UNIVERSITY OF BATH

DEVELOPMENT OF AN EDUCATIONAL GAME TO TEACH CONDITIONAL AND ITERATIVE CONTROL STRUCTURES

Amarnath Kakkar

A final year project submitted in partial fulfilment for the degree of Bachelor's in Computer Science and Mathematics with Honours

University of Bath

May, 2019

This dissertation may be made available for consultation within the University Library and may be photocopied or lent to other libraries for the purposes of consultation.

Signed:

Development of an Educational Game to Teach Conditional and Iterative Control Structures

Submitted by: Amarnath Kakkar

COPYRIGHT

Declaration

This dissertation is submitted to the University of Bath in accordance with the requirements of the degree of Bachelor of Science in the Department of Computer Science. No portion of the work in this dissertation has been submitted in sup- port of an application for any other degree or qualification of this or any other university or institution of learning. Except where specifically acknowledged, it is the work of the author.

Department of Computer Science
University of Bath

Supervisor: Dr. Alan Hayes
May, 2019

Abstract

This project describes the development of an educational game, Programmer's Dungeon, to teach iterative and conditional control structures. Research is conducted on the benefits and effectiveness of educational games. Research of the impacts of games on learning, engagement and motivation are assessed. Finally educational game design is assessed, with focus on learning techniques and game elements.

The research helps inform the functional and non-functional requirements of the game. The design and implementation expand and discuss the non-trivial components of the game, while highlighting any issues faced.

The game is then tested in three stages, and results were gathered. The first two stages of testing; system testing and initial playthroughs, helped identify and fix issues within the game. The final stage of testing user testing. This involved designing and carrying out an experiment which involved 21 participants. The experiment had a pretest-posttest design, and aimed to identity to which extent the non-functional requirements were fulfilled. Participant learning, engagement and motivation was also assessed.

Results from the user testing showed that participant knowledge of iterative and conditional control structures significantly increased. Participants also described the game as fun and enjoyable. Participant interest in iterative and conditional control structures also significantly increased after playing the game. Participants after the game were motivated to learn more complex iterative and conditional control structures.

The game developed in this project can be played online at the following URL: http://tutorial-env.4yzpcwhdm6.us-west-2.elasticbeanstalk.com/

Table of Contents

List of Figures

List of Tables

Outline of Project

This project describes the development of an educational game to teach iterative and conditional control structures.

First, the project description will be highlighted. This will identify the need to conduct the research and its relevance to educational games. This will include the project aims and objectives.

Chapter 2 will provide an overview of educational games, teaching iterative and conditional constructs and educational game design. Current research and state of educational games will be explained. A description of the definition of an educational game. The benefits of educational games will then be identified, regarding the learning, engagement and motivation. Some examples of educational games and their applications will be discussed. Method of teaching iterative and conditional statements will be identified, along with any issues. Finally those aspects of games that foster learning will be discussed.

Chapter 3 will identify the functional and non-functional requirements of the game. These requirements will reflect the areas of significance that were discussed in chapter 2.

Chapter 4 describes the design. The core idea, platform for the game, game mechanics, level design and user interface will be covered.

Chapter 5 will discuss the implementation. The implementation approaches and choices, non-trivial components of the system will be highlighted and explained.

Chapter 6 describes the proposed testing of the game. The system testing, initial feedback and user testing will all be explained.

Chapter 7 highlights the results from system testing, initial feedback and user testing. Any improvements and changes to the game will be discussed. Results from user testing will describe which non-functional requirements were fulfilled.

Chapter 8 will give an overview of the entire project. Conclude any results.

Chapter 9 summarises the conclusion from all previous chapters, to provide a walkway to future research.

Acknowldgements

I would like to acknowledge the valuable assistance of the following individuals, as without their continued support and assistance this work would not have been possible:

First and foremost, I would like to start by thanking my supervisor and assessor Dr. Alan Hayes, who provided motivation throughout this project. Without his advice and guidance this work would not have been possible.

I have faced many challenges in order to complete this project; the advice, guidance and input from friends, especially Dan Savage at the university of Bath made this project possible.

Last but by no means least, I would like to thank my parents, Baba Kakkar and Vinay Kakkar who have provided support throughout this project.

Chapter 1

Introduction

Educational games have been an area of increasing interest in both the academic and educational fields. The benefits of educational games is now widely recognized and has been thoroughly argued.

There are a number of advantages of using educational games as educational tools. Firstly, more people spend time playing games now than have ever done before. Almost a third of the human population regularly play games. However despite how familiar games may be to people, they are also great learning tools. They help promote a number of various skills and they are naturally engaging.

Educational games can also be played virtually anywhere given an internet connection. This differs from the more traditional methods of learning, where learners have to attend a specific location at a specific time to gain learning material. In addition, student motivation in computer science courses at higher education is below ideal. Students argue when starting out on such a course, they struggle in learning to program, and this is acknowledged amongst researches.

Many studies have looked into the effects of educational games on learning, engagement and motivation. Overall the results are positive, however determining how effective educational games are when compared to other forms of teaching is inconclusive. There is a lack of studies to deliver a full picture.

Chapter 2

Literature Review

2.1 Video Games

The video games industry has grown very rapidly in recent years, and is expected to continue to grow. The current market value is predicted as $150 billion USD and is predicted to reach $180 billion by 2022 (Newzoo, 2018). In 2006, video games were considered as one of the most popular forms of entertainment in the United States (Sherry et al., 2006; Ritterfeld and Weber, 2006). Currently, video games are considered a popular form of entertainment globally.

A video game can be defined as "a mental contest, played with a computer according to certain rules for amusement, recreation, or winning a stake" (Zyda, 2005).

2.1.1 Impacts

Initially, the majority of research on the effects of playing video games focused on the negative impacts, such as the potential aggression, addiction and depression from 'gaming'. However, researchers have argued that a more balanced perspective is needed (Granic, Lobel and Engels, 2014). Playing video games has been linked to an increase in perceptual, cognitive, behavioural, affective and motivational abilities (Connolly et al., 2012). Studies have also demonstrated that there are positive impacts from playing violent games, such as increased visuospatial skills (Ferguson, 2007).

2.1.2 Uses

Video games are now used for a wide variety of reasons. They are becoming increasingly useful in the global education and training market. Apart from entertainment, video games are being used in: military, government, education, corporate and healthcare (Susi, Johannesson and Backlund, 2007).

2.2 Educational Games

The term 'Serious Game' can be used to describe an educational game. A serious game refers to a game that has an educational purpose and is not intended to be played primarily for entertainment (Abt, 1970).

Serious games became an established academic field of study in 2007, after establishment of The Serious Games Institute (Wilkinson, 2016). The market value of the serious games industry in 2016 was predicted at $1.5 billion USD, and is predicted to reach $9 billion in 2023 (AlliedMarketResearch, 2017).

2.2.1 Definition

There is currently no singleton definition for term 'Serious Game'. Susi, Johannesson and Backlund argue that, groups and individuals define the term depending on their perspectives and interests, and that there are a wide variety of groups and individuals focusing on different issues (Susi, Johannesson and Backlund, 2007). The first recorded definition of the term was set out in 1970 by Abt (Wilkinson, 2016), who defined it as follows: "Games that have an explicit and carefully thought-out educational purpose, and are not intended to be played primarily for amusement. This does not mean that serious games are not, or should not be, entertaining." (Abt, 1970). In 2005, Michael and Chen built on this definition to come up with the definition that they are "Games that do not have entertainment, enjoyment or fun as their primary objective" (Michael and Chen, 2005). This definition suggested that serious games are not limited to only educational purposes. The commonly agreed upon definition closely matches the definition by Michael and Chen (see Susi, Johannesson and Backlund, 2007).

Another definition that is often referred to was set out by Zyda (see Susi, Johannesson and Backlund, 2007). This definition however contradicts the one set out by Michael and Chen (Susi, Johannesson and Backlund, 2007). In contrast, Michael and Chen argue that serious games should not, have entertainment or fun as their primary objective. On the other hand, Zyda expressed that the entertainment component of the game should come first, and also that the story of the game is more important than the pedagogy (Zyda, 2005). His definition is as follows: "a mental contest, played with a computer in accordance with specific rules, that uses entertainment to further government or corporate training, education, health, public policy, and strategic communication objectives" (Zyda, 2005). However this definition also suggests that serious games can only be digital (Djaouti et al., 2011).

Since there are a variety of definitions, for this dissertation, I will use the one provided by Abt and work entertainment around the primary purpose of the game - to teach.

2.2.2 History

Educational games have arguably existed since the 7th century. Among the oldest is the board game 'Chaturgana', which is argued by historians to be the precursor to chess (Wilkinson, 2016). The aim of the game was to teach officers to become better planners for battles (Wilkinson, 2016). Another board game created more recently - in the 20th century was 'Landlord's Game'; a precursor to monopoly. It was designed to illustrate the dangers of capitalist approaches to land taxes and property renting (Wilkinson, 2016). So we can see that, games designed to educate, have existed for a long time.

The interest in digital educational games, has been observed since 1967, when the first educational program was developed, 'Logo Programming' (Hayes and Games, 2008) . Logo programming was an environment that allowed players to utilise the programming language LOGO in order to learn mathematics (Feurzeig et al., 1969). Logo was popular among schools in the US (Lehrer, 1986), and it became a key part of educational strategies research (Hayes and Games, 2008).

Academic interest in games that could be used for purposes apart from entertainment emerged emerged in 1970 by Abt, in his book *Serious Games* (Breuer and Bente, 2010). The rise in digital games around this time created an opportunity for developing serious games (Wilkinson, 2016).

However, serious games did not gain much traction until 2002, when a game developed by the US Army, became hugely popular. 'America's Army' was developed as a training and recruitment game for the military. It is now considered as the forefront of modern serious games (Zyda, 2005; Wilkinson, 2016). In conclusion the potential of using games as educational tools has been demonstrated for a long time, and is now being further researched and developed, focusing on training and education in a number of different industries (Wilkinson, 2016).

2.2.3 Benefits

Serious games have become an interesting area for multidisciplinary academic research (Breuer and Bente, 2010). There are interests from fields such as psychology, computer science, pedagogy, sociology and cultural studies (Breuer and Bente, 2010). Many studies have looked into and discussed the benefits of serious games in educational contexts, and I will discuss some of these below.

2.2.3.1 E-learning

E-learning can be defined as an approach to teaching and learning, based on the use of electronic media and devices (Sangrà, Vlachopoulos and Cabrera, 2012). Thus, digital educational games can be seen as a type of e-learning.

Educational games have inherent beneficial properties. For instance, they are able to provide information on demand and just in time, and in the context of actual use and people's purposes and goals, something that does not often happen in schools (Gee, 2003).

Other properties of e-learning include: ease of accessibility; can be used in absence of teachers or instructors; provide opportunities for relations between learners, helping eliminate the potential of hindering participation; low cost per person served; allows self-pacing - allowing student to study at their own pace; high level of interactivity; ability to use attractive graphics, and is an engaging and entertaining activity (Arkorful and Abaidoo, 2015; Girard, Ecalle and Magnan, 2013).

In a study carried out on what university students thought about e-learning, students reported that they expected e-learning to be an integral part of the learning process within higher education (Connolly et al., 2012). Thus, the use of a digital game in higher education, may not be an unfounded concept to students.

2.2.3.2   Learning through Games

Games can be a great learning environment (see Prensky, 2003; Gee, 2003). Despite the vast research on the negative impacts of gaming, playing video games identified for imparting beneficial skills such as, visual attention, spatial skills, problem solving skills and creativity (Granic, Lobel and Engels, 2014).

When players first begin a new game, they first learn the rules and the controls of the game, then use this newly acquired knowledge to complete objectives or levels. As players progress through the game, it becomes more difficult and calls for them to use better skills and knowledge for them to move to the next stages of the game. Vygotsky coined the term *the zone of proximal development*, where students gain a higher degree of learning when they are presented with tasks which are just beyond their current level of ability but may require some help to complete (Vygotsky, 1978).

Today's learners have grown up immersed in digital technology (Prensky, 2001); Prensky calls this generation of people 'Digitial Natives', because they have spent long periods of time playing video games (Prensky, 2003). In 2016, 2.5 billion people were reported to actively play video games worldwide (AAStocks, 2016), and a year later, roughly 57% were aged between 10 and 35 (Newzoo, 2017). Therefore, this generation will be more receptive to computer-based learning (Girard, Ecalle and Magnan, 2013).

Playing games is naturally a fun and pleasurable activity (Prensky, 2001), and research has shown that fun is important for the learning process, as learners can be more motivated and willing to learn (Bisson and Luckner, 1996; Cordova and Lepper, 1996). Learners' enjoyment of a game has also been shown to improve learners' acquisition of knowledge (Giannakos, 2013).

2.2.3.3 Effectiveness of Learning through Games

Evidence of learning through educational games has been demonstrated (Connolly et al., 2012; Wouters et al., 2013; Girard, Ecalle and Magnan, 2013). However, the effectiveness of educational games is undetermined (Connolly et al., 2012; Girard, Ecalle and Magnan, 2013). There are also different viewpoints when it comes to determining their effectiveness. Two such meta-analyses, Wouters et al. (2013) and Girard, Ecalle and Magnan (2013), explored at the effectiveness of learning through games, and made different conclusions. These studies are compared in Table 2.2.1.

Table 2.2.1: Comparison of meta-analyses on the effectiveness of learning through games

	Wouters et al. (2013)	Girard, Ecalle and Magnan (2013)
Types of Games	Serious games	Serious games & video games
Measuring	Learning, retention and motivation	Learning and engagement
Learning Outcomes	Knowledge or skill acquisition	Knowledge or skill acquisition
Instructional Domain	Biology, maths, language or engineering	Various (including: academic knowledge, cognitive skills, professional knowledge, cancer therapies)
Studies Published Between	1990 - 2012	2007 - 2011
Experimental Design of Studies	Posttest or pretest-posttest	Atleast pretest-posttest
Studies Reviewed	39	9
Age Range of Studied Population	Wide range	9 - 47
Results support effective learning through games	Yes	Inconclusive

Wouters et al. found that serious games were more effective in terms of learning and retention when compared to conventional teaching methods, but not more motivating. On the other hand, Girard, Ecalle and Magnan found that only a few of the games resulted in improved learning, with the others having no difference when compared to traditional methods of teaching. A point to note is that in Wouters et al. (2013), serious games were compared to lectures, reading, drill and practice, or hypertext learning environments, whilst in Girard, Ecalle and Magnan (2013),

they were compared to face-to-face lessons, pencil-and-paper studying or no studying at all. The former is a more modern way of teaching, whereas the latter is very limited.

Wouters et al. evaluated a broad spectrum of studies, whereas Girard, Ecalle and Magnan only evaluated randomised control trial studies. Wouters et al. argued that if they only considered the studies with randomised samples with a pretest-posttest design, similar to Girard, Ecalle and Magnan study, the positive effects in favour of serious games may disappear.

In conclusion, there is a need for more empirical research to determine the effectiveness of serious games, and this is being addressed (Connolly et al., 2012).

2.2.4 Examples

There are many existing educational games designed to teach various topics. I will cover some educational games and interesting technologies that help further programming skills.

Scratch

Scratch is a block-based visual programming language and an online community for sharing, discussing and 'remixing' one another's projects. Projects can range from creating your own interactive stories, games, animations or simulations, which users can share with one another online (Resnick et al., 2009). Today, Scratch has accumulated almost 40 million users, with the core audience between ages 8 and 16, which have created and shared more than 40 million projects (Scratch, 2019).

The Scratch language is based on a collection of programming blocks that can be snapped together to create programs. Scratch blocks are shaped in a way such that they can only fit with other blocks to make syntactic sense (see Figure 2.2.1). The creator of Scratch and fellow researchers argued that the social aspect of Scratch was important for it to succeed, with a user claiming that she learnt a lot about different kinds of programming by looking at, downloading, and modifying the scripts from other peoples games. Users can also work together on projects, helping them develop their collaborative and team leadership skills (Resnick et al., 2009).
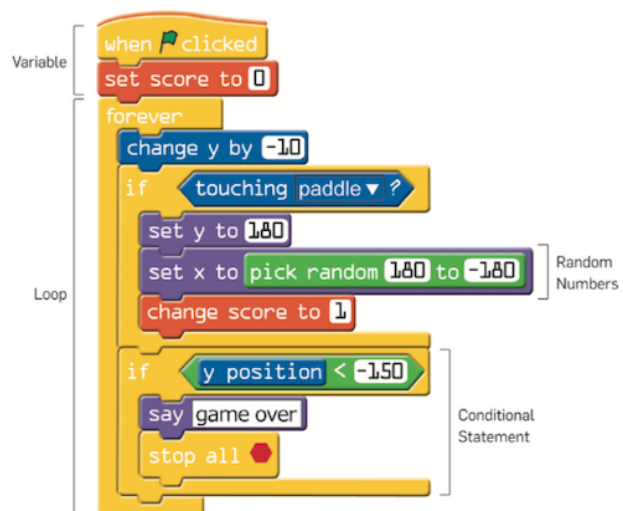
Figure 2.2.1: An example Scratch script (Resnick et al., 2009)

Scratch has also been used in higher educational contexts (Resnick et al., 2009). In a study conducted by Malan and Leitner (2007), Scratch was used at Harvard's summer school to introduce students to programming. After a total of five hours on Scratch, students transitioned onto Java for the remainder of the course. Malan and Leitner found that more than 75% of students felt that using Scratch, was a positive influence with their transition onto Java; a more complex syntactical language (Malan and Leitner, 2007). It is clear from the student responses that Scratch helped develop their computational thinking skills. This is often hard to do in languages like Java as students can feel overwhelmed trying to learn the complex syntax, which can hinder their ability to learn to program (Koulouri, Lauria and Macredie, 2014).

Prog&Play

Prog&Play was developed to tackle the issue of student motivation in computer science courses in higher education. Prog&Play is a 3D real-time strategy game designed to strengthen programming skills. It was built on top of an existing open-source multiplayer game, because of the potential advantage of the game already being robust. The developers of Prog&Play created an API that enabled students to interact with the game through programmable commands, as the original game did not contain any programming (Muratet et al., 2011).
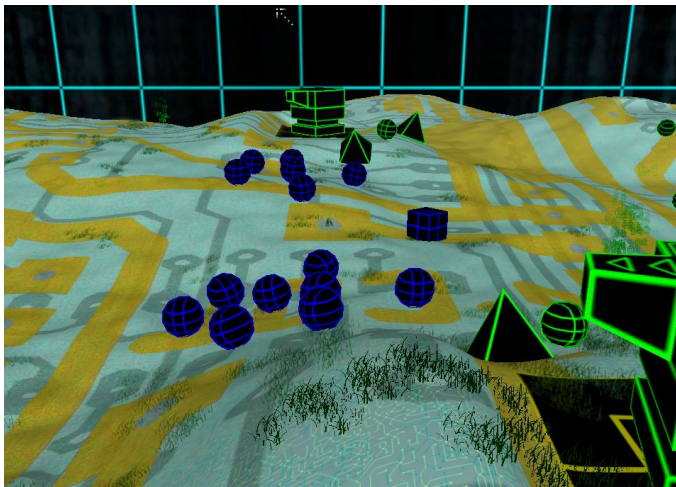
Figure 2.2.2: A screenshot of the game Prog&Play (Muratet, 2012)

Prog&Play introduced variables, functions and conditional and iterative control structures. The game is based on a war between the factions, 'Systems', 'Hackers' and 'Networks' (see Figure 2.2.2). Players choose a side and give orders to their units through a set of commands. The developers also created a single player mode, where students could be gradually introduced to learning topics and learn how to play. The multiplayer aspect then allowed students to create their own programs and compete with each other (Muratet et al., 2011).

Through examinations conducted by the teachers, Muratet et al. found that students who played the game achieved better marks than those who did not. Many students who played the game went on to choose a computer science unit the next semester, compared to those that did not play the game at all. Interestingly, a majority of students preferred the programming version, Prog&Play, to the original game which did not require any programming (Muratet et al., 2011).

Gidget

Gidget is a web-based game developed as part of a study to improve learning in novice programmers. The study tested the effects of personifying programming tool feedback. One such personification was to programming errors that were displayed to users (Lee and Ko, 2011).

In the case of Gidget, you have have to work with a damaged robot, also named Gidget, to help clean up a city and protect its animals after a chemical spill. The robot, Gidget, acts as a companion and feedback tool. It uses personified language, takes the blame for syntax and runtime errors, and has an emotional face (Lee and Ko, 2011). Lee and Ko claimed that this would change the role of the conventional feedback tool often perceived as an authoritative figure, which can be off-putting, to a collaborator needing assistance. The game teaches the design and analysis of basic algorithms in a language designed specifically for the game. Players have to debug the code produced by Gidget to complete the levels (Lee and Ko, 2011).

Lee and Ko carried out a study with 116 participants, using two versions of the game. One version was with a personified Gidget robot (see Figure 2.2.3), and the other with a Gidget that had a faceless screen that provided impersonal feedback. The results revealed that the group using the personified Gidget, completed more levels in roughly the same amount of time in comparison to the group using the faceless Gidget, suggesting that the personification of feedback had a positive effect on participations' motivation to play. Although, the personification of Gidget did not affect the enjoyment felt playing the game (Lee and Ko, 2011).
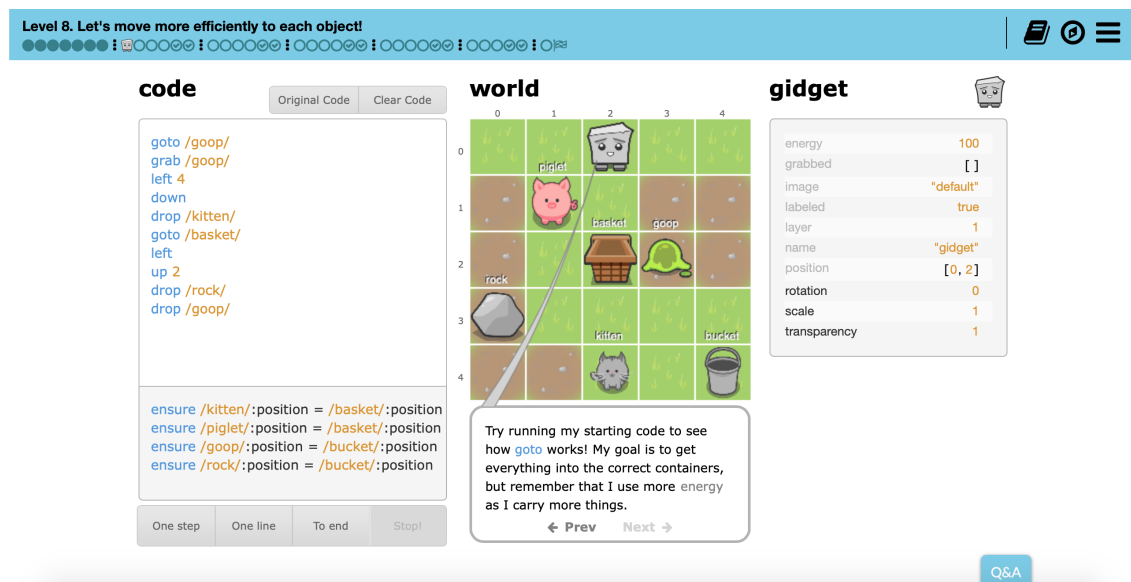
Figure 2.2.3: A screenshot of the game Gidget (Gidget, 2019)

2.3   Teaching Conditional and Iterative Control Structures

In programming language design and implementation, 'control structure' refers to those constructs which determine the sequence of execution, of operations and statements within programs (Riccardi, 1981). This definition is still relevant today (see Wikiversity, 2019). There are three types of control structures; sequential, conditional, and iterative (Leppänen, 2007). A sequential structure is execution of a code line by line. A conditional construct is used to make decisions, for example, *if* and *if else* statements in JavaScript. An iterative construct is used for looping, for example, *for* and *while* loops in JavaScript.

These control structures are likely to be taught in larger introductory units, which cover a wide range of fundamental concepts (ACM-IEEE, 2013). For example, at the University of Bath, conditional and iterative structures are taught in semester 1 of year 1 of a computer science undergraduate degree (UniversityOfBath, 2018).

2.3.1   Methods

At university, introductory programming units are conventionally structured courses based on lectures and practical laboratory work. Students learn about programming concepts through the programming language being taught (Robins, Rountree and Rountree, 2003; ACM-IEEE, 2013). Robins, Rountree and Rountree suggest that this approach is popular due to the importance of the programming knowledge gained from the programming language, and the sheer volume and detail of language related features that can be covered.

Another method for learning iteration and conditional constructs, which is becoming increasingly popular is online resources. These resources include tutorial websites, such as Codeacademy and Khan Academy, which have accumulated millions of users, block-based programming environments such as Scratch and Alice, which provide creative visual environments, and educational games (Lee and Ko, 2015).

Both traditional forms of learning and e-learning, guide the student through the learning experience and often set the students practical work to complete. However there are a few differences between these methods. In contrast, traditional methods of teaching centre on instructors who have control over class content and the learning process, whereas online resources, provide a learner-centred, self-paced learning environment (Zhang et al., 2004). Zhang et al. (2004) goes are as far to say that e-learning could replace traditional classroom learning. Nevertheless, using e-learning to complement the learning process can make learning more effective and improve the learning experience (Zhang et al., 2004; Concannon, Flynn and Campbell, 2005)

2.3.2 Issues

It is generally accepted that learning to program is a difficult task, and there are several problems associated with this (Koulouri, Lauria and Macredie, 2014). For leaners, the overhead of learning the syntax and semantics of a language at the same time, difficulties in combining new and previous knowledge, and developing their problem-solving skills, all add to the complexity of learning how to program (Koulouri, Lauria and Macredie, 2014).

There is also concern over the high drop out and failure rates of first and second year University students in computer science courses. This is potentially associated with considerable failure rates in introductory programming units (Koulouri, Lauria and Macredie, 2014). However, researchers have argued that there is little to no high-quality empirical data to support these claims (Bennedsen and Caspersen, 2007; Watson and Li, 2014). In two different studies; Bennedsen and Caspersen (2007), and Watson and Li (2014), both found that the failure rate of introductory units were 33% and 33.3%, from an average of 63 and 51 institutions respectively. Both studies argued that these rates were not "alarmingly high", but there is considerable potential for improvement (Watson and Li, 2014).

Koulouri, Lauria and Macredie (2014) carried out 4-year study and found that certain implementations of introductory units, could improve student learning performance. One finding was that programming proficiency of novice programmers is dependent on the teaching approach of an introductory programming unit. Introductory programming units are also considered difficult (Watson and Li, 2014). In conclusion, choosing an approach that is engaging and motivating to teach fundamental programming concepts, is important.

2.4   Educational Game Design

It is not sufficient to just assume that all forms of games are equally suitable for learning, and that simply presenting material in a game-like setting, will increase the quantity and quality of learning (Breuer and Bente, 2010). Designing educational games requires a focus that is different from general game design; otherwise, we may end up designing fun games with little or no learning value (Barnes et al., 2007).

Breuer and Bente (2010) argue the ideal educational game combines entertainment and learning in a way that the players/learners do not experience the learning part as something external to the game. This idea of stealth learning should inform any approach to designing, using and evaluating educational games (Breuer and Bente, 2010). However, combining compelling and interactive design elements with specific curricular content that aims to retain learner's interest and attention, is a difficult task (Prensky, 2003).

2.4.1 Games and Learning

Researchers have looked into the many factors that influence learning (Roungas and Dalpiaz, 2015). Below are some of these factors that influence learning in a positive way, and how these are implemented in games. Roungas and Dalpiaz (2015) argue that these following factors are important and should be considered when designing serious games.

2.4.1.1 Motivation

Motivation is a key factor that drives learning. Without motivation, learning stops (Gee, 2003). A key factor of motivation is interest (Rousseau, 1762) (as cited in Roungas and Dalpiaz (2015)). Learning is effective when learners are interested in the material being taught and how it is presented.

Games are natural sources of motivation. Games that bring about an optimal balance between challenge and frustration for players, is a very motivating state to be in (Gee, 2003). In his theory of flow, Csikszentmihaly describes that some leaners can become so focused in and absorbed by an activity that they lose track of time and completely ignore other tasks (Csikszentmihalyi, Abuhamdeh and Nakamura, 2014).

Researchers have looked into the intrinsic motivational properties of video games, that is, player motivation that arise due to factors within the players themselves. For example, pleasure and desire to develop a skill are forms of intrinsic motivations (Roungas and Dalpiaz, 2015). Malone and Lepper (1987) investigated these properties of games, and presented a set of aspects of games that would make game-based learning more fun and engaging. The work by Malone and Lepper is often referred to, but Whitton (2011) noted that Malone's work is not as valid today or as valid when considering adult engagement. With this, Whitton described five aspects that lead to engagement specifically in games targeted for higher education: challenge, control, immersion, interest and purpose.

2.4.1.2 Repetition

Repetition is considered crucial for learning, but some modern researchers do not consider it very important. However, it is deemed useful for learning skills such as programming (Roungas and Dalpiaz, 2015), where it could be used for learning programming syntax and function names.

Repetition is very common in video games. It allows players to learn, memorise and potentially master some rules or controls of the game. Once players are familiar with these skills, the game then introduces a new problem, which requires players to integrate their old skills with new ones. These new problems are practised until a problem requiring a different solution comes along. This cycle is repeated throughout the game (Gee, 2003).

However, repetition needs to be managed, as it could lead to boredom. To avoid this, Roungas and Dalpiaz (2015) suggest modifying the content of the task after each repetition. This can be achieved in games by slowly introducing new rules or controls that build upon previous game knowledge, or, by framing the task in different scenarios. Roungas and Dalpiaz (2015) also suggested managing how frequent the task is presented. A task repeated too often could lead to boredom, or a task repeated at very long time intervals, could reduce the learning benefits from repetition.

2.4.1.3 Rewards

The use of rewards and punishments in learning has various impacts. Firstly, rewards allow a learner to link learning with something pleasant. Secondly, rewards are form of extrinsic motivation, that is, learners are motivated by factors external to themselves. For example, achieving high grades, winning trophies or recognition by peers are all forms of extrinsic motivation (Roungas and Dalpiaz, 2015). Thus attempting to seek such rewards may give learners motivation to continue their engagement with the activity. Finally, rewards were linked to increased learning performance in a study that used rewards in an educational game (Filsecker and Hickey, 2014).

Wang and Sun (2011) goes as far as to say, rewards can foster intrinsic motivations, specifically, fun that arises from achieving rewards. An increase in intrinsic motivation in learners, has shown that learners are more deeply involved in activities and are more confident (Cordova and Lepper, 1996).

There are a variety of ways rewards can be expressed in video games, the following are some examples: score system, experience point system, item granting, resources, achievements, instant feedback, plot animations and unlocking mechanisms (Wang and Sun, 2011). However, the relevancy and frequency of rewards needs to be managed, otherwise they can lead to diminishing returns; rewards should be suitable to the target audience, and like repetition, the frequency of rewarding is important. Rewarding too often can cause players to lose interest, or rewarding very little can lead players to lose motivation to play (Roungas and Dalpiaz, 2015).

2.4.2 Game Elements

Games have several characteristics. Roungas and Dalpiaz (2015) argue the following characteristics are important for serious game design. These characteristics are aimed at keeping games engaging, exciting and educational.

1. **Rules**. Rules are essential to a game. They describe how the game works and limit the players' actions (Roungas and Dalpiaz, 2015). Whilst surveying players of introductory programming games, Barnes et al. (2007) gathered that rules should be provided and accessible throughout the game. This would allow players to refer back to the rules during the game

in case they needed reminding or help.

2. **Goals**. Goals allow players to judge their performance. They also act as a motivation for players (Roungas and Dalpiaz, 2015). (Barnes et al., 2007) found that goals should also be provided and accessible throughout the game. They must be clearly tied to in-game feedback, such as avatar health or score, that either motivates or penalises players (Barnes et al., 2007).

3. **Challenge**. Challenge leads to engagement, if, tasks are optimally balanced between challenge and skill (Whitton, 2011; Csikszentmihalyi, Abuhamdeh and Nakamura, 2014). Challenges need to be carefully designed in order to not be too boring or too difficult (Roungas and Dalpiaz, 2015).

4. **Feedback**. Feedback allows players to assess their performance (Roungas and Dalpiaz, 2015). It leads players towards correct answers if their assumptions are wrong, and thus creates learning opportunities. However in programming, feedback in the form of errors can be quite discouraging to novice programmers (Lee and Ko, 2011). Thus care should be taken when designing feedback in a beginner programming game. Feedback should be presented clearly and on time (Roungas and Dalpiaz, 2015).

Chapter 3

Requirements Specification

The literature review highlights several key areas of study that should be taken into consideration when developing an educational game. The following aims and objectives were determined:

1. Design game that takes into account the areas highlighted in the educational game design part of the literature review.

2. Implement the game as per the problem set out in the design and requirements, aiming to create a fun and engaging game.

3. Carry out user testing to determine the effects of game on user learning, engagement and motivation.

The aim of the current study is to develop an educational game to teach iterative and conditional control structures. This will be achieved through the following requirements. The requirements are split into functional and non-functional sections, and each requirement is given an importance rating of: high, medium or low, which refer to being essential, conditional or optional for the system respectively.

3.1 Non-Functional Requirements

3.1.1 **Game must improve novice programmers' knowledge on iterative and conditional control structures.** This is the main aim of the study. A key objective of an educational game is to educate the user.
Priority: High

3.1.2 **Game must not assume player has any existing programming knowledge.** The game should be a useful learning tool for anyone wanting to learn about iterative and conditional control structures, regardless of prior knowledge.
Priority: High

3.1.3 **Game must be easily accessible from home, work or school.** A key benefit of a digital educational game, is the ability to access it and learn from it anytime or anywhere.
Priority: High

3.1.4 **Game should not require complex installation.** The game should be easily and readily accessible, and be quick to start playing.
Priority: Medium

3.1.5 **Game should be fun to play.** Fun is a key aspect of an educational game as it can helps motivate players (Cordova and Lepper, 1996) and improve their learning performance (Giannakos, 2013).
Priority: Medium

3.1.6 **Player should be engaged throughout the game.** Engagement is key in learning.
Priority: Medium

3.1.7 **Player should feel challenged.** Challenge is a key aspect that leads to player engagement (Whitton, 2011).
Priority: Medium

3.1.8 **Game should have a reasonable difficulty curve.** Players should be able to get a reasonable understanding of a concept before being introduced to new ones. The game should also have a balance between challenge and skill.
Priority: Medium

3.1.9 **Game may foster player motivation to learn more complex programming.** Student motivation in introductory programming courses is very low and can be improved (Koulouri, Lauria and Macredie, 2014).
Priority: Low

3.2   Functional Requirements

3.2.1 **Game must at least introduce *if* statements and *for* loops.** These constructs can be considered essential iterative and conditional control structures, and so the game must cover these.
Priority: High

3.2.2 **Game must have levels with their own clear defined mini goals.** Clear defined goals will tell the player what to do each level. This can help to introduce iterative and conditional constructs.
Priority: High

3.2.3 **Game must give players some control over choice of action.** Control over choice of action is a key aspect that leads to engagement (Whitton, 2011).

Priority: High

3.2.4 **Game must give feedback to player solutions.** Feedback is important as it allows players to assess their performance and deal with any mistakes. Thus creating learning opportunities.
Priority: High

3.2.5 **Game must use some form of in-game rewards and punishments, such as score and hit points.** In-game rewards and punishment foster player engagement (Roungas and Dalpiaz, 2015).
Priority: High

3.2.6 **Game must have a help section.** Whenever players are stuck, they should be obtain help from the help section.
Priority: High

3.2.7 **Game rules must be accessible throughout the game.** Players must be able to refer back to the rules throughout the game.
Priority: High

3.2.8 **Game should give feedback in a personal and constructive way.** Personal and constructive feedback may improve player motivation (Lee and Ko, 2011).
Priority: Medium

3.2.9 **Game should have a tutorial to introduce the game.** A tutorial will be able introduce the game in an engaging and interactive way.
Priority: Medium

3.2.10 **Game should be web-based, and playable in three most popular web browsers.** These are Chrome, Safari and Firefox (StatCounter, 2019). Allows the game to be easily and widely accessible.
Priority: Medium

3.2.11 **Game may provide tips to the player about individual level.** In case player is ever stuck on a level and the help section does resolve the player issues, tips should help the player.
Priority: Low

3.2.12 **Game may save players progress.** Saving game state would allow players to exit the game for any reason, and return any time they wish.
Priority: Low

Chapter 4

Design

Since the start of the project, the idea for the genre of the game was a 2D maze adventure. Researchers have argued which genres of games are best suited for educational purposes (Moreno-Ger et al., 2008), however most of the discussion on educational game design does not revolve around the game genre, but instead on which game elements, pedagogical concepts and player feelings aroused by the game are most important. These aspects were discussed in the literature review and were used to inform and influence the requirements specification. This chapter aims to indicate how the problem, as specified within the requirements, is analysed to create a potential solution.

4.1 Core Idea

The core idea of the game is a character stranded in a dungeon, and the user must write pieces of code to control the character and help him escape. To do this, the user is given a set of pre-defined functions to use to control their character. These pre-defined functions will simply be referred to as 'functions' throughout this study. Throughout the game, users are slowly introduced to *if* statements and *for* loops, which they then must use with the functions to create pieces of code. The aim is for users to initially create simple pieces of code, then move onto creating more complex structured and efficient code as they progress.

To help discuss the design, the goals of the game are split into two. The first goal, and also the aim of this study, is to educate users on certain iterative and conditional control structures. These will be referred to as the educational goals. The second goal is to help the in-game character escape. This will be referred to as the game goal.

4.2 Web-based

A web-based design was adopted to develop the game. A web game has several advantages over other forms of video games. Namely, these are:

- Accessible on most web browsers without having to create different versions of the game for each browser. This is cost-effective and reduces hassle for users.

- Accessible through a number of different web-based devices, without having to create different versions of the game or install the game onto each different device. This allows the game to be accessible in more places.

- Ease of accessibility, given that computers with an internet connection are ubiquitous. Only requires a browser to play.

- Requires no additional download, installation or update to plug-ins. Reduces the complexity needed for users to access the game.

- Easy to update web applications once they are deployed. Requires no update from user to get the latest version of the game, apart from perhaps updating their web browser to the latest version.

- Can achieve greater levels of interoperability compared to desktop environments (Roungas and Dalpiaz, 2015).

4.3 Game Mechanics

Game mechanics can be defined as the methods invoked by agents for interacting with the game world (Sicart, 2008). In the context of this game, the agents are the players and the methods are those elements of the game, that the user interacts with either directly or through their actions, that effect the game state. The following subsections will discuss these elements in more detail and the justification for their adoption.

4.3.1 Functions

Commonly when using *if* and *for* constructs, statements are defined within the construct to be executed. For this game, functions that help the in-game character, were decided to be used as such statements. Furthermore, the use of functions to control in-game characters had a positive impact on player engagement in the educational game Prog&Play.

Since the idea was for the character to be stranded in a dungeon. One essential function would be to get the character to move. However, just using one function throughout the game may get a bit boring. Thus in attempt to add to the dungeon feel, another function was designed to shoot. The

dungeon would be filled with enemies, which the character would have to shoot to get through to the exit. Walk and shoot would be all the functions the player could use to directly interact with their character.

Two more functions were designed to be used as conditions within conditional constructs. Specifically these are, functions that would return true or false based upon facts about the game state: if a wall is nearby, or, if an enemy is nearby. Using these in conjunction with the walk and shoot commands, would give users flexibility to come up with their own unique solutions to a level. Thus essentially giving users some control. Learning to program occurs efficiently, when the learner is able to use trial and error, come up with their own solutions and learn from their mistakes.

Throughout the game players would have to use these functions with *if* statements and *for* loops to help achieve the game goal, in turn helping them achieve the educational goals. Thus these functions can be seen as the bridge between the entertainment aspect of the game and the educational aims.

However as discussed in the literature review, care needs to be taken when trying to introduce programming to novice programmers. They can often struggle with the complexity of the syntax and semantics of the programming language. To tackle this issue, the aim is to dissipate the introduction of these functions and constructs throughout the game, which will be discussed in more detail in the level design section. This is also why only four functions were designed, which all take one of four parameters: left, right, up or down. This would give players the freedom to create their own solutions, whilst keeping the complexity required to play the game low.

4.3.2 Scoring

Score is an essential aspect in many video games. It provides players feedback based on their performance, and create opportunities for them to want to improve. It also brings about a sense of challenge, by wanting to achieve a good score. In turn fuelling player motivation and engagement.

Initially the idea was to implement a score system that would score based on players' code. This however would be a complex system to develop, as there is not one fixed way to complete a level and hence not all code solutions will look the same. Instead, a scoring system that scored players based on the time taken to complete a level is more suitable.

The design of this scoring system is as follows:

1. Players get a fixed score at the start of the game, which would immediately begin to decrease.

2. Upon completion of a level, players' score will increase by a fixed amount, which will then continue decreasing.

3. This would repeat until the player completes the final level, upon which they would be

presented with their final score.

This style of scoring system would potentially challenge players to come up with solutions quickly in the aim for a high score. However a drawback may be that players get frustrated if they feel forced to come up with a solution quickly. To tackle this, the fixed amount a score would increase by would be a large number, and the score would decrease very gradually. Thus players are more likely to end up with a high score, which potentially may decrease any pressure to rush. In conclusion, the overall aim for this style of scoring system is to increase player engagement and bring a sense of challenge.

4.3.3 Hit points

Since the design of the game was for the character to be able to shoot, specifically at enemies. It is viable that the character can get 'hurt' by such enemies. This would involve the characters hit points decreasing. Hit points are generally present in games that involve shooting, so again it is a viable mechanic for this style of game.

The character will be given a fixed hit point at the start of the game, which will only decrease if the character walks into any enemies. This would only occur when players have not created an optimal solution for the current level. Upon losing all their hit points, or in other words if their character 'dies', the character would be reset to the initial position at the start of the same level. The player will also lose some points from their score. The users code will not reset, as to give them a chance to identify the issues with their code, amend it and try again.

This style of mechanic would give players the incentive to want to avoid such scenarios by planning and creating well-designed and thought-out code. The more the player thinks about their code, the more understanding they will develop about their code and the constructs they are working with. Specifically how these constructs work. The idea to reset the character upon death adds to this by giving players a chance to figure out where they went wrong, create a different solution and further their understanding.

4.4 Level Design

Level design is an important aspect of the design stage. The main decisions to be made for each level are which iterative and conditional constructs to introduce. The main considerations are: the difficulty of each level, and the pace of teaching.

The levels should not be too difficult that users can not complete, and nor too easy that the users get bored. As discussed, the levels should bring about an optimal balance between challenge and frustration. So the following concepts were considered to introduce: *if*, *if else* and *else* statements, *for* loops, nested *for* loops and a combination of all these.

Thus the user is not only is introduced to the basic constructs, but learns how these can be combined to create larger programs. Combining these constructs allows to produce game levels with varying levels of complexity. Initially the user is presented with basic *if* and *for* constructs, which are slowly developed on throughout the game, until they have to use most, if not all, of these constructs in order to complete a level. However care needs to be taken, that the users are familiar with certain concepts before being introduced to newer more complex ones.

Pace of teaching should be designed as to not overwhelm users with too much information, and not bore users by going very slowly. As soon as players are familiar with certain concepts, they should be introduced to new or more complex ones. To help make certain that users were as familiar as possible with these concepts, three types of levels were introduced: to complete unfinished code, debug given code, or create your own solution. These also aim to provide more functionality to the game and perhaps make the game more entertaining, as just doing only one of three objectives throughout the game may get boring.

These three types of level allow the same constructs to be reused in different scenarios. Repetition is key for learning, and these levels help provide this. At the start of introducing a new construct, users will first have to tackle a level that requires to complete unfinished code, as this can be made the most easiest of the three. This type of level, will allow the user to familiarise themselves with the construct. The next type of level they will face is to debug code. This goes a bit beyond completing unfinished code, as users can familiarise themselves with the syntax of the construct. Finally, the user can then be presented with a level that requires their own solution. This type of level should hope to bring together all the previous knowledge they gained and apply it to create a unique solution. The proposed structure of the levels can be seen in Table 4.4.1.

Table 4.4.1: Design of level structure of the game

Level	Type of Level	Programming Concepts Required
Tutorial	N/A	None
01	Unfinished code	for loop
02	Debug	for loop
03	Unfinished code	if statement
04	Unfinished code	for loop, if and else statements
05	Create your own	for loop and if statement
06	Debug	for loop, if and else if statements
07	Unfinished code	nested for loop
08	Create your own	Most

4.4.1 Tutorial

A tutorials role is to introduce the game to the player. Tutorials commonly entail the introduction of the game goals, game mechanics and rules. They can sometimes also have a walkthrough which leads the user through an example level. However, if a tutorial is too long or contains too much information, players might quickly get bored and lose any motivation they had, to play. Thus which information to present in the tutorial was carefully considered.

The tutorial will comprise of two sections. The first will be a dialogue that explains the educational goals, as they are the main objective of the game, the game goals, game mechanics excluding the functions, and rules. It would be more useful to introduce functions to players during the game when they are actually required. Constructs will also be introduced during the levels for the same reason. The second section will be a walkthrough of an example level. The example level will have already have code required to complete the level, the user will be instructed to execute it. This gives players a feel for how the functions and character interact.

Since the game can be played more than once by the same user, the tutorial will give users the option to skip and continue onto the main game. This is so users who already know how to play, do not have to spend unnecessary time recapping material already known. However throughout the game, the game interface will have a help section that contains a brief summary of the information presented in the tutorial.

4.5 User Interface
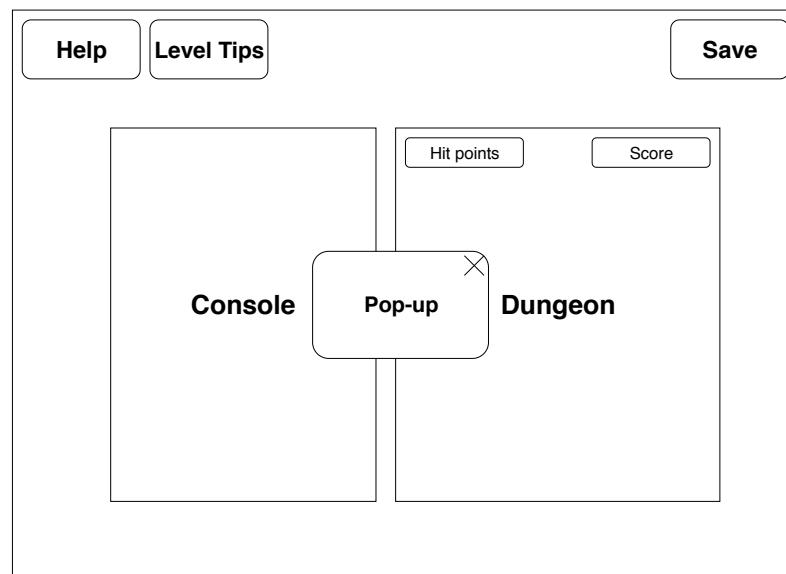
Figure 4.5.1: Design of game layout for Programmer's Dungeon

An important aspect of the interface is for it to be aesthetically pleasing. It is the first thing a user sees, so it must be designed to encourage motivation. Figure 4.5.1 shows the design layout of the game. The elements in the design were selected in order to satisfy the requirements. The description and purpose of each element is explained below:

- **Console.** An updatable console for players write and modify code. Console will also have an execute button to run code. Level requirements will also be displayed here.

- **Dungeon.** Displays the layout of the current level which includes: character, any enemies and the exit. Character will carry out the code executed in the console.

- **Pop-up.** This will only ever pop-up to introduce and explain unintroduced constructs and functions as they are required.

- **Hit points.** Displays the characters hit points.

- **Score.** Displays the players current score.

- **Help.** An interactive button which brings up a pop-up containing general help. The general help includes game rules, game goals, brief description of game mechanics including the functions and also constructs.

- **Level Tips.** An interactive button which brings up a pop-up that contains information to help to complete the current level.

- **Save.** A button that saves the players' current game state.

Chapter 5

Implementation

5.1 Technologies Used

The following subsections provide a high-level discussion of the technologies used during the implementation process. This section aims to highlight and reflect the implementation choices, any ancillary techniques used and the choice of programming language to develop the game while providing justification for such choices.

5.1.1 Source Control

Source control is a useful tool to help develop software. The source control system used for this study was GitHub. GitHub's source control system has several features that helped in the development process of the game. These are, the ability to save code locally and in their cloud servers, record and revisit code updates, and check merge changes. Code was committed every time a new functionality was added, or when there was a change to the code. This ensured that code was not committed too often, as this would be time consuming, and that the committed code was fully functioning.

5.1.2 Language Choice

The game was developed using Node.js, a JavaScript runtime environment. The game was written in JavaScript, Jade and CSS, and is entirely client side; there is no game server. This allows quick execution of code as all execution occurs within the users' browser; time saved sending and receiving messages to and from a server.

JavaScript is a programming language for frontend development. The main reason for using it is because of its simplicity, vast functionality and because it is supported in many browsers. This includes the three most popular web browsers, Google Chrome, Safari and Firefox (StatCounter,

2019). JavaScript was used for the entirety of game logic, browser control, animating page elements and creating interactive content.

Jade was used for defining the web page and scripts. It is a markup language that is simpler than HTML and has more functionality. CSS is a style sheet language and was used for describing the presentation of elements, such as positioning, colour, size and more, within the web page.

5.1.3 Character Animations

To increase the entertainment aspect of the game a sprite was created for the character, enemy and bullet using Piskel, a free online sprite editor. A sprite is a two-dimensional image that contains multiple smaller images. These smaller images are used to create an animation.
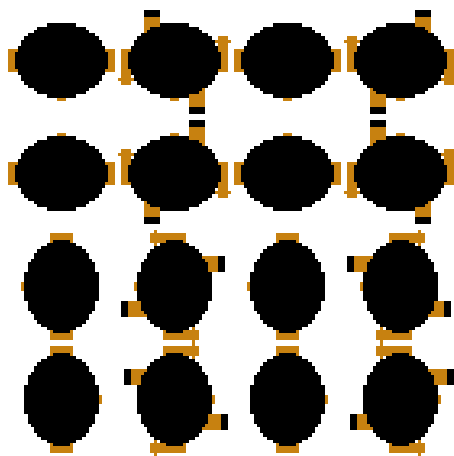
(a) Sprite sheet for character
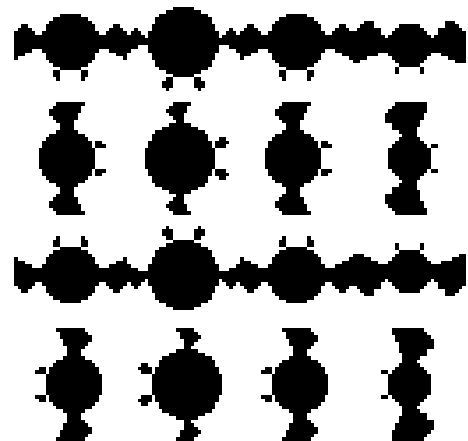
(b) Sprite sheet for enemy

Figure 5.1.1: Sprite sheets for character and enemy in Programmer's Dungeon

The JavaScript function, drawImage, was used to create the animation. drawImage takes parameters to specify which sections of an image to draw. This function was executed in a loop that updated the parameters with a counter, and thus creating the animation.

5.1.4 Hosting Game

Since one of the requirements was for the game to be accessible from home, work or school, it was necessary to host the game onto the world wide web. Anyone with a device with a browser connected to the internet would be able to access the game. This is not to mean that everyone who satisfies these requirements will play the game, but instead, users who wish to play, can do so from most locations.

There are many different hosting services that are specifically designed Node.js applications. However, I decided to use Amazon Web Services (AWS) as I had most familiarity with this software. AWS also provided a free domain name, with the URL: *http://tutorial-env.4yzpcwhdm6.us-west-2.elasticbeanstalk.com/*. However this URL is fairly long and does not reflect the contents of

the web page in any way. So a more suitable URL was chosen through the service, freenom: *http://programmersdungeon.tk/*. This was set-up to do a URL redirect to the URL provided by Amazon, as the game is hosted there. This URL is free for a year and so will be usable throughout the duration of this study.

5.2 Overview of System Architecture

The following subsections provide an overview of the non-trivial components of the game. These are the map, character and console. The user interacts with the console to influence the characters actions, which in turn updates the game state and displays the changes on the map.

5.2.1 Map

The 'map' refers to level layout in the dungeon, specifically this is the walls represented by black lines, the enemies represented by bats, and the exit tile represented by a red square. The character is not a part of the map, but instead 'within' the map. The implementation of the map is comprised of two elements. One is what the user sees (see Figure 5.2.1), and the second is a co-ordinate system that the character interacts with. This co-ordinate system has X and Y axis, with origin (the 0,0 co-ordinate) in the top left corner of the map. This co-ordinate system is measured in pixels; one unit of length is one pixel. The map that the user sees is created using graphical lines, graphical rectangles and enemy sprite animations.
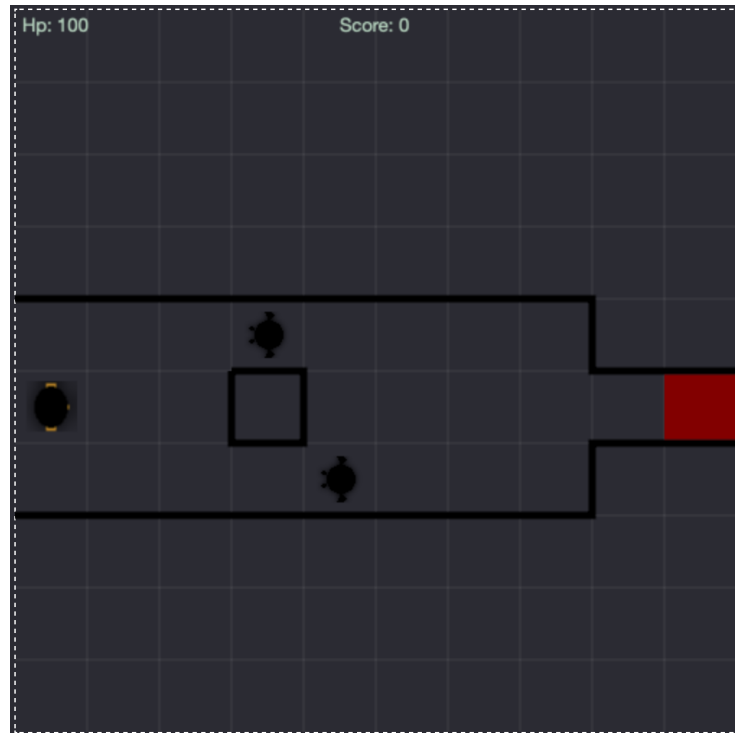

Figure 5.2.1: The map of level 6 in Programmer's Dungeon

The dimensions of the map are calculated, in pixels, using the *window.outerHeight* function. This means the map size is determined with regards to the user's browser window size. This allows the map to increase in size for larger screens and decrease in size for smaller screens. The graphical images and enemy size, conform to the size of the map as they take into account the dimensions of the map. This means that they remain in the correct location with appropriate size regardless of user screen size. Thus the game should look the same and be playable on different screen sizes.

The dark grey background and light grey lines in Figure 5.2.1, is present in every level. The light grey lines form the 'tiles' of the map. The character can only move from its current tile to any adjacent tiles, and there are ten tiles across and up the map. The distance the character moves is calculated by map.length/10. Since the map length and height are the same, this is used for the character to travel both horizontally and vertically.

5.2.2 Character

I have discussed how the character is animated and how it moves. Now I describe how it interacts with its environment, for example; getting hurt by an enemy or stopping against a wall.

Figure 5.2.2: Contact points for character and enemy in Programmer's Dungeon

The character has X and Y co-ordinates that represent its centre, which is used to move the character across the map. However, for the character to interact with its environment, it needs a 'contact' point(s). This would be used to determine whether the character is 'touching' anything else. The contact point could have easily been the characters centre. However this was not a good idea since the character would have been able to get almost half its body across a wall before having to stop. This would not seem right. So the character was given four points of contact at its edges. In Figure 5.2.2, the blue square represents the centre of the character and the red squares represent its contact points. The contact points are derived by adding/subtracting a fixed pixel length to its centre in the X and Y direction. Enemies have the same set of characteristics.

This means the character would stop just before a wall, as walls run down the middle of two tiles. For enemies, since they also have a similar configuration of contact points, the character and enemy are deemed to be touching if any of the characters's contact points are between all four of the enemys. If they are touching, then the characters hit points decrease.

5.2.3 Map Walls

The walls which are represented as black lines, are just graphical images. For them to interact with and stop objects such as the character, enemy and bullets, they need to be represented in the co-ordinate system. Such that if any of the character's contact point ever touched the contact point of a line, it would have to stop moving in that direction. Instead of defining every pixel in the co-ordinate system where a wall is present, a 2D array was used to define a group of pixels as the former would be very tedious to do. This 2D array represents the whole map and is comprised of three elements. A 0 represents space that could be moved around in, 1 represents a wall, and 2 represents the exit. The character can move around in areas where there is a 0, is stopped from moving if in contact with a 1, and completes the level if in contact with a 2.

5.2.4 Console

```
1  //Debug the code: Press Run to see what
2  //happens and then Reset.
3
4
5  //You can add multiple conditions to an 'if'
6  //statement by adding the && operator.
7
8  for (var i=15; i>0; i--) {
9
10     if (enemyPosition('right')) {
11         player.shoot('right');
12     }
13
14
15     if (wallPosition('right')) {
16         player.move('up');
17     }
18     else if (
19         wallPosition('right') &&
20         wallPosition('up')
21     ) {
22         player.move('down');
23     }
24
25
26     player.move('right');
27
28 }
```

Run

Figure 5.2.3: The default console layout for level 6 in Programmer's Dungeon

The console is comprised of two components; Ace, an open-source code editor written in JavaScript (Ace, 2019), and a 'Run' button. The Ace functionality has a method to execute any code within the code editor. The button was created to call this method when clicked.

The type of level is defined in the comments at the top console. This is explained within the tutorial. In some levels, there are a second set of comments below the first. These are there to provide additional information. This may be describing a new construct, or providing more information about an already introduced construct (see Figure 5.2.3). This will only occur when the specific construct is required to complete the level.

Turning the code within the console into actions carried out by the character involves a complex mechanism. The player.move function moves the character to an adjacent tile. The direction the character moves is determined by the parameter within the function. The enemyPosition and wallPosition functions determine if an enemy or wall is adjacent to the character respectively. These functions use the contact points and centre points mentioned above.

A potentially issue with this implementation is that even though the level type is defined, it is

not enforced. This is because the console is completely editable. A consequence of this is that any code that is present at the start of a level can be removed, and the level can subsequently be completed by using functions without any control structures. All levels can be completed this way. To resolve this, two solutions would be required. In the levels which require debugging or completing code, certain parts of the code must not be editable. Thus forcing users to use the control structures presented. On top of this, any solution that the user tries to run can only be executed if the solution contains a control structure. Thus levels that require users to create their own solutions, must have at least one construct. However this could still be worked around, and so is not an ideal solution. These solutions were not implemented as they are complex. They would take a considerable amount of time to implement. Instead, users are told within the tutorial to complete the levels as per the type of level specified within the comments. There is no mention anywhere in the game that the levels can be completed just by using functions.

Chapter 6

Testing

Throughout development, the game was tested to ensure that it was working as expected. Once implementation was complete, three stages of testing were defined. First system testing, then obtaining initial feedback from user playthroughs, and finally user testing. This chapter will discuss these testing methods in more detail, specifically justification for their adoption and the proposed methodologies.

6.1   System Testing

Testing was carried out throughout development. This was ensure that the game functionalities were working as expected and graphics were represented correctly. This covered the majority of unit test cases and integration test cases; every time a component was added, it was tested, and every time two components were combined, their input and outputs were tested.

Once the game is implemented as per the requirements and design, the game will undergo system testing. System testing refers to testing the complete and integrated software. This translates to conducting test cases involving playing through the entire game (in other words, a playthrough). To determine the focus of the test cases, two objectives were determined:

1. Check if the functional requirements are met.

2. Conduct certain test cases and edge cases were there may be a high chance to discover bugs.

The first objective is essential; the game was developed for teaching iterative and conditional control structures, and this was captured in the requirements. Determining if the non-functional requirements were fulfilled requires input from users. This can only be achieved after user testing. For the second objective, since the game was solely developed by myself, I could be certain on which areas may require testing. This includes areas that were perhaps not well thought over during the design phase. For this reason, and the time constraints on its development, the creation of a

formal test plan was deemed unnecessary.

6.2   Initial Feedback

Feedback from initial user playthroughs would be useful in order to identify any obvious overlooked design flaws. Since the game was solely developed by myself, any outside opinions would be very useful. The gathered feedback would then be used to make improvements in the game.

The main objective of the phase was to determine what users felt about the difficulty, usability and enjoyment of the game. As well as any other suggestions that might have. These tests would be conducted using the think-aloud protocol. This is a method of usability testing. Usability testing refers to evaluating a product by testing it on users. Furthermore, this protocol is claimed to be widely used in the gaming industry and is very effective (Desurvire and El-Nasr, 2013). Thus it was chosen to be used in this stage. The technique involves players verbally expressing their thoughts during gameplay. Participants are advised to say whatever comes into their mind. This might include what they are experiencing, their actions or their feelings. This data is then collated.

To ensure feedback is as relevant as possible, participants with appropriate backgrounds and interests will be considered. In addition to the think-aloud protocol, participants will be asked afterwards to provide suggestions for any changes to the game in areas they felt needing improvement. This was so if the participant provided criticism on the game during the think-aloud protocol, they could also provide suggestions for improvements on those specific aspects.

6.3   User Testing

Once system testing and initial feedback and improvements is complete, the game can be tested by users. The aim of this stage is to measure to which extent the non-functional requirements were fulfilled.

The experiment is a pretest-posttest design, comprised of three parts. The first part will require participants to complete a pretest questionnaire. This questionnaire will assess participants interests and knowledge in iterative and conditional structures. The second part requires the participant to play the game. They will be asked to try and get as far as they can; completing the game will not be a requirement. This is because participants can get stuck and can not continue, may lose interest and no longer wish to play or any other reason. After they finish playing, the final part of the experiment will require participants to complete a posttest questionnaire. This questionnaire will assess users enjoyment and perceived difficulty of the game, their knowledge on iterative and conditional structures, their motivations and ask for any possible feedback or suggestions.

Chapter 7

Results

7.1 System Testing Results

All functional requirements were easy to check as game functionality was developed based on these requirements. Moreover, these functionalities would have also been checked during implementation. However, conducting specific test cases allowed to determine whether they were working as expected.

Playthroughs were very useful as they identified several issues, of which all were resolved. These were:

1. When played on a larger screen size than the one used for development, the character did not increase in size, whilst the map did. The character was able to get extremely close to the map walls, as its contact points remained close around the character. The characters size was changed to increase according to map size.

2. Some features of the game did not work in Firefox. Corresponding JavaScript functions were changed in order to support Firefox.

3. Clicking play again at the end of the game resulted in duplicate texts in the tutorial box.

4. In larger screen sizes, help section extends outside browser screen size. Text box was made wider to shorten text box height.

The game was also tested on Internet Explorer (IE), although this was not part of the requirements. IE is the fourth most popular non-mobile browser (StatCounter, 2019). However the game did not display correctly; the console was overlapping the map. This may be due to IE reading and interpreting some JavaScript code differently compared to the three most popular browsers. Since only 2.58% of browser users use IE and time constraints, this was deemed unnecessary to fix.

7.2 Initial Feedback and Improvements

Two participants were selected to conduct playthroughs using the think-aloud protocol. Results from the think-aloud and suggestions were collated and some improvements were made to the game.

7.2.1 Feedback

Participant 1

The first participant was a first year undergraduate computer science student from Royal Holloway University. The following comments were gathered:

- Too many comments in the console. There is a lot of information to take in. No suggestion.

- Level 6, which requires debugging a more complex *for* and *if* code, was too difficult. Suggested making level easier.

- Level 7, which requires completing a nested *for* loop, too difficult. No suggestion.

- Last level, level 8, which requires creating your own code to a complex level, was most challenging and enjoyable level.

Participant 2

The second participant was an aerospace engineer graduate who was interested in learning how to code. The following comments were gathered:

- Too much text in the tutorial. Suggested making it more precise.

- Colour scheme of tutorial does not match colour scheme of the game. Suggested making tutorial match colour scheme of console.

- Reset level removes any changes to the code by resetting the text in the console back to default. Suggested not resetting the code, as you are not able to see your mistakes.

- Suggested scrolling animation for text in the tutorial text box.

- Help section is very useful because you can check construct and function definitions.

7.2.2 Changes
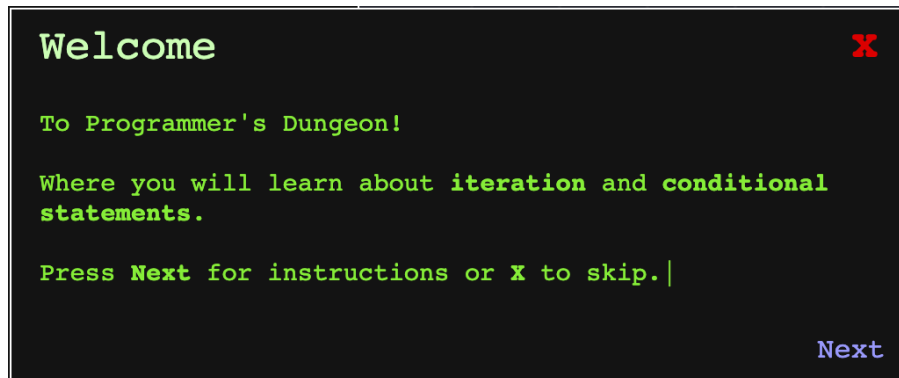
Figure 7.2.1: The tutorial text box in Programmer's Dungeon

The feedback obtained was extremely helpful. Feedback obtained identified issues to do with level difficulty, game appearance and the information provided in the game. Most suggestions only required changing some colour or text, so this was simple to do. However, making the tutorial text scroll was a more complex task. This was implemented because this was deemed to make the tutorial more interesting and engaing. The following changes were made:

- Tutorial text is more concise; only provides information to introduce the game. Some information was moved to the help section.

- Colour scheme of tutorial now matches the colour scheme of the console (see Figure 7.2.1).

- Tutorial text scrolls across the text box.

- Comments in the console about type of level and construct information was thought over and made more concise.

- In level 6 and 7, constructs were kept the same, but the code was changed to make the level easier by making solutions more obvious.

These changes should result in making the game look more aesthetically pleasing, the difficulty curve smoother, and not overwhelm users with too much information.

7.3 User Testing

The experiment was completed by 21 participants. The mean age of participants was 19 with a standard deviation (SD) of 1.7 years. 66% of participants already had experience with iterative and conditional constructs.

7.3.1 Learning

In the pretest and posttest questionnaire, participants were asked to rate their perceived ability using iterative and conditional constructs. In addition to this, both questionnaires contained two questions in order to test participants actual knowledge on iterative and conditional constructs respectively. These questions were designed to assess how participants perceived and actual learning on iterative and conditional constructs had been affected through playing the game.

Participants to rate their perceived ability on a scale between 1 to 5, with 1 being not confident and 5 being very confident. Before playing the game, the mean perceived ability was 2.76 (SD = 1.26). After playing the game, the mean perceived ability was 3.48 (SD = 1.12). A paired t test showed $t(20) = 2.904$, with $p = 0.0076$ less than 0.05. Therefore there is a significant increase in perceived confidence after playing the game.

What would you rate your ability in using iterative and conditional statements?
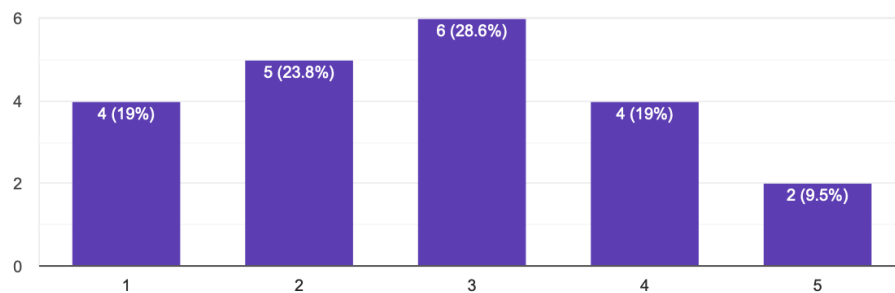
21 responses

Figure 7.3.1: Results of participant confidence in iterative and conditional constructs pretest

What would you rate your ability in using iterative and conditional statements now?
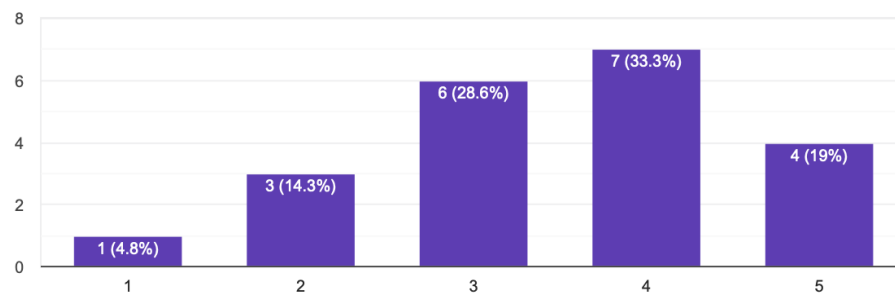
21 responses

Figure 7.3.2: Results of participant confidence in iterative and conditional constructs posttest

Participants were also asked questions which tested their knowledge on iterative and conditional constructs. These were multiple choice questions; a) which of the following are used to create conditional statements, b) which of the following are used to create iterative statements. To both these questions they were given four options: for, if, if else and nested for. Table 7.3.1 highlights pretest posttest results for iterative question. Table 7.3.2 highlights pretest posttest results for conditional question.

Table 7.3.1: Results from pretest posttest questionnaires on iterative constructs

	Pretest	Posttest
for	17	19
nested for	15	19
if	5	3
if else	5	4

Table 7.3.2: Results from pretest posttest questionnaires on conditional constructs

	Pretest	Posttest
for	6	3
nested for	7	4
if	17	20
if else	16	18

Results from these tables show that after playing the game, the number of correct answers increased. This supports the result of increase perceived ability in correctly using iterative and conditional constructs. Thus players did actually further their knowledge through playing the game. This indicates that the game fulfils requirement 3.1.1 "Game must improve novice programmers' knowledge on iterative and conditional control structures".

However, when asking participants to assess their knowledge on a given topic, participants can mistakenly assess their understanding of a topic to be better than it really is. This is called the Dunning-Kruger effect (Dunning, 2011), and participants are more susceptible to this bias before learning a topic compared to afterwards. This could explain how some participants rated their ability lower after playing the game, specifically 2 participants, as from playing the game their knowledge of iterative and conditional constructs is not likely to decrease.

7.3.2 Engagement

To assess participant engagement in the game, two questions were asked. The first asked participants how many levels they completed. There was a total of eight levels. The second asked how they found the game in terms of enjoyment. This does not give a precise picture in terms of player engagement, however, it is still helpful.

The mean number of levels completed was 5, with the majority of participants, 28.6% having completed all 8 levels. The questionnaire asked participants to rate their enjoyment from a scale of 1 to 5, with 1 being not enjoyable and 5 being very enjoyable. The mean enjoyment felt was 3.52 with a standard deviation of 1.14. It is clear from this that participants somewhat enjoyed the game.

In addition, participants were also asked to rate on a scale between 1 to 5, how difficult they felt the game was. The mean score was 3.33 (SD = 1.28). The majority of participants, 33.3% rated the game at a 4. This suggests that the game is slightly difficult.

Given the majority of participants felt the game was difficult, but on average completed more than half the levels, coupled with the fact the game was enjoyable, indicates that the game was challenging yet participants were also engaged. Thus this partially fulfils requirements 3.1.6 "Player should be engaged throughout the game" and 3.1.7 "Player should feel challenged".

7.3.3 Motivation

A few different questions were asked to assess participant motivation. Specifically, if their interest in iterative and conditional constructs increase after playing the game, if they would consider learning more complex iterative and conditional constructs, and if they would consider playing games to learn iterative and conditional constructs. The aim of these questions were assess how the game affected their motivation for continued learning on specific subject.

Both questionnaires asked participants to rate their interest in iterative and conditional statements. Participants had to rate between 1 and 5, with 1 being not interested and 5 very interested. The mean interest score was 2.81 pretest (SD = 1.03) and 3.48 posttest (SD = 0.98), with a paired t test, $t(20) = 2.697$ and $p = 0.0095$ which is less than 0.05. Therefore there is a significant increase in participant interest in iterative and conditional constructs after playing the game.

When asked whether participants would consider learning more complex iterative and conditional constructs, 7 said yes, 9 said maybe and 5 said no. However when asked whether they would consider learning the same thing by playing games, 15 said yes, 6 said no and none said no.

Participants motivation to learn more complex programming increased, but only if the material was being taught in the form of a game. Otherwise, it is inconclusive if participants would consider learning if material was taught by other means. This indicates that the game fulfils requirement 3.1.9 "Game may foster player motivation to learn more complex programming" if the material is taught through games.

7.3.4 User Feedback

Participants at the end of the posttest questionnaire, were asked to make any comments on any feedback or suggestions. Of the 21 participants, only 12 left comments.

Three participants made comments about the console being completely editable. All three suggested either forcing users to use for loops and if statements or making parts of the code not

editable. These issues and solutions were already discussed previously. No different solutions were provided.

Another three participants mentioned that made them want to learn more about programming and iterative and conditional constructs.

One participant only managed to complete the first level. Their comment mentioned that what was required from them in the game was unclear. Two other participant mentioned that there was too much information in the tutorial and that it could have been made more concise. However as this game has an additional educational goal compared to other games, it is necessary to explain certain pieces of information to be able to play the game.

7.3.5   Threats to Validity

This experiment has a number of limitations that limit its generalisability. First the sample size of participants was not very large. A larger sample size would provide a more accurate result, more easily identify outliers and provide a smaller margin of error. All of which would help confirm or reject the findings of this experiment.

The second threat to validity is that participants were comprised of people with and without prior experience to iterative and conditional constructs. Results may be more accurate and perhaps different if two experiments were carried out with these two groups separately. The experiment with the experienced group could be further separated based on how familiar participants were with iterative and conditional constructs. Another consequence is that some participants could have been be very experienced, and in turn may have not learn much, if not nothing, from the game. Moreover, two participants rated their perceived ability as 5 in both the pretest and posttest.

Finally, participants were also asked in the pretest if they were motivated to learn programming. Three participants answered no, two answered unsure and six answered somewhat. This is not ideal, as they went on to play a game involving programming. This may have produced a bias within the results. Again splitting these groups up into different experiments would provide more accurate results.

Chapter 8

Conclusions

The main aim of this study was to design and develop an educational game for the purpose of teaching iterative and conditional control constructs. Educational games and educational game design was researched in order to gain an understanding of the current state of the field. This research highlighted areas to take into consideration when developing an educational game. Through this research, the requirements of the game were formed. The game was then designed as per the problem set out in the requirements and issues discussed in the literature review.

The design described the mechanics of the game and the layout. The implementation focused on aspects of the non-trivial aspects of the game. This included the map, console and the interaction between in-game entities. An issue was also identified with the implementation of the console.

After implementation, three stages of testing were conducted. These were system testing, initial playthroughs and improvements and finally user testing. System testing and initial playthroughs identified several game issues, which were subsequently resolved. The final stage of testing, user testing, was conducted through an experiment. The aims of the experiment were to identity to which extent the non-functional requirements of the game were fulfilled. The experiment consisted of a pretest-posttest design. The results were particularly used to assess participant learning, engagement and motivation.

Results from user testing revealed that user knowledge of iterative and conditional constructs increased significantly after playing the game. This supports the findings in the literature review, that educational games improve learning. Users also found the game enjoyable. This was a key objective for the game. Results highlighted that the game may be engaging, but it was difficult to reach any conclusions. Through playing the game, participants were more motivated to learn more complex programming concepts. This was highlighted as an issue within the literature review. Moreover, participants were more interested in learning more complex programming concepts, when the material is delivered through games.

Chapter 9

Future Work

Taking into consideration participant feedback from the user testing stage, the following changes would be implemented in the game.

1. Parts of the console would be made non editable. This would be particularly useful in the levels that require debugging or completing code.

2. Text in the tutorial would be made more concise.

Finally, further investigation need to be conducted in order to determine how the knowledge gained by participants was effected over longer periods of time. This would require conducting longitudinal studies.

Bibliography

AAStocks, 2016. Number of active video gamers worldwide from 2014 to 2021 (in millions) [Online]. Accessed: 2019-04-12. Available from: `https://www.statista.com/statistics/748044/number-video-gamers-world`.

Abt, C.C., 1970. *Serious games: The art and science of games that simulate life in industry, government and education.* New York, NY: Viking.

Ace, 2019. The high performance code editor for the web [Online]. Accessed: 2019-05-04. Available from: `https://ace.c9.io/`.

ACM-IEEE, 2013. Computer science curricula 2013 [Online]. Accessed: 2019-04-02. Available from: `https://www.acm.org/binaries/content/assets/education/cs2013_web_final.pdf`.

AlliedMarketResearch, 2017. Serious games market [Online]. Accessed: 2019-04-04. Available from: `https://www.alliedmarketresearch.com/serious-games-market`.

Arkorful, V. and Abaidoo, N., 2015. The role of e-learning, advantages and disadvantages of its adoption in higher education. *International Journal of Instructional Technology and Distance Learning*, 12(1), pp.29–42.

Barnes, T., Richter, H., Powell, E., Chaffin, A. and Godwin, A., 2007. Game2learn: Building cs1 learning games for retention. *SIGCSE Bull.*, 39(3), pp.121–125.

Bennedsen, J. and Caspersen, M.E., 2007. Failure rates in introductory programming. *AcM SIGcSE Bulletin*, 39(2), pp.32–36.

Bisson, C. and Luckner, J., 1996. Fun in learning: The pedagogical role of fun in adventure education. *Journal of Experiential Education*, 19(2), pp.108–112.

Breuer, J. and Bente, G., 2010. Why so serious? on the relation of serious games and learning. *Journal for Computer Game Culture*, 4, pp.7–24.

Concannon, F., Flynn, A. and Campbell, M., 2005. What campus-based students think about the quality and benefits of e-learning. *British Journal of Educational Technology*, 36(3), pp.501–512.

Connolly, T.M., Boyle, E.A., MacArthur, E., Hainey, T. and Boyle, J.M., 2012. A systematic

literature review of empirical evidence on computer games and serious games. *Computers & Education*, 59(2), pp.661–686.

Cordova, D.I. and Lepper, M.R., 1996. Intrinsic motivation and the process of learning: Beneficial effects of contextualization, personalization, and choice. *Journal of educational psychology.*, 88(4), pp.715–730.

Csikszentmihalyi, M., Abuhamdeh, S. and Nakamura, J., 2014. Flow. *Flow and the foundations of positive psychology.* Springer, pp.227–238.

Desurvire, H. and El-Nasr, M.S., 2013. Methods for game user research: studying player behavior to enhance game design. *IEEE computer graphics and applications*, 33(4), pp.82–87.

Djaouti, D., Alvarez, J., Jessel, J.P. and Rampnoux, O., 2011. *Origins of serious games*, Springer London, pp.25–43.

Dunning, D., 2011. The dunning–kruger effect: On being ignorant of one's own ignorance. *Advances in experimental social psychology.* Elsevier, vol. 44, pp.247–296.

Ferguson, C.J., 2007. The good, the bad and the ugly: A meta-analytic review of positive and negative effects of violent video games. *Psychiatric Quarterly*, 78(4), pp.309–316.

Feurzeig, W. et al., 1969. Programming-languages as a conceptual framework for teaching mathematics. final report on the first fifteen months of the logo project.

Filsecker, M. and Hickey, D.T., 2014. A multilevel analysis of the effects of external rewards on elementary students' motivation, engagement and learning in an educational game. *Computers & Education*, 75, pp.136–148.

Gee, J.P., 2003. What video games have to teach us about learning and literacy. *Comput. Entertain.*, 1(1), pp.20–20.

Giannakos, M.N., 2013. Enjoy and learn with educational games: Examining factors affecting learning performance. *Computers & Education*, 68, pp.429–439.

Gidget, 2019. [Online]. Accessed: 2019-04-17. Available from: `https://www.helpgidget.org`.

Girard, C., Ecalle, J. and Magnan, A., 2013. Serious games as new educational tools: how effective are they? a meta-analysis of recent studies. *Journal of Computer Assisted Learning*, 29(3), pp.207–219.

Granic, I., Lobel, A. and Engels, R.C.M.E., 2014. The benefits of playing video games. *American psychologist.*, 69(1), pp.66–78.

Hayes, E.R. and Games, I.A., 2008. Making computer games and design thinking: A review of current software and strategies. *Games and Culture*, 3(3-4).

Koulouri, T., Lauria, S. and Macredie, R.D., 2014. Teaching introductory programming: A quantitative evaluation of different approaches. *Trans. Comput. Educ.*, 14(4), pp.26:1–26:28.

Lee, M.J. and Ko, A.J., 2011. Personifying programming tool feedback improves novice programmers' learning. *Proceedings of the seventh international workshop on computing education research.* pp.109–116.

Lee, M.J. and Ko, A.J., 2015. Comparing the effectiveness of online learning approaches on cs1 learning outcomes. *Proceedings of the eleventh annual international conference on international computing education research.* ACM, pp.237–246.

Lehrer, R., 1986. Logo as a strategy for developing thinking? *Educational Psychologist*, 21(1-2), pp.121–137.

Leppänen, M., 2007. A context-based enterprise ontology. *International conference on business information systems.* Springer, pp.273–286.

Malan, D.J. and Leitner, H.H., 2007. Scratch for budding computer scientists. *ACM Sigcse Bulletin*, 39(1), pp.223–227.

Malone, T. and Lepper, M., 1987. Making learning fun : a taxonomic model of intrinsic motivations for learning. *Conative and Affective Process Analysis.*

Michael, D.R. and Chen, S.L., 2005. *Serious games: Games that educate, train, and inform.* Muska & Lipman/Premier-Trade.

Moreno-Ger, P., Burgos, D., Martínez-Ortiz, I., Sierra, J.L. and Fernández-Manjón, B., 2008. Educational game design for online education. *Computers in Human Behavior*, 24(6), pp.2530–2540.

Muratet, M., 2012. Prog&play [Online]. Accessed: 2019-04-17. Available from: `http://progandplay.lip6.fr/index_en.php`.

Muratet, M., Torguet, P., Viallet, F. and Jessel, J.P., 2011. Experimental feedback on prog&play: a serious game for programming practice. *Computer graphics forum.* vol. 30, pp.61–73.

Newzoo, 2017. Distribution of video gamers worldwide in 2017, by age group and gender. [Online]. Accessed: 2019-04-12. Available from: `https://www.statista.com/statistics/722259/world-gamers-by-age-and-gender/`.

Newzoo, 2018. *2018 global games market report* [Online]. Available from: `https://resources.newzoo.com/hubfs/Reports/Newzoo_2018_Global_Games_Market_Report_Light.pdf?`

Prensky, M., 2001. Digital natives, digital immigrants part 1. *On the Horizon*, 9, pp.1–6.

Prensky, M., 2003. Digital game-based learning. *Computers in Entertainment (CIE)*, 1(1), pp.21–21.

Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., Millner, A., Rosenbaum, E., Silver, J.S., Silverman, B. et al., 2009. Scratch: Programming for all. *Commun. Acm*, 52(11), pp.60–67.

Riccardi, G.A., 1981. The independence of control structures in abstract programming systems. *Journal of Computer and System Sciences*, 22(2), pp.107–143.

Ritterfeld, U. and Weber, R., 2006. Video games for entertainment and education. *Playing Video Games: Motives, Responses, and Consequences*, pp.399–413.

Robins, A., Rountree, J. and Rountree, N., 2003. Learning and teaching programming: A review and discussion. *Computer science education.*, 13(2), pp.137–172.

Roungas, B. and Dalpiaz, F., 2015. A model-driven framework for educational game design. *International conference on games and learning alliance.* Springer, pp.1–11.

Sangrà, A., Vlachopoulos, D. and Cabrera, N., 2012. Building an inclusive definition of e-learning: An approach to the conceptual framework. *The International Review of Research in Open and Distributed Learning*, 13(2), pp.145–159.

Scratch, 2019. Scratch statistics [Online]. Accessed: 2019-04-16. Available from: `https://scratch.mit.edu/statistics/`.

Sherry, J., Greenberg, B., Lucas, K. and Lachlan, K., 2006. Video game uses and gratifications as predictors of use and game preference. *International Journal of Sports Marketing and Sponsorship*, 8, pp.213–224.

Sicart, M., 2008. Defining game mechanics. *Game Studies*, 8(2).

StatCounter, 2019. Browser market share worldwide [Online]. Accessed: 2019-04-28. Available from: `http://gs.statcounter.com/browser-market-share`.

Susi, T., Johannesson, M. and Backlund, P., 2007. *Serious games : An overview.* University of Skövde, School of Humanities and Informatics.

UniversityOfBath, 2018. Department of computer science programme catalogue 2018/9 [Online]. Accessed: 2019-04-18. Available from: `http://www.bath.ac.uk/catalogues/2018-2019/cm/USCM-AFB06.html`.

Vygotsky, L.S., 1978. *Mind in society: The development of higher psychological processes.* Harvard university press.

Wang, H. and Sun, C.T., 2011. Game reward systems: Gaming experiences and social meanings. *Digra conference.* pp.1–15.

Watson, C. and Li, F.W., 2014. Failure rates in introductory programming revisited. *Proceedings of the 2014 conference on innovation & technology in computer science education.* ACM, pp.39–44.

Whitton, N., 2011. Game engagement theory and adult learning. *Simulation & Gaming*, 42(5), pp.596–609.

Wikiversity, 2019. Control structures [Online]. Accessed: 2019-04-18. Available from: `https://en.wikiversity.org/wiki/Control_structures`.

Wilkinson, P., 2016. *A brief history of serious games*, Springer, pp.17–41.

Wouters, P., Nimwegen, C. van, Oostendorp, H. van and Spek, E.D. van der, 2013. A meta-analysis of the cognitive and motivational effects of serious games. *Journal of Educational Psychology*, 105(2), pp.249–265.

Zhang, D., Zhao, J.L., Zhou, L. and Nunamaker, Jr., J.F., 2004. Can e-learning replace classroom learning? *Commun. ACM*, 47(5), pp.75–79.

Zyda, M., 2005. From visual simulation to virtual reality to games. *Computer*, 38(9), pp.25–32.

Appendices

Appendix A

Screenshots

These are screenshots of the game as they were played by users in the experiment.
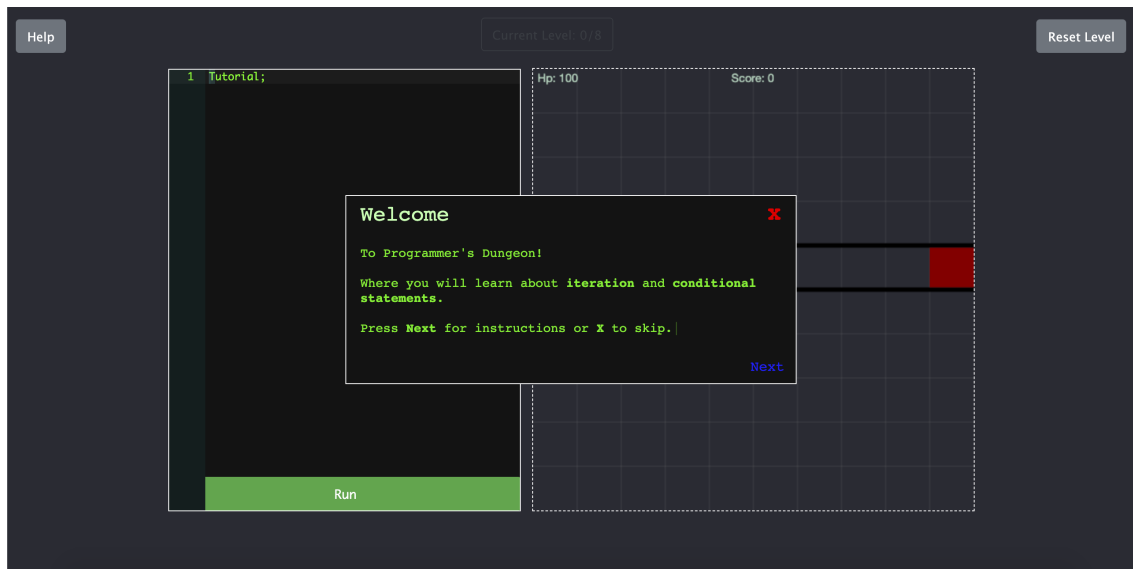

Figure A.0.1: Screenshot of the start of Programmer's Dungeon

Figure A.0.2: Screenshot of the tutorial explaining the console in Programmer's
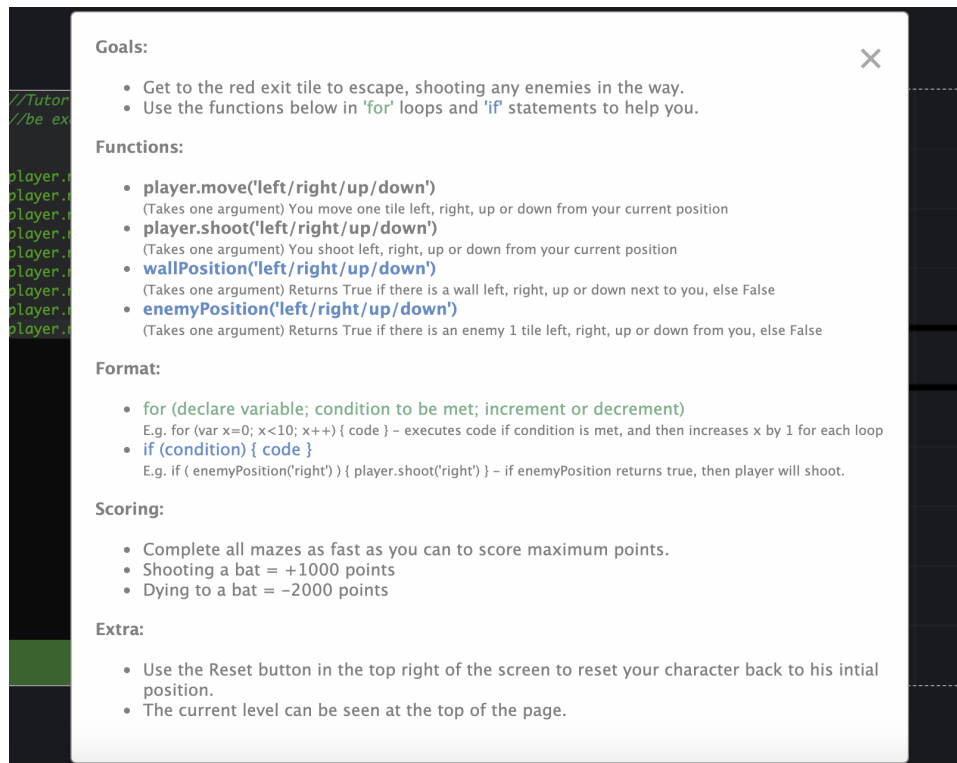Dungeon

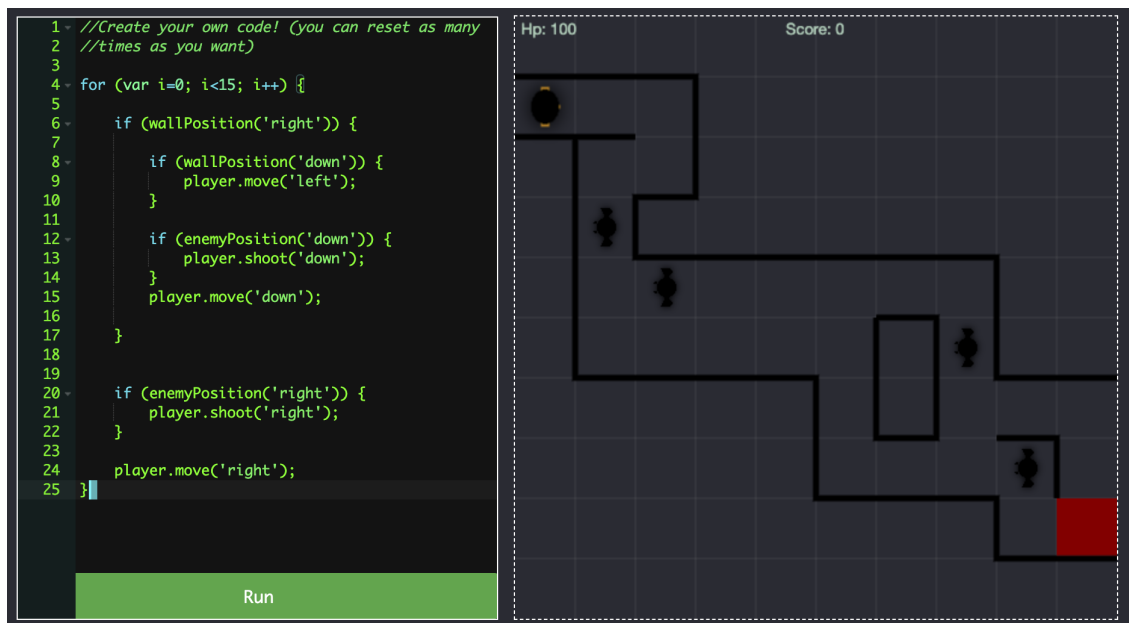**Figure A.0.3: Screenshot of the help section in Programmer's Dungeon**

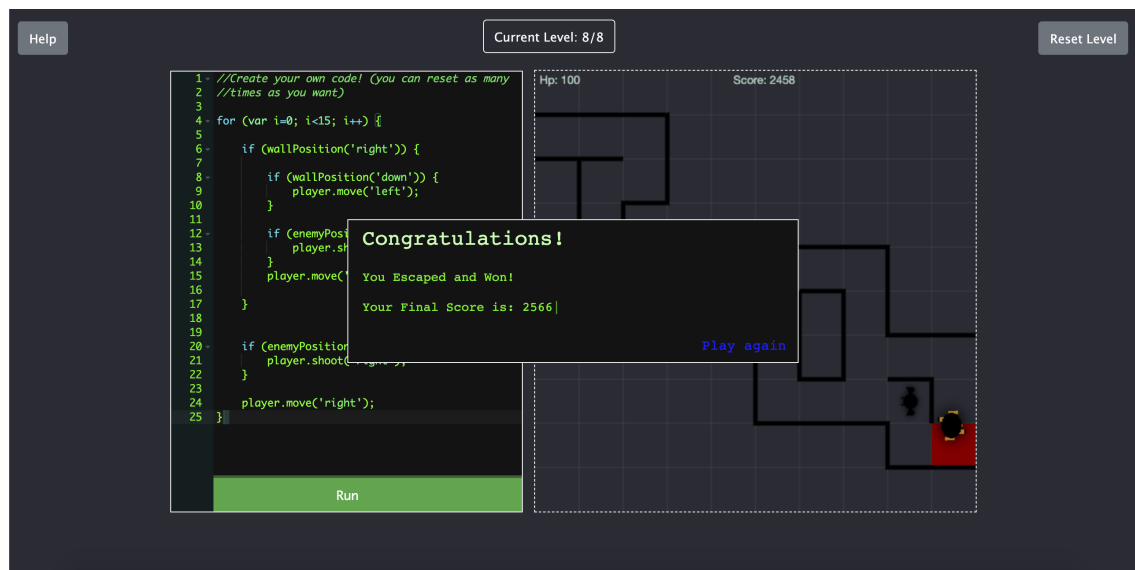**Figure A.0.4: Screenshot of the a solution to the final level in Programmer's Dungeon**

Figure A.0.5: Screenshot of end game state in Programmer's Dungeon

Appendix B

Ethics Checklist

UNIVERSITY OF BATH

Department of Computer Science

13-POINT ETHICS CHECK LIST

This document describes the 13 issues that were considered carefully before involving other people ('participants') for the collection of information as part of this project.

1. Have you prepared a briefing script for volunteers?

Before participation, participants were explained what they were required to do, the kind of data that will be gathered and how it will be used.

2. Will the participants be using any non-standard hardware?

Participants completed the tests either in their own homes, school or work. They chose their own form of hardware in order to carry out the tests. There is no requirement of special hardware in order to carry out any of the tests.

3. Is there any intentional deception of the participants?

To my understanding, no crucial information was withheld from participants. The information provided was of relevance to the study and their participation.

4. How will participants voluntarily give consent?

Participants were asked over email if they would like to partake in the tests. The only requirement to partake was to have a computer, internet connection and browser. The game could be played straight away by following a URL. The results were gathered through Google Forms, which again was accessed by following a URL.

5. Will the participants be exposed to any risks greater than those encountered in their normal work life?

The tests only required participants to use a computer for a short period of time. This risk of harm from this is deemed greater than in ordinary life.

6. Are you offering any incentive to the participants?

No incentive was offered to the participants for their participation.

7. Are any of your participants under the age of 16?

No participants were under the age of 16.

8. Do any of your participants have an impairment that will limit their understanding or communication?

No participants reported any impairments that would limit their understanding or communication. Participants were asked if they had any such impairments before their participation.

9. Are you in a position of authority or influence over any of your participants?

No authority or influence was demonstrated over the participants. This includes to their participation and their continued participation.

10. Will the participants be informed that they could withdraw at any time?

Participants were informed during the brief that they could withdraw from the entire test at any time.

11. Will the participants be informed of your contact details?

Participants were given my name, email and the University I attend. All participants must be able to contact the investigator after the investigation. They should be given the details of the Unit Lecturer or Supervisor as part of the debriefing.

12. Will participants be de-briefed?

The participants were de-briefed after the tests, in order to enable them to understand the nature of the investigation. There were also given mine and my Supervisor, Dr Alan Hayes' contacts details.

13. Will the data collected from the participants be stored in an anonymous form?

Results from the observed playthroughs will be stored on Microsoft Word. This document will saved only on my local machine. No personal information will be collected. Results from the user testing were collected and stored in Google Forms. The form did not collect information that could identify particiapnts.

Appendix C

Initial Playthroughs

C.1   Brief

Thank you for volunteering to take part in this initial playthrough. The game you will be playing is an educational game designed to teach iterative and conditional control structures.

You will be asked to play this educational game and think aloud any thoughts that come up while playing. I will be writing up notes on Microsoft Word, based on your thoughts and performance. After playing, you will be given the opportunity to express any feedback and suggestions you may have. You will be given the chance to review the information collected at the end.

Your results will be used to assess the game and make any changes if it improves the game. You can withdraw from this test at any point. The data will be entirely anonymous, no personal information will be collected or used. The results collected will only be used for the purpose of this dissertation. Afterwards I will give you mine and my supervisors contact details, if you wish to get in touch for any reason.

C.2   De-brief

Thank you for taking part in this test. I hope you enjoyed the game. You will now be given the opportunity to go over the notes taken.

The data will be entirely anonymous, no personal information will be collected or used. Your results will be used to assess the game and make any changes if it improves the game. The results collected will only be used for the purpose of this dissertation. If you would like to contact me or my supervisor, our emails are ak2022@bath.ac.uk and ah347@bath.ac.uk.

Appendix D

User Testing

D.1   Brief

Thank you for volunteering to take part in this test. The test is comprised of three parts. The game you will be playing is an educational game designed to teach iterative and conditional control structures.

First you will be asked to complete a questionnaire, designed to assess your knowledge on iterative and conditional control structures. This should take roughly two minutes. You will then be asked to played to play the educational game. Try to get as far as you can. You are not forced to complete the game. Finally, you will be asked to complete another questionnaire, designed to assess the impacts and your thought about the game.

You can withdraw from this test at any point. The data will be entirely anonymous, no personal information apart from your age will be collected. The results collected will only be used for the purpose of this dissertation. Afterwards I will give you mine and my supervisors contact details, if you wish to get in touch for any reason.

D.2   De-brief

Thank you for taking part in this test. I hope you enjoyed the game. The data collected will be used to inform the effectiveness of the game in terms of learning, engagement and motivation. The results collected will only be used for the purpose of this dissertation.

If you would like to contact me or my supervisor, our emails are ak2022@bath.ac.uk and ah347@bath.ac.uk.

D.3   Pretest Questions

1. How old are you?

2. Have you come across iterative and conditional statements before? For example for loops and if statements.

3. What would you rate your interest in iterative and conditional statements?

4. What would you rate your ability in using iterative and conditional statements?

5. Which of the following are used to create conditional statements

6. Which of the following are used to create iterative statements

7. Are you motivated to learn programming?

D.4   Posttest Questions

1. Was the game fun to play?

2. How many levels did you complete?

3. How did you find the overall difficulty?

4. What would you rate your interest in iterative and conditional statements now?

5. What would you rate your ability in using iterative and conditional statements now?

6. Which of the following are used to create conditional statements

7. Which of the following are used to create iterative statements

8. Would you consider learning more complex iterative and conditional statements?

9. Would you consider playing games to learn more complex iterative and conditional statements?

10. Was the interface clear and easy to use?

11. Do you have any feedbacks or suggestions?

Appendix E

Questionnaire Results

E.1 Pretest

Table E.1.1: Individual participant results from pretest questionnaire

Participant	Q1	Q2	Q3	Q4	Q5	Q6	Q7
A	16 - 18	Yes	3	3	if, if else	while, nested for, do while	Yes
B	19 - 21	Yes	3	3	for, if, if else, nested for	nested for, for	Unsure
C	19 - 21	Yes	3	4	for, if, if else, nested for	if else, nested for, for	Yes
D	16 - 18	Yes	2	3	if, if else	nested for, for	Yes
E	19 - 21	Yes	2	3	if, if else	nested for, for	Unsure
F	19 - 21	Yes	5	4	for, nested for	if else, if	Yes
G	16 - 18	Maybe	1	2	if, if else	nested for, for	No
H	22 - 24	Yes	3	5	if, if else	nested for, for	Somewhat
I	19 - 21	Yes	4	4	if, if else	nested for, for	Yes
J	16 - 18	Yes	2	2	if, if else, nested for	for	Somewhat
K	19 - 21	No	3	1	if	if	Yes

Participant	Q1	Q2	Q3	Q4	Q5	Q6	Q7
L	16 - 18	No	1	1	if, if else	nested for, for	Somewhat
M	19 - 21	Maybe	4	2	if	for	No
N	19 - 21	Yes	4	4	if, if else	nested for, for	Yes
O	19 - 21	No	2	1	if else	for	Somewhat
P	19 - 21	Yes	3	3	if, if else	nested for, for	Yes
Q	16 - 18	Maybe	2	2	for, if, if else, nested for	if else, nested for, if, for	Somewhat
R	16 - 18	Yes	3	3	if, if else	nested for, for	Somewhat
S	16 - 18	No	3	2	for, nested for	if else, if	No
T	16 - 18	Yes	4	5	for, nested for	if else, if	Yes
U	19 - 21	Yes	2	1	if, if else	nested for, for	Yes

E.2 Posttest

Table E.2.1: Individual participant results from posttest questionnaire

Participant	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Q11
A	4	8	4	4	4	if, if else	do while, nested for, while	Maybe	Yes	Yes	Add Music
B	3	3	4	3	3	nested for, if, if else, for	nested for, if else, for	Maybe	Maybe	Yes	
C	2	8	1	4	4	nested for, if, if else, for	if, nested for, if else, for	Maybe	Maybe	Yes	You need to add a code, that forces the user to use for loop, if and etc. As midway through the game, I decided to test whether i can just copy a paste player.move("right") multiple time. This allowed to completes level quicker. But I only tried this for one level. Game was overall good, but could use more instruction to make it easier for people who do not know this.
D	4	6	3	4	4	if, if else	nested for, for	Maybe	Yes	Yes	
E	5	4	1	4	5	if, if else	nested for, for	Yes	Yes	Yes	

Participant	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Q11
F	4	7	3	5	5	if, if else	nested for, for	Yes	Yes	Somewhat	The tutorial was long and I didn't want to read multitudes of text. Using arrows and small messages that point to various parts of the UI instead of a big paragraph would have been appreciated (example: coolors.co). The IDE was pretty nice, Alt+upp/down arrow key to move lines of text was nice but the tab could have been adjusted to match the current scope. The game itself was pretty cool although I think having fixed lines of code that the player is not able to change would have added a level of difficulty, as it forces iterative over purely procedural programming. A tips section would have been cool in case there was a difficult programming puzzle, and an introduction to each function would have been better than needing to click "Help" to find out shooting an enemy is a thing. Overall pretty dope game.
G	4	8	3	3	4	if, if else	nested for, for	Yes	Yes	Somewhat	Help section was very useful.
H	2	5	5	2	2	if, if else	nested for, for	No	Maybe	Somewhat	Alright game

Participant	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Q11
I	5	8	2	3	4	if, if else	nested for, for	No	Maybe	Yes	Should force user to use for loops and if statements
J	4	7	4	4	4	if, if else	nested for, for	Yes	Yes	Somewhat	
K	2	3	5	2	2	if, if else	nested for, for	Maybe	Yes	Somewhat	It was very helpful to learn the iteration and conditional but it does get boring after a while.
L	4	3	4	4	3	if	if	Maybe	Yes	Somewhat	Good game, fun to play, really made me want to learn more about programming
M	1	1	5	1	1	if	for	No	Yes	No	What I was supposed to do was unclear. Though I was told to get the player to the red square, I couldn't see the red player until I clicked run. The second time I got to click run, it wouldn't work
N	3	4	4	4	3	if, if else	nested for, for	Maybe	Yes	Not very	
O	5	7	1	5	5	if, if else	nested for, for	Yes	Yes	Yes	Sometimes I had to restart because of the website not responding after clicking the 'run' button. Other than that the game seems both helpful and makes the player want to learn more about iterative and conditional statements!

Participant	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Q11
P	4	8	3	3	3	if else	for	Maybe	Maybe	Yes	Perhaps the text could have been more concise during the step to step guide at the beginning of the game.
Q	4	5	5	4	3	nested for, if	nested for, if else	Yes	Yes	Somewhat	was hard but has made me interested
R	2	2	4	4	4	if, if else	nested for, for	No	Yes	Yes	
S	5	7	3	3	3	if, if else	nested for, for	Yes	Maybe	Not very	
T	3	5	2	4	2	nested for, if, if else, for	if, nested for, if else, for	Maybe	Yes	Not very	
U	4	8	4	3	5	if, if else	nested for, for	No	Yes	Somewhat	

Appendix F

Code

The source code for the game contains the following six vital JavaScript files. These define the map layout, the game logic and and draw functions:

1. Canvas.js - Defines the the layout of entire website. This includes map, console and button positions.

2. createLevel.js - Defines the map co-ordinate system (2D array), character and enemy position, and the text within the console at the start of a level.

3. Entities.js - Defines the character, enemy and bullets, and their corresponding methods. This contains main game logic, such as checking collisions and drawing entities.

4. ifLogic.js - Defines the logic for wallPosition and enemyPosition functions.

5. Main.js - Contains main update loop. Also contains method of executing the code within the console.

6. Map.js - Defines the graphical layout of the game. This includes the map walls and exit tile.

The entire source code can be found on GitHub, using the following URL:

https://github.com/amarnathkakkar/programmersdungeon