. Marc J. Rubin in his paper on the *Effectiveness of Live-Coding to Teach Introductory Programming*, suggests the use of different teaching strategies including scaffolding, concept mapping and constructivism.

### Scaffolding Technique

Scaffolding is a technique where teachers will provide support to students and progressively reduces that support as students advance in their understanding on a subject, building on their ability to solve problems independently (Rubin, 2013).

Scaffolding can appear in two different forms; Either soft or hard scaffolds (Simons & Klein, 2005). Soft scaffolds refer to the actions of the teacher and how they provide support to a student when they have a specific need (Simons & Klein, 2005). As an example, teachers will actively question students on their understanding of a subject, and then provide them with feedback on their progress. Hard scaffolds are supports created by a teacher in advance to help provide a stronger explanation of a topic that would typically be harder for students to understand (Simons & Klein, 2005).

### Concept Mapping

Concepting mapping is the technique of using alternative means of conveying information to students using things like graphical representations (Rubin, 2013).

### Constructivism Technique

Constructivism involves students being required to construct their own understanding of something as they are taught, and allows their teacher to correct any misunderstood concepts where necessary (Rubin, 2013).

## Learning Styles

Different people learn in different ways, which directly impacts the effectiveness of various teaching techniques (Thomas, et al., 2002). Richard M. Felder and Linda K. Silverman in their paper, *Learning and Teaching Styles in Engineering Education*, created a model that displays the various learning styles and the corresponding teaching style that is most effective in supplementing that given style of learning (Felder & Silverman, 1988). This model is known as the Index of Learning Style (ILS) (Felder, n.d.), and is a commonly used model in other research related to learning styles and teaching techniques.

### Why Learning Style Matters

The style in which somebody learns is an influential factor in how effective different teaching techniques are for any given individual. Our learning styles define how we process information on a given subject (Norwawi, et al., 2009). In the paper, *The Impact of Learning Styles in Introductory Programming Learning*, the authors state that they believe it is important that the learning styles of each student is taken into consideration when planning classroom learning activities to help each student maximize their learning capability (Carmo, et al., 2007). The problem with this is that it is difficult for teachers to create material for a single class that caters to multiple learning styles. This leads to teachers being forced to use a single piece of material for every student, which may be adequate for some, but not for others (Carmo, et al., 2007).

### Active Learners

Active learners learn through practical tasks and working with others to achieve a goal (Thomas, et al., 2002), and is “… considered to be any form of learning where the student is doing something besides just listening.” (Duffany, 2017, p. 1 Section 3). Common examples of active learning are the use of practical exercises that puts a student in a situation where they need to complete a physical task to learn about a specific topic, or to complete a project outside of a classroom to reinforce their ability to learn about something other than simple listening to a teacher/lecturer (Felder & Brent, n.d.). Writing programs and discussing subject material with peers are the effective examples of how active learners prefer to learn (Norwawi, et al., 2009). These kinds of examples reinforce the idea that active learning is an effective learning style in increasing performance in a student’s ability to progress in their education in programming, as ‘learn through doing’ is a believed to be the most effective method of learning how to program.

### Sensing Learners

Sensing learners prefer to be taught through facts, solving problems through established methods, and being tested on material that has been covered explicitly in class (Felder, n.d.). Sensor learners are believed to understand information best if they are shown examples of how what they are being taught connect to the real world (Felder, n.d.). An example of this in teaching programming would be using code examples of a programming concept or algorithm. Sensing learners tend to have a good memory, focus much easier to the content being taught, and ability to problem solve well (Yeh & Yen, 2015).

### Visual and Verbal Learners

Visual learners prefer to be taught through graphical representations of subject material, such as diagrams, figures, and flow charts (Carmo, et al., 2007). Verbal learners learn best through words, including both written and spoken explanations of a topic (Felder, n.d.). Anybody that is not impaired visually or has difficulty hearing learns visually and verbally, and in some cases, may have a preference to one of the other (Felder, n.d.).

### Sequential Learners

Sequential learners best learn from ordered steps where the next step logically expands from the previous, progressively making a clearer understanding of a given subject (Zualkernan, et al., 2006). A sequential learner is more likely to expand their understanding on a topic using books, and teaching strategies used by their teachers that are aimed to their specific style of learning (Carmo, et al., 2007). Based on this assumption, sequential learners wouldn’t benefit from what the project aims to achieve which is to determine if games could be used a supplementary component in a classroom, but based on the fact the project aims to teach concepts in a linear fashion, where each step expands from the previous, sequential learners could still easily benefit from the work done in this project.

## Existing Software for Teaching Programming

### Scratch (scratch.mit.edu, n.d.)

Scratch is a visual programming software designed primarily for students in the age group of 8 to 16 (scratch.mit.edu, n.d.), and is the most popular software used by schools for teaching beginner programming to students. Scratch can be used to create a variety of multimedia applications such as games, simulations, and animated stories, achieved through connecting various command blocks together to create a set of instructions (Maloney, et al., 2010).

Scratch is an effective tool for teaching because of its ability to allow learning through experience and experimentation, without the need to worry about producing errors (Maloney, et al., 2010). This is because of the block design in which users utilize to create their applications. Scratch can be compared to LEGO bricks, which don’t give error messages, it restricts you to piecing parts together in specific ways (Maloney, et al., 2010). This doesn’t necessarily mean though that code is completely error free, as connected blocks can still not produce intended results.

In the paper, *Habits of Programming in Scratch*, the authors state that they found students using scratch only actually learnt programming concepts when they had been explicitly taught them by their teacher, and not through their own experimentation and exploration of the scratch software (Salant, et al., 2011). They also state that teaching technique and supporting learning material is an important factor in maximising the learning capabilities of students using scratch (Salant, et al., 2011).

Even though scratch is a powerful tool for teaching the basics of programming, it doesn’t necessarily teach good practices, and leads to users creating code that is difficult to read, maintain, and debug in the event any problems arise (Moreno & Robles, 2014). Moreno & Robles identify issues with naming conventions for things such as sprite names not being changed from their default “Sprite X” names, and repetitive code being the biggest issues with scratch applications that they had analysed.

Although scratch itself is not a game, it does allow people to create their own games, which could themselves be used as methods for teaching programming, or any other subject. Scratch in comparison to this project, does share the same goal which is to teach beginner programming, however this project aims to expose users to writing physical code, rather than connecting blocks together.

### Pex4Fun (pexforfun.com, n.d.)

Pex4Fun is a browser based educational software, designed to teach entry level to advanced programming and software engineering in schools to graduate tier courses (Halleux, et al., n.d.). The Pex4Fun software features something called coding duels, where players are given a puzzle that is created by another player, which you must solve to reproduce the same functionality as the author (Beste, n.d.). Having competitive features like this can help in providing motivation to users to completing these challenges; The more challenges someone completes, the more they learn.

The authors of the paper, *Teaching Computer Science in a Web-Based Environment*, conducted an experiment using Pex4Fun on undergraduate students, where they were given Pex4Fun to find out whether they found it an effective tool in aiding their learning ability. This experiment found that students who used the Pex4Fun, found it easy to use, and helped with motivating them to work, but were undecided on whether the software was a suitable tool for learning (Čisar, et al., 2013).

### Code Combat (codecombat.com, n.d.)

Code combat is a 2-Dimensional RPG game that aims to teach people as young as the age of 6, how to program in several different languages by writing actual code (codecombat.com, n.d.). Code combat compared to the software reviewed above, is a product that is most like this project in terms of intended functionality and how to project aims to teach its users.

Code Combat teachers by progressively introducing new programming concepts to the user (things like variables, loops, and conditionals) as they progress through each stage (Jemmali & Yang, 2016) (See section 2.2.1.2). Code combat is suited well to new programmers because of its smooth learning curve; where each level becomes more complex as they progress with the introduction of more objects in each stage, such as traps and fences, as well as additional player mechanics such as special attacks and abilities (Jemmali & Yang, 2016).

Code Combat provides users with a tutorial for each stage on the available methods they can use. Each level then requires the user to create a solution to the level by either simply collecting a gem, or completing a set objective, such as killing all enemies or obtaining certain objects such as keys. Certain levels will also require the user to complete the level in a set number of lines of code using certain techniques such as using loops to perform large amounts of repetitive movements, instead of writing a large list of move commands. To keep motivation for the user to complete each level, every level has several hints that the user can access on how to complete the required objectives.

## Conclusions and Summary

This section has identified different teaching techniques and learning styles in programming education, as well as covered existing software that is used for teaching beginners programming. From this review, the project does seem feasible given the success of existing software used today, such as scratch and code combat.

## Issues

For beginners, understanding the basics is the hardest part of getting into programming (Corral, et al., 2013). This is primarily linked to different learning styles not all being supplemented by existing teaching techniques. The aims for this project are to produce a game, that will serve as a teaching tool that can potentially be used a long side existing teach techniques to further improve the learning ability of students.

## Existing Software

Existing software used in teaching environments, such as scratch, is primarily aimed at teaching young learners, specifically primary school students, and doesn’t directly expose those students to writing actual code, and teaching them about correct programming syntax. Scratch for example, is limited to allowing users to simply connect blocks together. Although this is great for improving users computational thinking skills and problem solving skills, it doesn’t teach them how to write actual code, which will be a requirement in the future if they wish to expand their ability.

This project will serve the purpose of teaching users how to write code from the start, whilst also teaching the user about the basic concepts of programming (loops, variables, and objects), as well as programming syntax where necessary.

# Methodology

## Introduction

This section of the paper will cover the proposed solution to a new game that will be designed to teach beginner level programming using Lua. It will also discuss the intended data gathering and analysis methods that will be used to determine if the project proves or disproves the project hypothesis.

## Project Strategy

The project will follow the Agile development methodology. The agile methodology principles allow for change in design and requirements throughout the development life cycle (agilemethodology.org, 2008), which is most likely to occur during the development of each level in this project.

## Unity

The Unity game engine (Unity Technologies, 2017) was used for the development of the projects game. The reason for using Unity is because of its developed 2D development features that make the design and development of core gameplay features fast and easy, and the UI framework that makes designing and implementing UI functionality very easy, which will be an important part of the project.

Unity also supports a wide range of different platforms, which allows the game to be built for multiple platforms (Unity Technologies, 2017), primarily Windows. Unfortunately, a Web GL distribution will not be possible because of compatibility issues with the Moon-sharp Lua interpreter and Web GL.

## Moon-Sharp

The Moon-Sharp Lua interpreter (moonsharp.org, n.d.), was used within the Unity game engine to allow the written Lua code to be interpreted by the C# API written with Unity’s scripting framework. The reason the project used Moon-Sharp was because of how it is designed to be fully compatible with the Unity game engine, which requires the Moon-Sharp library to be compatible with the Mono framework (moonsharp.org, n.d.). Moon-Sharp also provides full support for the Lua standard library, and easy to use error handling for interpreted Lua code, as each error in the written Lua code will throw a dedicated exception in C# (such as syntax or interpreter errors) (moonsharp.org, n.d.), making the implementation and testing process of integrating the interpreter into the game a lot faster and easier.

Compatibility with Unity was an important requirement when searching for an interpreter to use. Moon-Sharp along with its full compatibility with the Mono framework (Crucial for Unity). The library also allows you to bind Unity types to the interpreter to be used in Lua (moonsharp.org, n.d.). This allows the Lua code to access Unity object properties much easier, making the process of linking the character API in C# with the Lua interpreter very straight forward.

## Lua

The game developed in this project uses the Lua programming language (lua.org, n.d.) because of its simplicity and easy to understand syntax compared to other languages, such as Python. Lua is also the current leading scripting language for games (lua.org, n.d.), making it a good choice of languages to teach.

Regarding the syntax of Lua, users won’t need to worry about the use and positioning of common symbols such as semi-colons or braces, and only need to worry about closing statements with keywords such as “end”.

There are however parts of the Lua design that may be misleading to the user when transitioning to other languages. Arrays starting at index 1, rather than 0 in Lua (lua.org, n.d.) is an important thing to convey to the user where accessing array index’s is necessary. Where this occurs in the final game, it will be noted that in other languages, arrays are 0 indexed.

## Project Design and Development

## Design

The game will feature five levels, each containing a tutorial that teaches the user a new concept of programming through Lua code. Each level will then require the user to utilize what they learn in the tutorial, and possibly previous tutorials from past levels, to move the player square through a maze. The game will start of very easy, requiring the user to only make the player move in basic directions, and will progressively increase in difficulty requiring the user to achieve level objectives, such as completing the task in a set number of lines of code, or using specific techniques.

The UI will consist of an input field, which will be where the user writes the Lua code they need to complete the level; An error box that will display where any errors are in their written Lua code; There will also be two buttons used for starting and stopping the Lua execution.

Each tutorial will feature a demonstration to assist in the explanation of what is being taught, and hints will be provided in the input field in the form of Lua comments to provide an extra layer of help; Some levels will also provide some starting code to get the player started. This was done to help the user understand the material taught as much as possible.

When the user presses the play button, their code will be checked before running for errors in the code they have written. The code they have written however, will not be checked to determine if it can successfully reach the goal of the level; It will however check for script constraints set for each level, such as line requirements. Levels that include line restrictions are put in place to force the user into making use of what they have learnt in the tutorial, rather than brute forcing the level with simple move commands.

## Development

The development of this project will follow the Agile methodology (As mentioned in section 3.2). The functionality of the project will be implemented as described in the above section.

Testing will be carried out during throughout the development of the project, testing the functionality of each feature implemented. This will help with speeding up development by decreasing the number of major bugs in the project as it is developed, reducing potential bug counts once the development has finished. Once development has been completed, test cases will be used to ensure the final functionality is operating as intended, and that the game plays without issue. The test case results can be found in the appendix section (Appendix 2).

GitHub will also be used for version control during the development of the project, to keep track of file changes and act as a backup for the project.

## Data Gathering

Data for this project will be gathered using quantitative methods, through an online questionnaire given to people that play through the game; The questionnaire will be created using Google forms (google.co.uk, n.d.).

The Google forms questionnaire URL will be given to participants of the project to fill out online, rather than a paper based approach, to allow the raw data to be translated into various graphs. This will also help in keeping questionnaire results safe and accessible.

The questionnaire asks participants questions that are either multiple choice, linear scale based (1-5), or require a small amount of writing. The questions are based on the experience they had with the game, and their opinions on its effectiveness; See the appendix section (Appendix 1) for a full preview of the questionnaire.

The target audience of this project is any individual of any age that has either little, to no prior programming experience. This is to ensure the data generated by this project is accurate, and matches the aims of this project, which is to teach beginners the basics of programming (See section 1.1).

Participation in the questionnaire will be anonymous and only the participants email address will be recorded when filling out the questionnaire; Their email address will not be used in any of the data analysis. The only requirement to fill out the questionnaire will be to have played the game developed in this project; Completion of the game will not however be a requirement.

## Data Analysis

Raw data from the google forms questionnaire will be translated into several different graphs, such as bar charts and pie charts, to create more readable and understandable results that can then be further analysed and evaluated.

## Conclusions

The design of the game described in the above section (Section 3.3.1) fits the aims of this project well. The aims of the game are to teach beginner level programming through direct teaching of the basics through an interactive game. The design of this project’s game will teach each new concept in a separate level where the user needs to navigate through a maze using what they have learnt.

Most existing software (See section 2.4), aims to teach beginners the basic concepts of programming, but not through writing code, but instead joining blocks together or linking nodes, whereas this project will introduce users to writing code from the start. Now that the game has been developed, the next stage in this project will be to now hand the game over to willing participants that match the target audience (See section 3.4), and gather their opinions on the software using the questionnaire also mentioned in the above section (Section 3.4).

# Findings and Analysis

## Introduction

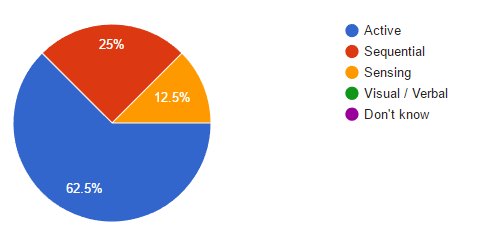
This section will cover and show the findings from the questionnaire given to participants of the produced game for this project, and discuss the analysis done on those findings.

## Findings

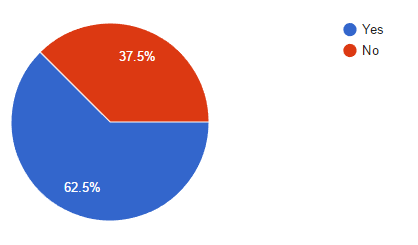
Using Google forms for the questionnaire, I used the built-in analysis tools to produce graphs that are relevant to the type of question asked.

### Results and Basic Analysis

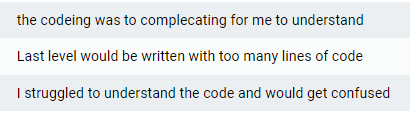
*Q1. What style of learner are you? (8 responses)*



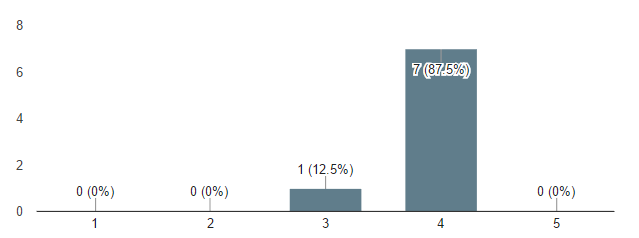
*Q2. Did you manage to complete each level? (8 responses)*



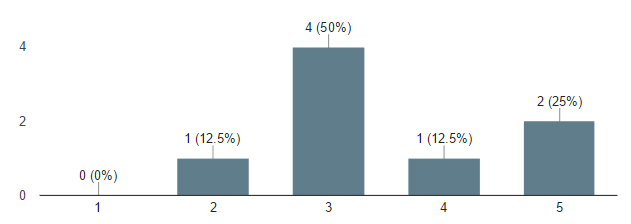
*Q3. If you answered no to the able question, please give a short reason as to why you did not finish. (3 responses)*



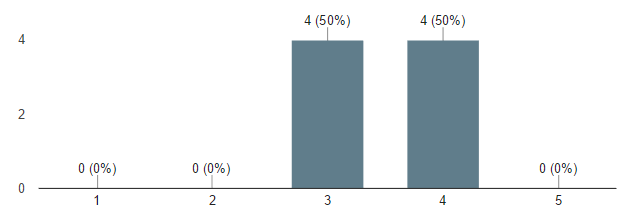
*Q4. How enjoyable did you find the game? (8 responses)*



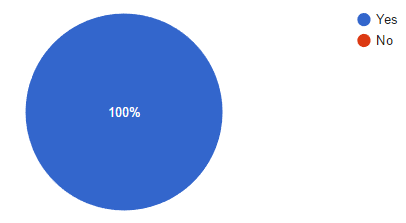
*Q5. How helpful did you find the tutorial for each level? (8 responses)*



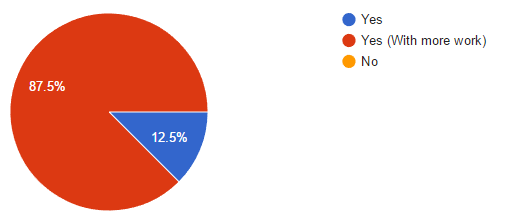
*Q6. How clearly do you feel the game explained what you needed to do? (Explanation of code etc.) (8 responses)*



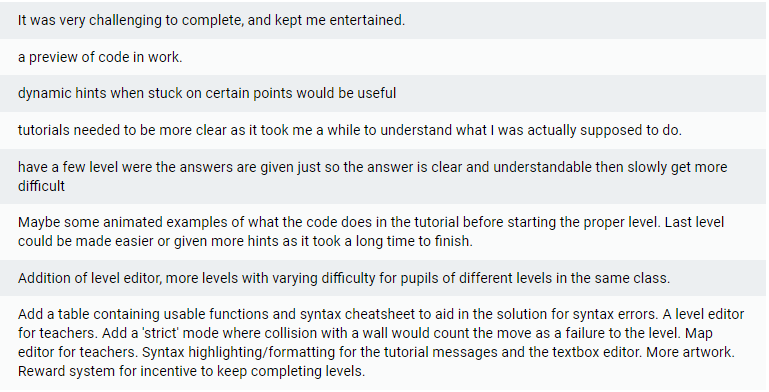
*Q7. Do you think the game, or others like it, could provide better motivation to students learning how to program? (8 responses)*



*Q8. Do you feel the game could be used as a teaching tool? (8 responses)*



*Q9. Finally, do you have any additional comments about the game? (What could be improved, etc.) (8 responses)*



### Further Analysis

Additional analysis was done on the raw data produced from the questionnaire to allow for a more in depth evaluation and comparison of the data. This analysis was done using Excel to pair related questions together.

Analysis shown here compares the data from Q1 and Q2 (See Appendix 1), which compares the number of completions and incompletions between the different learning styles.

# Discussion

## Introduction

This section of the paper will discuss the results and analysis from the previous section, and evaluate whether this project has achieved its aims / objectives, and if the project hypothesis has been met.

## Questionnaire Results

The results produced from the questionnaire met expectations. The data has given a clear indication to how students feel about a game being used as a teaching tool and how they feel they could be improved.

From the results of Q1, it shows a large bias to the active learning style (62.5%), compared to the low percentage of sequential (25%) and sensing learners (12.5%). Active learners, based on the research undertaken in the literature review above, are the individuals expected to perform better than others, followed by sequential and sensing learners (Section 2.3). From the further analysis undertaken, comparing game completions and learning styles, it is (Section 4.2.2), shows that active and sequential learners are the only styles to have been able to successfully complete the game. This data also shows that statistically, sequential learners are the most successful learners (100%) followed by active learners (60%). This is not a surprising result considering sequential learners prefer being tested on material, which is partly what the project aims to do by testing the user on what they learn through each level.

Based on the results of Q2, if participants were unable to complete the game, they were asked to comment on why they were unable to finish the game. From the results of this question, a common occurrence in the comments provided were that they struggled to understand the code required to complete the level they had reached. There was also a comment about the number of lines the participant felt they needed to use would exceed the level they were stuck on allowed. This result is based both on the participant’s interpretation of the material taught in each level, and how the material was explained. It is understandable for some participants to struggle to understand the material being taught without the presence of a teacher to support. This is partly why this project isn’t designed to act as a replacement to existing teaching techniques used today, but to instead provide additional means of explaining content to students.

Analysis of Q4, which asks participants to rate how enjoyable they found the game shows that most participants found the game enjoyable with the majority (87.5%) giving a rating of 4 of 5. An important factor of games being an effective teaching tool is ensuring that they are enjoyable to the user. If the game isn’t enjoyable, it will struggle to grab the full attention of students.

Q5, which asks participants to rate the helpfulness of each level tutorial, produced mixed results, with the majority (50%) rating the helpfulness a 3 of 5. Considering the tutorial is the part of the game where the user is taught what is necessary to complete the level, it is important that it appropriately explains what needs to be done and is written so that it is easy to understand. Based on the results, it seems the tutorial for each level could have been better to make the explanations easier to understand. In fact, Q9 asks for participants to comment on what they feel could have been improved about the game, one participant said the following:



This can be interpreted as making the level objectives clearer (what you need to do to complete the level), or the actual programming tutorial. Either way, both are required to be clearly stated and in a way that a beginner programmer will understand.

Q6 expands slightly from Q5, which asks the participant to rate how they clearly the game explained what you needed to do. The focus of this question was aimed towards how the game explained overall objectives, and not just the programming tutorial part of the game. The results show that participants felt that the general objectives of the game were explained in an acceptable way, with room for improvement.

It is important that the game provides additional motivation to beginners learning how to program; Students that are motivated are more likely to pick up programming faster. Q7 asks participants to answer yes or no to whether they think the game produced in this project, or other games that exist like this one, could provide better motivation to students learning how to program. 100% of participants voted yes to this question. This shows that games could help with reducing the drop out percentage of first year programming students at University, with motivation to learn material being the biggest reason for drop outs.

Participants were also asked to answer yes, yes (with more work), or no to whether they think the game could be used as a teaching tool. 87.5% of participants voted yes (with more work) to this question, and 12.5% voting just yes. Based on these results it’s clear the game requires additional work before it could be used as a teaching tool, but the fact that participants feel it could be used as a teaching tool goes a long way towards proving the hypothesis of this project.

The final question asked participants to comment on what the thought could be improved. There was a range of feedback providing general comments on what they thought of the game, and what they feel could be added to improve the game. An interesting suggestion from one participant was:



This is a reasonable suggestion for the game as it would potentially increase the number of successful completions of the game by supporting students with small hints on how to complete parts of the level. It would be important however to ensure the hints wouldn’t make the solution to each level too obvious and keeps the game challenging. Hints could be provided in the form of popup examples related or like the objective of the level they are currently on. For example, whilst observing participants playing the game, all of them struggled to understand the requirements of level 3 (Appendix 3). Understanding how while loops worked was the issue with this level, as the example provided during the tutorial didn’t explain in detail enough how commands written inside a while loop worked. Participants would typically only write movement commands once, thinking that would be enough. As a dynamic hint for this level, an example using a different maze style could have been shown to show how movement commands behave inside the while loop.

Another participant answered the following to this question:



This is a very good idea as it would ease the user into the game, rather than throwing them straight into programming from the first level. Basic examples, such as moving the player in simple directions and then progressing into more advanced movements would probably help demonstrate how code is written and what the player needs to write to do specific actions.

Another participant also commented the following:



This would work nicely with the suggestion about dynamic hints, as animated examples could be provided inside of these hints to visually demonstrate what code is doing. This comment also shows that participants potentially found the tutorial for each level confusing or poorly written.

Based on the results of the questionnaire, and the discussion of those results in this section, although the project developed for this project may not be a perfect example of using games for teaching, it still shows that participants enjoyed the experience and agreed that games have the potential to be used as teaching tools in a classroom. Therefore, the hypothesis for this project was correct.

# Conclusions and Recommendations

## Introduction

This section of the paper will conclude the work carried out in this project, and make recommendations to how the project could be improved, and finally any additional work / research that can be carried out in the future to aid in proving the project hypothesis.

## Conclusion

Overall, based on the research carried out in this project, it demonstrates that games can be used as effective teaching tools if well designed. Games can help in increasing student motivation, which is a primary reason for student drop outs in computer science courses.

The developed solution proposes by this paper has achieved the aims and objectives set in this paper. The solution however in its current state would require additional work in before it could be used in a teaching environment based on the feedback received from participants of this study. The solution developed provides a strong starting point however for future work, and if additional fixes and features are implemented, the game could be used as a teaching tool in classes, and prove the project hypothesis.

## Recommendations

The game requires further improvements in the form of tutorial quality and gameplay flow. Current tutorials have proven to be not as helpful as intended based on the results of the questionnaire in the above section. To improve these tutorials more research should be done into how programming is taught to beginners, and ways of explaining the concepts to beginners in a way that is easy to understand.

Gameplay flow also needs improving, as mentions by one of the participants in Q9. The game currently doesn’t count movement commands that would move the player into a wall as a failure, and make it possible to create solutions that are unintended to work. This will require changes to the character C# API to reset the level if the player moves into a wall.

## Future Work

### Automatic Code Formatting

Automated formatting of the code written by the user would help in making the code written more readable, and introduce the user to programming writing standards. Formatting should include automated tab insertion on new lines where necessary inside of statements, and spacing between symbols. Further research will need to be carried out to figure out how to implement this functionality into Unity’s input fields.

### Syntax Highlighting

Syntax highlighting would also make reading code easier. Currently, all code, including comments are displayed in the same colour. Highlighting keywords and statements would make the overall readability better. Research will need to be carried out to figure out how this functionality could be implemented effectively within Unity input fields. Rich text would be a starting point for this implementation.

### Integrated Level Editor

One suggestion made by a participant was to create a level editor (Section 4.2.1, Question 9). During the development of this project, a level editor was developed and used to create the levels found in the game, but lacked the ability to design tutorials for the levels, and each tutorial had to be manually written in the Unity editor. As an improvement to the existing project, packaging the level editor with the distributed game, and implementing a system that allowed custom tutorials to be written for each level would make the game much more useful to teachers, and would allow them to create their own levels for the content they wish to teach very quick and easy. Another limitation however would be the character API that is written in C#; This would need to be expanded upon to make more complex interactions with the game.

# References

agilemethodology.org, 2008. *The Agile Movement.* [Online]   
Available at: http://agilemethodology.org/  
[Accessed 28 March 2017].

Beste, E. d., n.d. *Literature Review for Gamification in Education,* s.l.: s.n.

Carmo, L. C. d. S. C., Marcelino, M. J. M. & Mendes, A. J., 2007. *The Impact of Learning Styles in Introductory Programming Learning,* s.l.: ICEE.

Čisar, S. . M., Pinter, R., Čisar, P. & Radosav, D., 2013. *Teaching Computer Science in a Web-Based Environment,* s.l.: IEEE.

codecombat.com, n.d. *CodeCombat - Learn how to code by playing a game.* [Online]   
Available at: https://codecombat.com/

Corral, J. M. R. et al., 2013. *A game-based approach to the teaching of object-oriented programming languages,* s.l.: Science Direct.

Duffany, J. L., 2017. *Application of Active Learning Techniques to the Teaching of Introductory Programming,* Gurabo: IEEE.

Felder, R. M. & Brent, R., n.d. *Active Learning: An Introduction,* s.l.: s.n.

Felder, R. M., n.d. *Learning Styles and Strategies,* s.l.: s.n.

Felder, R. M. & Silverman, L. K., 1988. *Learning and Teaching Styles In Engineering Education,* s.l.: s.n.

Felder, R., n.d. *Index of Learning Styles (ILS).* [Online]   
Available at: http://www4.ncsu.edu/unity/lockers/users/f/felder/public/ILSpage.html

google.co.uk, n.d. *Google Forms - create and analyze surveys, for free..* [Online]   
Available at: https://www.google.co.uk/forms/about/  
[Accessed 24 March 2017].

Halleux, N. T. J. d., Xie, T. & Bishop, J., n.d. *Pex4Fun: A Web-Based Environment for Educational Gaming via Automated Test Generation,* s.l.: Semantic Scholar.

Jemmali, C. & Yang, Z., 2016. *May's Journey: A Serious game to teach middle and high school girls programming,* s.l.: Worcester Polytechnic Institute.

lua.org, n.d. *The Programming Language Lua.* [Online]   
Available at: https://www.lua.org/  
[Accessed 16 March 2017].

Maloney, J. et al., 2010. *The Scratch Programming Language and Environment,* s.l.: ACM.

moonsharp.org, n.d. *MoonSharp.* [Online]   
Available at: http://www.moonsharp.org/  
[Accessed 13 March 2017].

Moreno, J. & Robles, G., 2014. *Automatic Detection of Bad Programming Habits in Scratch: A Preliminary Study,* s.l.: IEEE.

Norwawi, N. M., Abdusalam, S. F., Hibadullah, C. F. & Shuaibu, B. M., 2009. *Classification of Students' Performance in Computer Programming Course According to Learning Style,* Malaysia: IEEE.

pexforfun.com, n.d. *Pex for fun - from Microsoft Research.* [Online]   
Available at: http://www.pexforfun.com/

Rubin, J. M., 2013. *The Effectiveness of Live-Coding to Teach Introductory Programming,* s.l.: ACM.

Salant, O. M., Armoni, M. & Ari, M. B., 2011. *Habits of Programming in Scratch,* s.l.: Semantic Scholar.

scratch.mit.edu, n.d. *Scratch - Imagine, Program, Share.* [Online]   
Available at: https://scratch.mit.edu/

Simons, K. D. & Klein, J. D., 2005. *The Impact of Scaffolding and Student Achievement Levels in a Problem-based Learning Environment,* s.l.: Springer.

Thomas, L., Ratcliffe, M., Woodbury, J. & Jarman, E., 2002. *Learning Styles and Performance in the Introductory Programming Sequence,* Aberystwyth: ACM New York.

Unity Technologies, 2017. *Unity - Game Engine.* [Online]   
Available at: https://unity3d.com/  
[Accessed 13 March 2017].

Yeh, C. H. & Yen, H. H., 2015. *Flow Experience Research of Sensing-Intuitive Dimension Learning Styles base on Visual Programming Language,* s.l.: IEEE.

Zualkernan, I. A., Allert, J. & Qadah, G. Z., 2006. *Learning Styles of Computer Programming Students: A Middle Eastern and American Comparison,* s.l.: IEEE.

# Appendices

## Appendix 1

Participation questionnaire URL: <https://goo.gl/forms/A9rrAXFJGeXJqCTI2>

*Q1. What style of learner are you?*

**(Multiple choice: Active; Sequential; Sensing; Visual / Verbal; Don’t know)**

*Q2. Did you manage to complete each level?*

**(Multiple choice: Yes; No)**

*Q3. If you answered no to the above question, please give a short reason as to why you did not finish.*

**(Short-answer text, to allow participant to comment on why they didn’t complete the game if they answer no to the previous question.)**

*Q4. How enjoyable did you find the game?*

**(A linear scale: 1 being “Not Very”, 5 being “Very”)**

*Q5. How helpful did you find the tutorial for each level?*

**(A linear scale: 1 being “Not Very”, 5 being “Very”)**

*Q6. How clearly do you feel the game explained what you needed to do? (Explanation of code etc.)*

**(A linear scale: 1 being “Not Very”, 5 being “Very”)**

*Q7. Do you think the game, or others like it, could provide better motivation to students learning how to program?*

**(Multiple choice: Yes; No)**

*Q8. Do you feel the game could be used as a teaching tool?*

**(Multiple choice: Yes; Yes (With more work); No)**

*Q9. Finally, do you have any additional comments about the game? (What could be improved, etc.)*

**(Short-answer text: An optional question to allow participants to make any additional comments on the game.)**

## Appendix 2

Test case results performed during development:

|  |  |  |  |
| --- | --- | --- | --- |
| **Test #** | **Test Description** | **Result** | **Comments** |
| **1** | Test basic Lua script execution with moon-sharp. | PASS |  |
| **2** | Test behaviour of loops in scripts. Expected behaviour is Unity will not lock up. | FAIL then PASS | The system for processing the Lua script with moon-sharp needed to put inside a coroutine, and auto yielded every 100 function calls to allow unity to catch up. This is moon-sharp provided functionality. |
| **3** | Test character C# movement API and Lua script integration | PASS |  |
| **4** | Test character collisions when moving with Lua script. | FAIL THEN PASS | Originally, move functions in C# could only check if the player could move to other tiles based on the position of the player at the time the function was called by the Lua script. This had to be changed to a virtual position system so the collision check system knew exactly which tile the player would be on when the move function was finally executed. |
| **5** | Test error output on Lua script errors prints error correctly to the UI error output box. | PASS |  |
| **6** | Test run script button doesn’t run multiple instances of the script, and resets any existing instances instead. | FAIL THEN PASS | Originally, the run script button would just run the Lua script over any other instances already running. The button now terminates any existing instances before running. |
| **7** | Test reset button correctly terminated the execution of the Lua script if running, and resets the player to the start of the level. | PASS |  |
| **8** | Test character can move function in Lua script. | PASS |  |
| **9** | Test player Lua global after level switch. | FAIL THEN PASS | The player Lua script global was not accessing the correct player object on level switches. This was because the old player reference in the Character API class was not being removed before the level changed. |
| **10** | Test level switching when player reaches target tile in each level. | PASS |  |
| **11** | Test max line count in scripts for levels that require them. | PASS |  |
| **12** | Test tutorial UI buttons for moving between the different pages. | PASS |  |
| **13** | Test UI button for re-opening the tutorial windows for each level correctly opens the tutorial UI and on the first page. | PASS |  |
| **14** | Test when Lua script finishes execution that player is reset to the start of the level if they did not reach the end tile. | PASS |  |
| **15** | Test questionnaire button on the end scene opens the Google Forms questionnaire when pressed. | PASS |  |

## Appendix 3

Level 3 of the game; The level where every participant struggled to understand the functionality of ‘while’ loops.

