The Development of a Serious Game to Teach Aristotle's Syllogisms

James Treasure

BSc (Hons) Computer Science

University of Bath

May 2017

The Development of a Serious Game to Teach Aristotle's Syllogisms

submitted by James Treasure

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 support 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.

Signed:

I

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.

Abstract

Research shows that the learning process can be enhanced through the use of serious, educational games. The focus of this dissertation was to build a web game that taught the basics of mathematical logic including Aristotle's syllogisms and set theory. This project was also concerned with designing a user experience and user interface necessary to create an immersive game. Whilst the technology used was not of primary importance some investigation was carried out on the tools available when creating web games.

Contents

1. Introduction

The term 'serious game' is broadly used to describe games created to teach people about a variety of topics in industries such as education, military and engineering. Games have been used as a way to educate since the beginning of the 1900s with paper-based games seeing a popularity surge in the 1960s [Rice, 2007]. The prevalence of computers in today's world has breathed new life into this industry and there is again focus on the benefits of serious games.

Aristotle was a philosopher, mathematician and student of Plato's who lived during the 4th century BCE in Ancient Greece. One area of Aristotle's work was on syllogisms, one of the first examples of formal logic.

The aim of this project was to bring these two together and develop a web based serious game that can be used to teach people the basics of logic, set theory and syllogisms. The game aimed to be as accessible as possible with no pre-requisite knowledge of set theory. The aim was to introduce new concepts slowly and with adequate explanations in order to make create an effective tool for learning.

This involved exploring the methods currently used to teach syllogisms, their shortcomings and how these methods can be improved upon. The reasons that make serious games so effective such as their accessibility, and motivating and engaging nature were also looked at. By combining all of this information, a game was created to try and improve the current state of syllogism education.

Venn diagrams were chosen as the representation for syllogisms with dragging, dropping, and highlighting segments as the core mechanics behind the game. This option was chosen as it was both the simplest way to depict syllogisms as well as one that would be the most familiar to users.

Once developed, an evaluation was carried out on a group of users to determine the effectiveness of the game's ability to teach about syllogisms. The results showed that participants with more mathematical experience performed better at the game. However, nearly all participants who played the game reported that they did successfully learn something and found the experience enjoyable and engaging.

1.1   Objectives

- Investigate the history of Aristotle's syllogisms and how they are currently taught

- Investigate why serious games are effective tools for learning

- Gather a list of requirements required in order to develop a successful serious game

- Design and develop the serious game

- Evaluate the effectiveness and usability of the serious game

- Discuss future developments for the project

2. Literature Review

2.1 Syllogisms

2.1.1 History

Aristotle's work on formal logic was the earliest known study of the topic, which remained at the forefront of academia until the 19th century following Gotlobb Frege's work on first order logic. As syllogisms were so prominent for such a long period of time they have had huge cultural significance on logic in the Western world [Smith, 2016]. Up until the 12th century, medieval logicians only had access to a small portion of Aristotle's work with the notable exclusion being Prior Analytics, which contained his work on syllogisms. This period of time is known as logica vetus, or old logic, and it was not until the 12th century when Prior Analytics resurfaced in the west that the logica nova, or new logic, began. Immanuel Kant, a prominent 18th century philosopher, went as far as to describe Prior Analytics as "a closed and completed body of doctrine" demonstrating just how highly Aristotle's work on syllogisms was thought of.

2.1.2 What are syllogisms

Aristotle defined his syllogisms as *a discourse in which, certain things having been supposed, something different from the things supposed results of necessity because these things are so.* Put more simply syllogisms are a type of deductive reasoning that, when used on a logical argument, allows a conclusion to be drawn. Aristotle's focus was on categorical syllogisms, essentially a logical argument that contains three categorical propositions. These three categorical propositions in turn are made up of two premises and a conclusion. Each categorical premise is made up of categorical terms, of which each is used twice in the syllogism as a whole. Each categorical premise can be described as a sentence that connects a predicate and a subject by a verb.

$$All\ M\ are\ P \rightarrow \text{major premise}$$
$$All\ S\ are\ M \rightarrow \text{minor premise}$$
$$All\ S\ are\ P \rightarrow \text{conclusion}$$

Syllogisms are capable of being applied to any logical argument but to allow them to be compared to each other, they must be represented in a standard form. A categorical syllogism that follows standard form is always structured in the same way, first the major premise, then the minor premise followed by the conclusion.

As created by medieval logicians studying Aristotle's work, the mood and figure of a syllogism provides a notation to represent all the logically unique variations that can occur. The mood refers to the order in which the categorical propositions appear in the syllogism and is represented by four different classes.

- **A** - All A is B → universal affirmative proposition

- **I** - Some A is B → particular affirmative proposition

- **E** - No A is B → universal negative proposition

- **O** - Some A is not B → particular negative proposition

Syllogisms also have a figure that describes the placement of the two middle terms.

	Figure 1	Figure 2	Figure 3	Figure 4
Major	M-P	P-M	M-P	P-M
Minor	S-M	S-M	M-S	M-S

By combining all the different possibilities of mood and figure there are 256 logically unique syllogisms, eg., AOO-2.

$$All\ P\ is\ M$$
$$Some\ S\ is\ not\ M$$
$$Some\ S\ is\ not\ P$$

Only 24 syllogisms out of the possible 256 are logically valid, and of these the existential fallacy is broken by 15 of them.

Existential Fallacy

The existential fallacy occurs whenever an argument is invalid because the premises lack existential import. Existential import is an attribute of a categorical proposition that implies the existence of the subject [Hurley, 2005]. There are two schools of thought on the existential fallacy with regards to syllogism, Aristotelian and Boolean.

Traditionally, as laid out by Aristotle, the existence of a subject is assumed when making universal statements. I and O propositions are said to have existential import and in Aristotelian logic I and O propositions must follow from A and E propositions due to subalternation. Subalternation is an inference made between A and I or E and O propositions. It says that if A is true it can be inferred I is true, and similarly with E and O. This leads to A and E propositions also having existential import as a proposition with existential import cannot be derived from a proposition without existential import. By considering the syllogism in 2.1 we can say that it does not commit the existential fallacy as dogs do exist whereas in 2.2 the existential fallacy is committed as dragons do not exist.

However, as modern day logicians and mathematicians often reason about empty sets or imaginary objects the Aristotelian approach to the existential fallacy was problematic. George Boole, a 19th century philosopher, built upon Aristotle's work to create his own Boolean syllogisms to address this issue. He stated that universal propositions do not have existential import, thus allowing empty sets [Hill, 2016]. In Boolean syllogisms both figure 2.1 and 2.2 commit the existential fallacy.

All dogs are mammals	*All gorillas are lizards*
All mammals are animals	*All dragons are lizards*
Some dogs are animals	*Some dragons are not gorillas*
Figure 2.1	Figure 2.2

The general rule to reason if a syllogism commits the existential fallacy is that if both premises are universal then the conclusion cannot be particular. In Boolean logic if this rule is broken, then it is a fallacy but in Aristotelian logic the rule must be broken and the critical term must also not exist.

2.1.3   Current teaching

Syllogisms are a rather niche topic, currently featuring almost exclusively on philosophy courses, typically as part of units on formal logic. However, Venn diagrams and basic set theory are taught to children in school. So whilst people are not directly taught about syllogisms, they are already familiar with some of the most common ways to represent them. There are a few different ways that are currently used to depict syllogisms when teaching, with most courses using a diagrammatic, set theory and sentential notations.

It is common to initially introduce syllogisms in sentential form, using real world examples as opposed to a more abstract representation. The advantage of this is it becomes far easier for the learner to visualise the propositions. For example, visualising the statement "all dogs are animals" is simpler than "all A are B" despite stating the same idea.

Representing syllogisms this way is not the most effective as they can be logically quite complex. As [Larkin and Simon, 1987] explains, sentential form requires inferences to be made, and then those inferences to be held in memory. Working through the syllogism, for example to decide if it is valid, means that these inferences must be continually remembered. As shown by Johnson-Laird [1980], sentential forms of syllogisms are more difficult to understand and may be mentally translated into diagrammatic representations internally anyway.

Venn Diagrams

Venn diagrams are the most commonly used diagrammatic representation of syllogisms. The advantage over other diagrammatic forms is that most people have been taught them at a young age and are therefore very familiar with them.

In this representation each circle is a term, which is then filled in to represent the empty set or marked with a X to show at least one member. To achieve the final diagram the two previous diagrams are combined which can then be used to check the validity of the syllogism. It can be seen from table 2.3 that even complex syllogisms are able to be represented with relatively little complexity using Venn diagrams which contributes to their popularity.

A	I	E	O
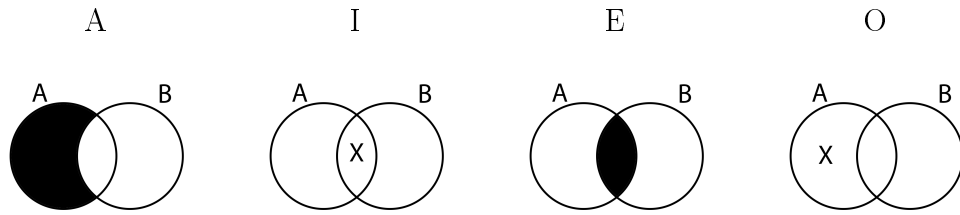
Table 2.1: Premises represented by Venn Diagrams

All Men Are Mortal

All Greeks Are Men

All Greeks Are Mortal

Table 2.2: AAA-1 Syllogism

No lizards are mammals

All lizards are reptiles

Some reptiles are not mammals
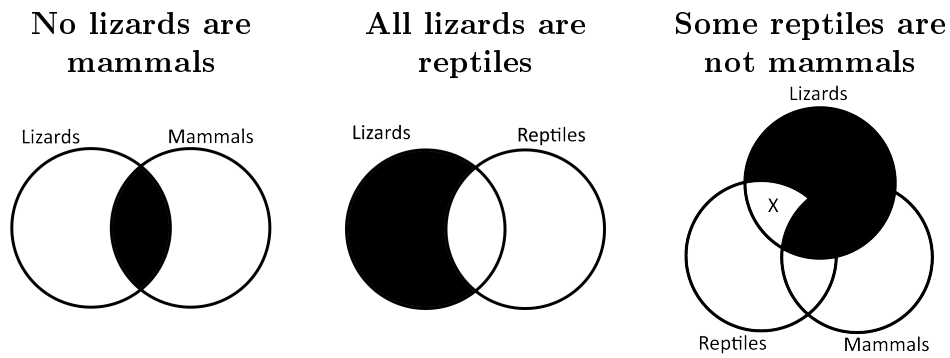
Table 2.3: EAO-3 Syllogism

Euler Circles

Euler circles are another popular way in which syllogisms can be represented. These are a very natural way of representing sets, with Erickson [1978] putting forward the theory that when people are thinking about syllogisms that they are in fact being mentally thought of as Euler circles. As table 2.4 shows, when used to represent the individual premises Euler circles can be very intuitive.

| A | I | E | O |
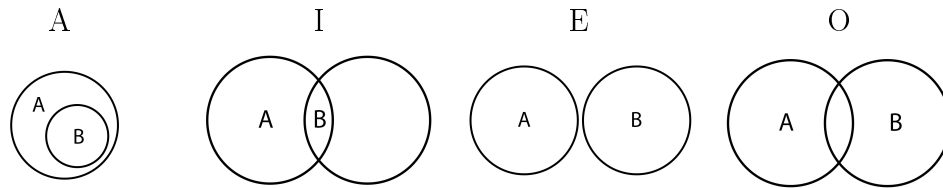
Table 2.4: Premises represented by Euler Circles

| All Men Are Mortal | All Greeks Are Men | All Greeks Are Mortal |
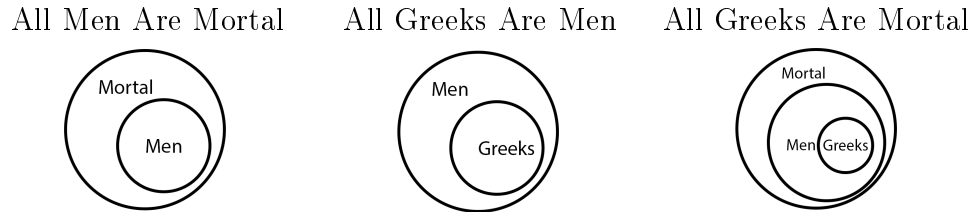
Table 2.5: Premises represented by Euler Circles

Euler circles also concisely represents simpler syllogisms as table 2.5 shows. However, the problem with this representation arises when more complex syllogisms are constructed. When there are multiple ways to represent the conclusion the result needs more than one diagram to depict all of the possibilities as seen in table 2.6 . It is at this point that the intuitive benefits are quickly outweighed by the added overhead of managing multiple conclusive diagrams.

No homework is fun   Some reading is homework   Some reading is not fun
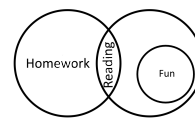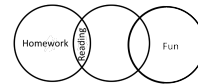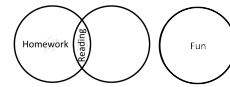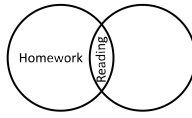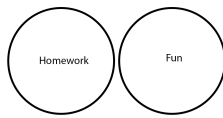
Table 2.6: EIO Syllogism

Linear Diagrams

Linear diagrams are an alternative system that was devised by Englebretsen [1991]. As the name suggests they are linear instead of planar like Venn and Euler diagrams. The reasoning behind this was that Englebretsen said it allowed linear diagrams to represent 4 terms, unlike their planar counterparts. Elements are represented by points which are then connected by lines to show sets.
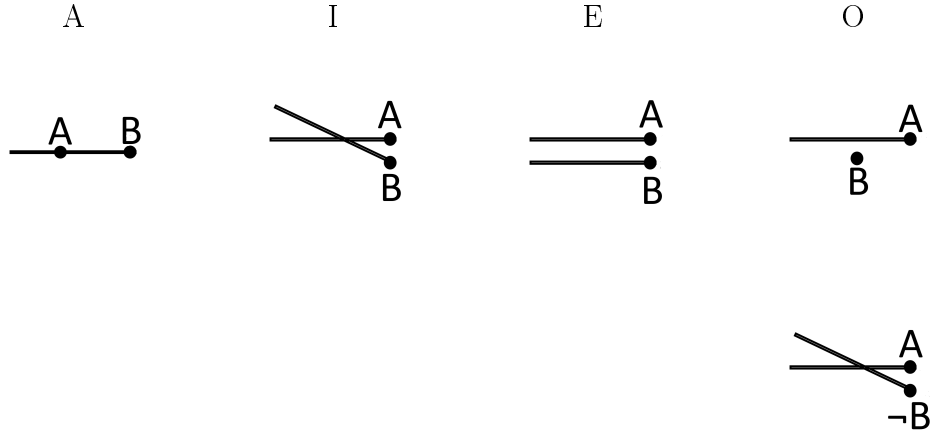
Table 2.7: Premises represented by Linear Diagrams

However, as shown by Lemon et al. [1998], linear diagrams do not successfully manage to circumvent the same limitations that planar figures are susceptible to. Lemon et al. [1998] goes as far to say that linear diagrams should only be used for trivial objects and relations and that they are unsuitable for more complex sets.

Categorical Pattern Diagrams

As introduced by Cheng [2012], Categorical Pattern Diagrams offer a relatively new approach with regards to representing syllogisms. The aim behind them is to simplify the process of checking validity, better represent syllogisms with more than 2 premises and generally make the diagrams easier to infer from.

The issue with this form of representation is the notation used is not intuitive. Categorical Pattern Diagrams require the a whole new notation to be learnt before they can begin to reason about the syllogism that it is representing.
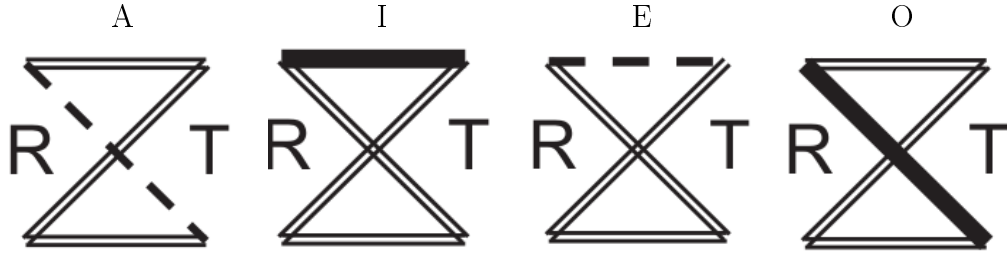
Table 2.8: Premises represented by Categorical Pattern Diagrams

2.1.4 Additional Logic Systems

So far the only formal logic system looked at has been syllogisms, but many more exist. There are number of systems that would be ideal to include in the syllogism game to allow the player to gain a broader understanding of formal logic.

Polysyllogisms

Polysyllogisms, unlike standard syllogisms, can contain more than 3 propositions. This poses a problem for many graphical representations, with traditional circular Venn diagrams being limited to 3 sets. As figure 2.3 and 2.4 show it is possible to represent more than 3 sets using different shapes but as the diagrams become exponentially more complex as sets are added it is quickly becomes impractical to use them.

As reasoned by Cheng [2014], linear and Euler diagrams also struggle with this problem. As both use spatial-overlap it means the diagrams can only represent a series of universal affirmative propositions as further propositions are nested in. As can be seen in table 2.7 and 2.6 there are already multiple ways to represent syllogisms with 3 terms so it is easy to see how it would quickly become unmanageable as the number of propositions increases.
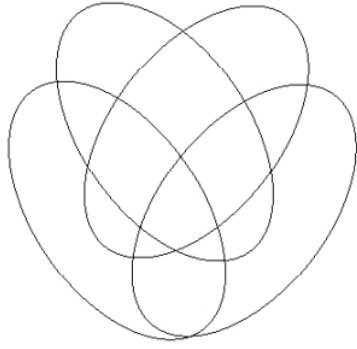
Figure 2.3: 4 Set Venn Diagram
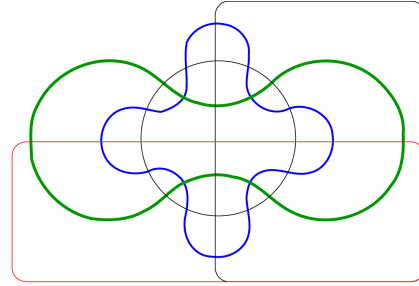
Figure 2.4: 5 Set Venn Diagram

Modal Logic

Modal logic is an extension of classical proposition logic that uses the modal operators *it is possible that* and *it is necessary that* to describe a way in which the rest of the proposition can be true [Zalta, 1988]. Modal logic allows statements to be expressed that would otherwise not be possible in (propositional) classic logic as it is not possible to convey the prospect of events happening. The following are examples of modal propositions:

It is possible that it will snow tomorrow
It is necessary that it is either snowing here now or it is not snowing here now.

The necessity operator is denoted by $\Box$, whilst the possibility operator is represented by $\Diamond$.

Temporal Logic

Temporal logic is a formal system for specifying and reasoning about propositions with regards to time. It is often used in the formal verification of software systems to interpret concurrent events [Lamport, 1983]. Temporal logic can be split into two subsections, linear and branching. Linear temporal logic represents time as a set of paths with each part being a sequence of moments. From any given moment there is only one possible future moment, a unique successor. Branching linear logic represents time

as a tree with the root as the present and the branches being the future.
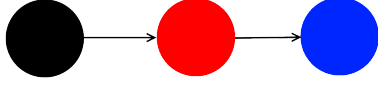
Figure 2.5: Linear Temporal Logic
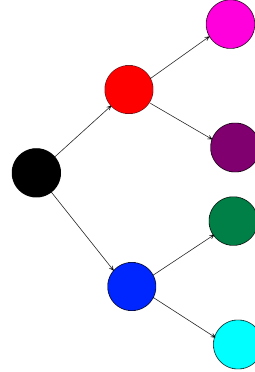
Figure 2.6: Branching Temporal Logic

Set Theory

Set theory is the mathematical theory of well-defined collections known as sets. These sets are made up of objects called elements. Given a set S and an element x it is said $x \in S$ if x is an element of S and conversely $x \notin S$ if x is not an element of S [Bagaria, 2016].

Diagrammatically they appear very similar to syllogisms as both utilise Venn diagrams so it would naturally follow to expand a game featuring syllogisms to encompass set theory. This is especially true when it is considered how syllogisms can be represented using set theory notation as in table 2.9.

Syllogism	Set Notation	Sentential Representation
AaB	$B \subseteq A$	All B is A
AiB	$B \cap A \neq \emptyset$	Some B is A
AeB	$B \cap A = \emptyset$	No B is A
AoB	$B \cap A \nsubseteq B$	Some B is not A

Table 2.9: Set Theory Notation for Syllogisms

13

2.2   Serious Games

2.2.1   Introduction

Serious Games are a movement within the game industry comprised of software developers and researchers who are using games to educate. Whilst there is some debate around what exactly construes a serious game, Michael and Chen [2005] define one as any game where entertainment is not the primary purpose, but where instead the focus is on education. They aim to tackle a growing problem whereby educational techniques have not adapted to the needs of the current generation. As Lim [2008] discusses, learning in the classroom has been driven by the national curriculum, with a massive focus on standard and grades. This goes against the belief that to attract the attention of students the focus should in fact be on creating an engaging learning environment. In contrast with previous generations, the younger generation of today have grown up in a world consumed in technology. Oblinger et al. [2005] described them as the *Net Generation*, the generation who always need to be connected, require immediate feedback and crave social interaction. The research of Prensky [2001] explains how this vast amount of technology now experienced in everyday life has led to these newer generations having their minds rewired. These cognitive changes have caused a different set of educational preferences when compared to previous generations, with teaching methods not evolving in order to take this into account.

2.2.2   Why They Are Good For Education

Accessibility

One of the benefits of using games to teach is their accessibility in comparison to other methods. In the United Kingdom access to the internet has doubled over the past 10 years, with 82% of adults now using the internet on a daily basis [Office For National Statistics, 2016]. With such a huge number of people using the internet it means that games hosted on the web have an extremely large potential audience. This broad audience coupled with the widespread appeal of video games allows the crossing of demographic boundaries such as age, gender and educational status [Griffiths, 2002]. Serious Games also have a lower barrier to entry

than other educational techniques by allowing a much more "pick up and play" approach to learning with less reliance on prerequisite background knowledge. In the case of syllogisms this would be achieved by using graphical representations to replace the set theory notations. By teaching the concepts first, set theory notation could then be introduced at a more appropriate time in the learning process when the player had a greater understanding of the topic.

Motivation

As discussed by Malone [1981] games can be intrinsically motivating, such that there are no external factors as to why a person is playing other than for their own enjoyment. This is in contrast to typical classroom learning that is usually more extrinsically motivated through grades, exam results and certificates. As Csikszentmihalyi et al. [1997] says, the school attendance would see a sharp drop if extrinsic rewards were removed. Of the two, intrinsic motivation is more resilient than extrinsic motivation and learning that takes place as a result of intrinsic motivation is of higher quality and longer lasting [Kawachi, 2003].

However, it is important to remember that not all games are by definition intrinsically motivating, but need to be structured correctly to achieve this. Malone included challenge, fantasy and curiosity as the three cornerstones of intrinsically motivating serious games. In order to create a challenging game, it should include uncertain goals that are not too easy for the players to complete. Ensuring a game is challenging is crucial as demonstrated by Malone's work surveying children's opinion on 25 different computer games. The results conclusively showed that the most popular games all shared one thing in common; they all contained a defined goal. Similarly in a study carried out by Abuhamdeh and Csikszentmihalyi [2012] on chess players, it was found that the more challenging the game the greater the player's enjoyment. This is important to bear in mind with this project

A fantasy environment in a game is said to "evoke mental images of phyical or social situations not actually present". Malone divides these fantasies into intrinsic and extrinsic, with intrinsic being more interesting and instructional resulting in a greater cognitive effect on learning. Lepper [1988] explains how if learning takes place away from the functionality of the knowledge then this decontextualisation results in the learner becoming demotivated. This can be tackled by using real world examples demonstrating how the concepts are applied. For syllogisms this can just be as simple as using actual examples

that exist as opposed to using letters to represent sets. However, as this is not always feasible, Lepper proposed inserting the content of the topic into a fantasy context. Curiosity, like challenge, is stimulated from an optimal level of complexity. Curiosity is also broken down into two categories; sensory and cognitive. Sensory curiosity, as the name suggests, uses lights, sounds and graphics to gain the attention of the participant. Cognitive curiosity is achieved by making the participant doubt their existing knowledge, as people desire to have "good form" in their cognitive structure. By using this taxonomy when developing a serious game, it is clear that this could be used to engage people who otherwise might not have been interested in learning.

Feedback

Feedback is information regarding a person's understanding of a concept provided with the intention of improving the person's ability. Feedback is a crucial part of the education process and significantly influences the effectiveness of learning.

As stated by Hattie and Timperley [2007], "feedback is one of the most powerful influences on learning and achievement". A further study explored the most effective medium to provide feedback. Computer-assisted, audio and video feedback were found to be most effective forms with the more traditional approach of praise, punishment and providing extrinsic rewards being the worst [Hattie and Timperley, 2007].

One of the main advantages of using games for education is the speed at which feedback can be delivered to the learner which is especially important to the *Net Generation*. The term *fast feedback* was coined by Lumsden and Scott [1988] to demonstrate how this timely feedback removed any uncertainty about progress and allowed any corrective action to be taken during the learning process. However, fast feedback is not a silver bullet and as such the speed at which feedback is delivered should depend on the difficulty of the task. The more complex the task, the greater the benefit of delaying the feedback to allow the information to be processed Clariana et al. [2000].

Feedback during the learning process is known as formative assessments, with summative assessments usually being carried out at the end. As explained by Irons [2007], formative assessments can be beneficial as they enhance the learning process by creating a positive environment which results in greater motivation for learner.

Serious Games lend themselves extremely well to delivering formative feedback to players through in-game features such as progress bars, score count and countdown timers. Feedback can also be provided to the player as they make mistakes allowing the player to engage with the game and keep on track to completing their goals. This can also help the player from becoming frustrated and unable to progress.

2.2.3   Existing Serious Games

Serious Games can be split into two main categories, mini-games and complex games. Mini Games tend to be far shorter, often only taking less than an hour to complete. The focus is aimed at a very specific topic within a subject and is usually browser based. Progression tends to be added through multiple levels, all with similar design but with increasing difficulty. This is the far more common type of Serious Game as they are far cheaper to develop.

In contrast, complex games can take potentially hundreds of hours to complete and cover a very broad topic. The game may contain multiple skills, paths to follow and complex goals.

Serious games do not just apply to education, but are used by a wide variety of different industries. The United States military created America's Army to aid in their recruitment and Microsoft's Flight Simulator is used by pilots in training [Homan and Williams, 1998].

MyMaths

MyMaths is a web based service that is used by schools in over 70 countries, with over 4.5 million users. It is marketed as an online learning platform that provides lessons and homework to school children. It allows them to complete all of this work online and is then automatically marked. As figure 2.7 shows, MyMaths makes use of fast feedback to provide the player with the results as soon as they have completed the task. However, MyMaths does not provide an immersive or particularly enjoyable experience for players. Lee and Johnston-Wilder [2013] found that while students said they enjoyed using computers to learn, they did not want to use MyMaths. This could be due to that fact that MyMaths is very one dimensional, merely presenting questions in a quiz like format. This does not engage with the player, incite any emotive feelings or provide any fantasy environment. It would more
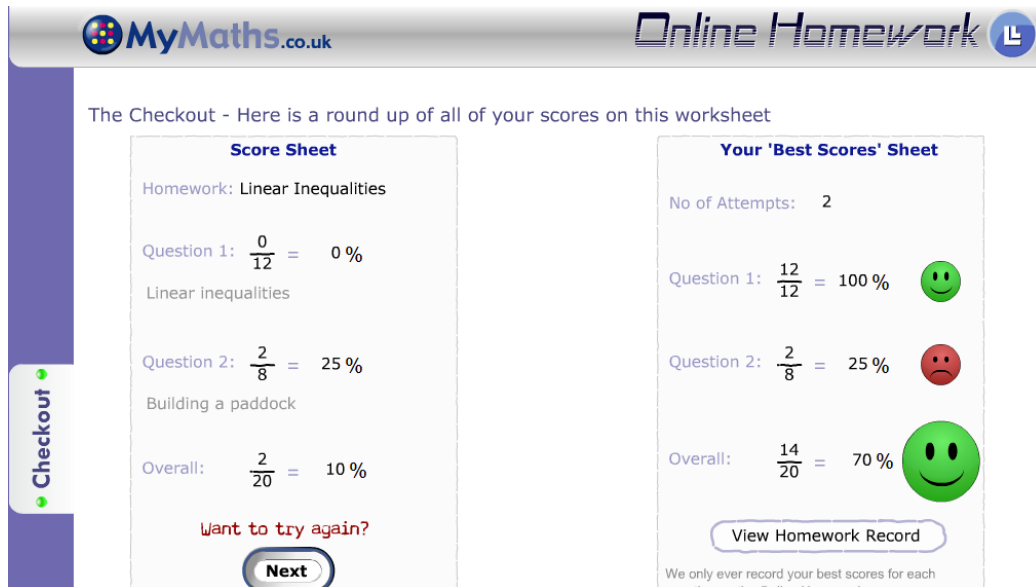
Figure 2.7: A homework exercise in MyMaths

enjoyable if the technologies available were used to make it more interactive.

BBC Bitesize

BBC Bitesize is a website maintained by the BBC (British Broadcasting Corporation) that provides resources for children in the United Kingdom on a variety of school subjects. Maths is one of their main focuses with both static content and a number of educational games. Bitesize has a large number of games available, with 11 Key Stage 1 games, 25 Key Stage 2 games and 4 Key Stage 3 games. Key Stages are a grouping of school years used in the United Kingdom, with Key Stage 1 being for ages 6-7 up to Key Stage 5 being for ages 17-18.

With such a large number of games it allows each one to be very focussed on the topic it is teaching. This is also a disadvantage as for simpler games there is only so much that can be done within a certain topic before the user will start to get bored. As a further criticism, the vast majority of games remain at the same difficulty level throughout and fail to increase the challenge as the player becomes more proficient. This lack of challenge will lead to players losing interest.

From a technical perspective all of BBC Bitesize games are built using Adobe

18
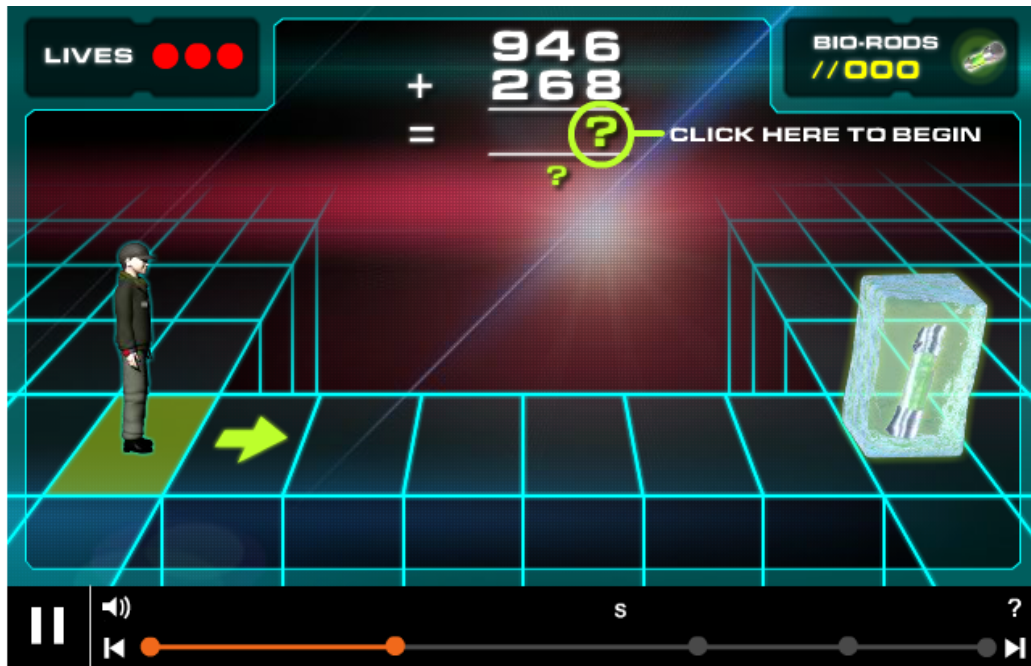
Figure 2.8: Addition and Subtraction game on BBC Bitesize

Flash. This is a serious accessibility problem with major browsers phasing out support for this ageing technology. Google's Android operating system and Apple's iOS do not support Adobe Flash and as these games are targeted at school children and with the popularity of tablets and smartphones it possible that they are severely limiting how many people can play their games.

From the above examples a trend can be seen that shows that whilst there are an extensive number of educational maths games available on the internet they are aimed almost exclusively at a younger audience. This is perhaps because of the connotations that games are predominantly for children but as discussed previously the benefits of using games for education apply to all ages. This means that most games are based on rather simple, concrete mathematical ideas. There are very few games out there that tackle harder, more abstract concepts such as syllogisms. Also it could be said that the majority of existing games are used to reinforce concepts that have been taught in schools by teachers. The games are aimed at practising those concepts by repeating the ideas over and over. This in turn means that the games themselves are not responsible for actually teaching the core ideas as that is not their purpose. As syllogisms will almost certainly be completely new to anyone playing the game this is something that will have to be focussed on.

3. Requirements

3.1 Introduction

As stated previously, the main aim of this project is to build an educational game to explain formal logic, mainly through the use of Aristotle's syllogisms. The following requirements are used as a guide for the project during development and to ensure the project remains on track. They are separated into functional and non-functional requirements. Functional requirements are concerned with how the game should behave in particular situations with relation to its functionality. Non-functional requirements place constraints on how the game's functions will complete their tasks, often applying not just to individual functions but the system as a whole [Sommerville, 2006].

3.2 Sources

Literature Review

The main source of requirements is the work in the literature review. Areas such as accessibility, motivation and feedback were extensively researched and the findings have been used to influence these requirements. The discussion of existing serious games and in particular their advantages and disadvantages also provide a source to draw upon.

Project Supervisor

Dr. Willem Heijltjes, the supervisor for this project, has experience with overseeing the development of serious games. His opinion was given many times during the course of the project with regards to requirements and the content that should be included.

3.3 Functional

1 **The game must scale dynamically to the screen resolution**
It is important when supporting devices of different sizes and screen
resolutions that the experience remains consistent throughout. In the
context of the game this relates to the need for the user interface to
scale appropriately. The chosen font and font size must be readable
across an array of devices and there must be no distortion of images.

2 **The game must be playable using only the mouse**
As the game is a simple puzzle game there is no benefit to allowing
the user to play with the keyboard in addition to the mouse. By
allowing only the mouse to be used to play the game it will also make
it compatible with devices that do not have a keyboard.

3 **The game should be playable on tablets and mobile phones**
By allowing the user to interact with the game using only the mouse
(finger if on a touchscreen device) and requiring no use of the
keyboard it will allow the game to be played across all devices. This
aids accessibility as players may not have access to a traditional PC
or may want to use a touchscreen device.

4 **The system shall track time taken to complete each level and
moves made**
In order to analyse how effective the game was there will need to be a
number of data points recorded. The length of time each level took to
complete, along with the number of moves made will be saved as this
allows conclusions to be drawn about the performance of the user. By
combining this data with initial questionnaire data it will be possible
to analyse the effectiveness of the game on different groups of people.

5 **The game must have multiple levels**
This allows the game to progress in difficulty, allowing the player to
begin with the basics and work towards learning more advanced aspects
syllogisms and logic.

6 **The system shall easily allow new levels to be added**
With there being 256 syllogisms, although just 24 logically valid, it is
important to develop a way to easily add new levels to the game. If the
mechanism used to add new levels is simple enough it could be possible
for users to create their own levels adding another dimension to the
game.

7 **The game must successfully recognise when the player has completed a level**
The game must be able to accept any valid answer from the user. By not recognising all potential end game scenarios it could confuse the user into thinking they are incorrect which would lead to reduced engagement with the game.

8 **The game must feature a tutorial when introducing new concepts**
A tutorial is crucial to explain the premise and controls of the game. With the focus being on the educational aspect it is important to make interacting with the game and getting to grips with the controls as simple and painless as possible so the focus can be on learning.

9 **The user shall be able to go backwards and forwards through the tutorials**
The tutorials will be a vital aspect of the game, with nearly all the new content and information being delivered through them. It is therefore important that the user can navigate backwards and forwards through the instructions in case they need to go over a certain piece of information for a second time.

10 **The user shall be able to undo and redo actions**
An undo and redo function is an important feature for a good user experience, especially so in systems involving manipulation of graphics as errors are so easily made. As most desktop and many web applications support this functionality, it would detract the user from the experience if this was not an option. By supporting these functions it allows the user to explore the interface with the freedom to experiment without fear of making an irreversible mistake.

11 **The game should provide visual hints to assist the user**
One of the biggest advantages of using games for education, as mentioned in the literature review, is their ability to provide feedback to the user whilst playing. Therefore it is important to utilise this by providing hints, positive reinforcement, and help to the user through on screen graphics during the game.

12 **The game shall have a leaderboard ranking users**
A leaderboard showing how the user performed compared to other people who played the game will increase engagements levels. The leaderboard will rank players based on time taken to complete the game as well as acting as a summative assessment informing the user

how well they did.

13 **The user must be able to save the game.**
There should be the functionality for the user to save their place in the game so they can return to it at a later date. This could be useful if the user is interrupted whilst playing the game so they can return to it and carry on playing from where they left off.

3.4   Non-functional

1 **It must be possible to play the game on all major browsers**
This requirement is imperative as accessibility is one of the reasons behind building a web based game. It is important to have a consistent experience across all browsers to not detract from the experience of playing the game. The browsers targeted will be the current list of most used browsers as maintained by W3Schools [W3Schools, 2017]. As most minor browsers are built using the same underlying technology as one of the major ones, provided it works on all of the most popular ones there is a very high chance it will work on the minor ones.

2 **The game must have an elegant user interface**
It is important to not crowd the users with an excessive number of buttons and options in the game. As no keyboard controls will be used, the challenge will be to provide a way to operate all controls of the game efficiently.

3 **The game must appear smooth to the user**
When creating applications that a user interacts with it is important to consider the delays taken between the user interacting with the system to the result being shown on screen. For the game to appear to run without delay, it would ideally react to user input within 100ms. Nielsen [1994] explains a delay of less than 100ms will give the user the feeling of the system reacting instantaneously. A delay of longer than 100ms can lead to the user losing their focus on the task and can require feedback from the system informing the user of the delay. User experience guidelines published by Google delve further into this area and say that 16ms is the maximum response time for animations [Kearney, 2017]. As this system is a game, that will heavily feature animations, this is important to consider.

4 **There must be an initial prototype of the game**

23

Following an agile methodology when carrying out software development requires the application to be built incrementally. A minimum viable product should be built and then features continually added. By using this approach when developing the game it will mean that there will be an initial prototype that can be used to gather user feedback. Supervisor feedback can also be given to steer the project in the right direction and anything that needs to be added can be picked up at the development phase before full user testing takes place.

5 **The navigation toolbar must remain consistent throughout the game**
It is important that the navigation bar does not change location throughout the game as it could confuse the user. It is also important that all the options available are present for every level. For example, if some levels do not have an undo option it could lead to confusion for the user.

6 **The game must motivate the user to learn about syllogisms**
The game must engage with the user and maintain their attention. It is important to create an environment where the user wants to be playing the game so that the intrinsic desire to complete the levels is strong. It is therefore important to balance the educational aspect with enjoyment and getting the difficulty of the levels correct as to not demotivate the user.

4. Design

4.1 Game Design

The game is styled in a drag and drop puzzle format that requires the player to move words, numbers and pictures into Venn diagrams. Each level has a statement or fact that must be represented diagrammatically through the correct placement of objects in the Venn diagrams. Representing syllogisms using diagrams was the plan from the outset of this project, which was further reinforced by the comments of Larkin and Simon [1987] and Johnson-Laird [1980] on the shortfalls on sentential representations. Venn diagrams were chosen over the other diagrammatic options discussed in the literature review as they are both universally understood and provide the most concise way to reason about syllogisms.

By having users drag, drop and fill in sections of the circles it creates an interactive experience. This leads to the users experimenting, exploring the interface and having the freedom to make mistakes when trying to complete the levels.

Tutorials

There will be a number of tutorials throughout the game and so there must be a clear and consistent structure to them. By making it obvious to the player when a tutorial is happening the player will be reminded to pay close attention. The structure of the tutorials will involve instructions and information being displayed on screen with relevant areas highlighted to the user. Forwards and backwards buttons are present at either side of the screen for the user to navigate their way through the tutorials.

Level Design

It is very important to design the levels with the correct level of difficulty. If the game starts off too hard it would demotivate the player and stop them from wanting to play. Conversely, if the game is too easy and does

not introduce harder concepts as the levels progressed there would be no challenge to keep the player interested.

The general idea is to initially have very simple levels that introduce the mechanisms and controls for how the game works. These simple levels can then also act as a refresher on Venn diagrams and other simpler set theory ideas.

After this there will be a number of syllogism levels. The first one will be structured with a very tutorial heavy focus. Every step of process from dragging the words into the correct circles to completing the syllogism line by line will be aided by the tutorial system guiding the user through the process. After the tutorial the syllogisms will get progressively harder with less assistance from the system.

As an addition to the game, once the syllogism levels are completed there will be a few levels that cover set theory. They aim to teach the basics such as the union and intersection of sets as well as the distributive and associative laws.

4.2   User Interface

The goal of the user interface was to be as simple as it possibly could be whilst still allowing all the required functionality to be added. The approach taken with the design was to keep it as clean and minimalist as possible so not distract the player with unnecessary clutter. The relatively small amount of on-screen items also was a benefit when ensuring the game worked well on smaller resolution devices where screen real estate is at a premium. UI mock ups were created that could be shown to users for feedback and acted as a starting point once development began.
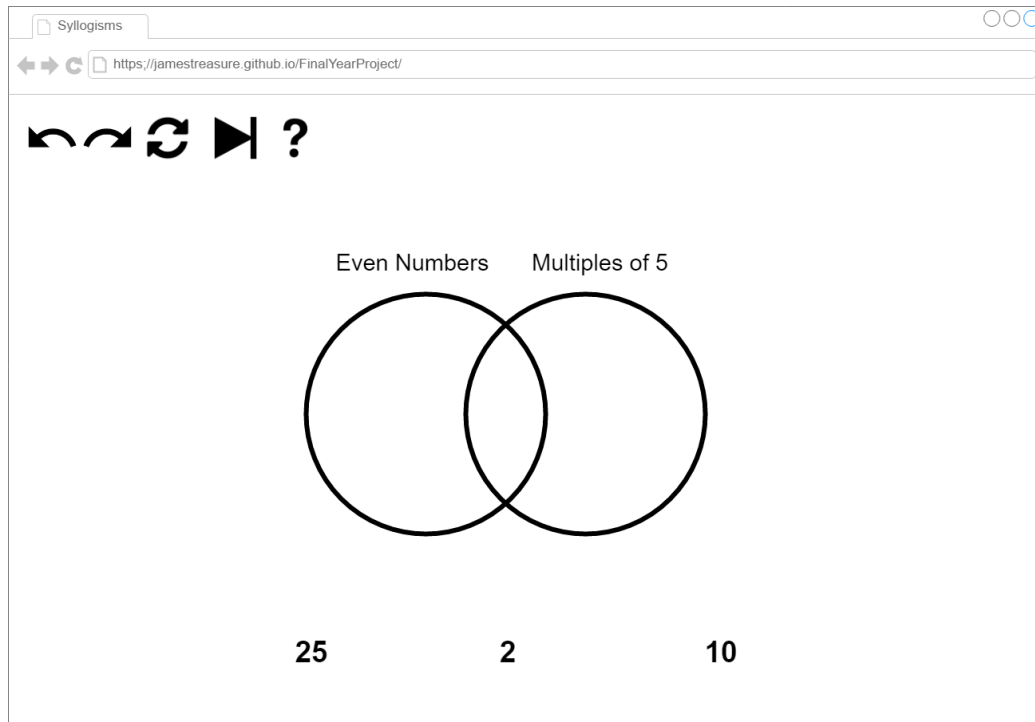
Figure 4.1: UI Mockup

4.3   Control Bar

There are a number of on screen controls that need to be present at all times. A control bar that remains consistent both in its placement and options available throughout the game will stop the user from feeling confused and allow them to be confident when interacting with it. The icons chosen were important as there would be no labelling of them so their functionality would have to be shown purely through how they looked. The icons chosen are based on Google's Material Design icons. The choice of icons will hopefully be familiar enough to users that their functionality is obvious.
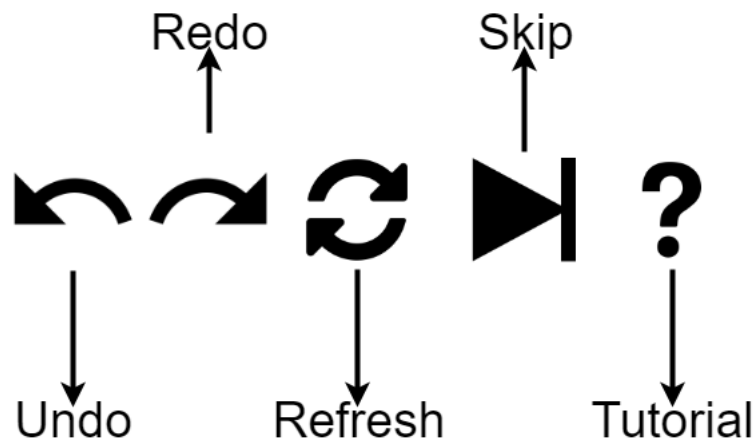
Figure 4.2: Control Bar

4.4 Technologies

Some technologies are forced upon the developer when creating web applications but there are some areas where there is a selection. The following discusses what technologies will be used in order to create the game and a discussion on the potential options available.

4.4.1 HTML

HTML (Hyper Text Markup Language) is a markup language that describes the structure of web pages. Websites created with HTML are then able to be viewed by a web browser. HTML uses tags to tell the web browser how to display certain content with support for links, pictures and video. HTML is only used to define the content of a web page and not how it should be formatted for the end user.

An important part of HTML is the DOM (document object model). When a web page is opened, the browser creates a document model of the page to which objects can be added, modified and removed.

4.4.2 CSS

CSS (Cascading Style Sheets) defines how HTML elements are displayed on a page, essentially allowing formatting to be added such as colour, fonts and

layouts. It allows web pages to react to different screen sizes and speeds up the development of projects as styles can be applied to groups of HTML elements.

4.4.3   JavaScript

Initially the web was made up of sites built using HTML and CSS serving static content. As the web progressed there was a desire from developers to create more interactive websites and to respond to these interactions without needing to make relatively expensive calls to a server. This is where JavaScript, a client side programming language, came in. JavaScript allowed elements in the DOM to be interacted with dynamically resulting in a much richer browsing experience.

4.4.4   Web game technologies

There are three potential choices when making a web based game - Adobe Flash, SVG and Canvas. Adobe Flash has been around for the longest, being released in 1996, and became hugely popular in the early 2000s as both a medium to play videos and make games. In 2001 SVG (Scalable vector graphics) was released as alternative, followed by Canvas a few years later. The choice of which one to use depends on a number of factors and can greatly impact the development time, accessibility and appearance of a game.

Flash

Adobe Flash allows games, animations and much more to be written using a subset of the JavaScript language, ActionScript. Due to its age it is a mature language and up until recently it was widely supported on all major platforms. As of September 2016, Google Chrome, the world's most popular web browser began to block flash and request the user to enable it if needed [LaForge, 2016]. Flash is also completely unsupported on Apple's iOS and despite being briefly available for Android is also no longer supported. For these reasons it was decided that flash would not be used as the development language for this project.

SVG

SVG (Scalable vector graphics) is a technology that was released in 2001. It uses XML to define vector images that are retained by the browser. SVG is used with the <svg> tag and acts similarly to a <div> HTML tag. It acts on a higher level than Canvas as objects are stored and remembered in the DOM. This means that all objects created are remembered by the DOM meaning they can be kept track of much easier and natively handle events such as mouse clicks.

Canvas

Canvas is a technology that was originally designed by Apple in 2004. It was later adopted as a web standard and included as part of HTML5, the current HTML standard introduced in 2014. By adding the <canvas> element to HTML it is possible to draw graphics using JavaScript. Canvas allows for the drawing of immediate mode graphics, meaning that once the pixels have been draw to the screen there is no record of these pixels maintained. In turn this means that any time something on the Canvas changes, everything needs to be redrawn. Canvas has been chosen for this project over Flash and SVG as it is the current cutting edge technology for building web games whilst providing a greater deal of flexibility over the other two.

Canvas Libraries

There are a large number of Canvas libraries out there that aim to simplify many aspects of using Canvas. Fabric.js, Paper.js, Easel.js and Phaser were all looked at with regards to being used in this project. The main features of these libraries include adding support for clicking and dragging objects on the Canvas, simplifying the drawing shapes and providing assistance in animation.

The disadvantage of using these libraries is that they are more focussed on traditional games that rely heavily on animations and not the slower puzzle style that is being developed in this situation. They also abstract away the inner workings of Canvas and can add unnecessary complications to developing what is a reasonably simple game. For this reason it was decided to write the game with no libraries but to just use pure JavaScript.

5. Implementation

5.1 Introduction

With essentially every feature of the game needing to be implemented from scratch there were a number of difficult problems that needed solving. This section discuses some of the more interesting and challenging issues and how they were solved. The reasoning behind the decisions is also mentioned, specifically regarding performance of the game, including how multiple solutions were tried to determine the best approach.

5.2 Multiple Canvases

One of the biggest optimisations that can be made to the Canvas environment is to use multiple Canvas elements to act as layers [Ranfelt, 2012]. By doing this it is possible to be selective about what items are cleared and redrawn in the animation loop. When anything is moved on the Canvas, everything must be cleared and redrawn. Initially, when only one Canvas was in use this meant redrawing the entire scene every time. When there is only a small amount of objects to draw this is negligible. Unfortunately, due to the nature of this game and the use of a flood fill algorithm to highlight segments of circles, using a one Canvas approach caused many problems.
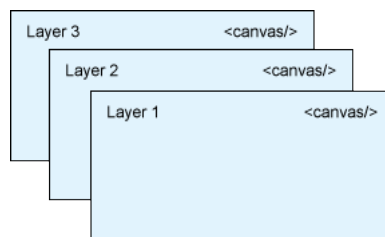
Figure 5.1: Canvas Layering

McGivery [2013] ran a number of tests that showed that creating as many as 500 Canvas elements had no noticeable effect on the time it took the function

requestAnimationFrame to complete an animation render cycle. Therefore seperating as many parts of the game in to their own Canvas environment added huge performance gains for only a marginal increase in complexity. These findings resulted in multiple Canvas elements being used in this project that allowed much finer control over when and what was being drawn to the Canvas.

5.3   Clicking on items

As previously mentioned, Canvas uses immediate mode graphics to draw pixels to the screen that are forgotten about once they are drawn. In turn this means that recognising when a user clicks on an object is not something that is natively supported and must be implemented manually.
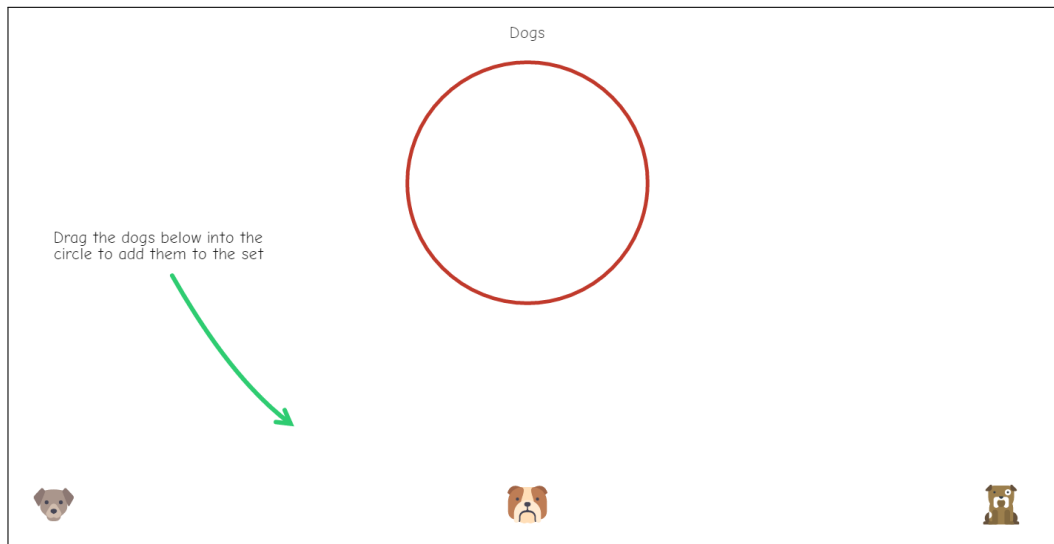
Figure 5.2: First level of the game

For example, in figure 5.2 there are three dogs that need to be dragged into the circle. In order to recognise when an object has been clicked on there a number of steps that must be taken. Firstly, when the level begins an array is created which holds the three images along with image properties such as the location on the canvas and its width and height.

```
[
  {
    "id": "dog1",
    "width": 51,
    "height": 51,
    "y": 857,
    "x": 294,
  },
  {
    "id": "dog2",
    "width": 51,
    "height": 51,
    "y": 857,
    "x": 934,
  }
]
```

Any time the user clicks anywhere on the canvas a check is done to see if the location of that mouse click is inside one of the images in the array. This is done by looping through the array that is holding the images and checking to see if the x and y of the mouse click fall inside the bounding rectangle of the image.

```
function imageClickedOn(x, y, movableTextArray) {
    for (var i = 0; i < movableTextArray.length; i++) {
        if (x >= movableTextArray[i].x &&
            x <= movableTextArray[i].x + movableTextArray[i].
                width &&
            y >= movableTextArray[i].y &&
            y <= movableTextArray[i].y + movableTextArray[i].
                height) {
            //return array index of image
            return i;
        }
    }
    return -1;
}
```

5.4 Dragging items

Dragging images and text around the screen is a core aspect of the gameplay. Firstly, when the user clicks on an image the algorithm in section 5.3 finds out which image was clicked on. The image that was clicked on is then stored through its array index. A drag offset, which contains the exact location on the image that was actually clicked, is also stored.
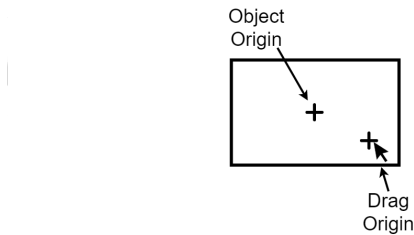
Figure 5.3: Initial click on object
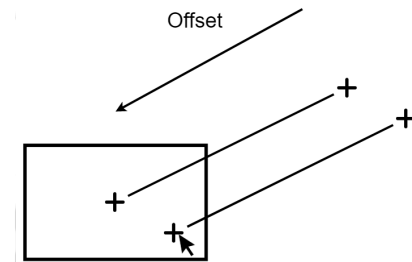
Figure 5.4: Object after being dragged

When the user then moves their mouse the mouse move event is fired. This updates the x and y location of the image to the new mouse location minus the drag offset that was stored. This reason that a drag offset must be stored is that without it the dragging would always take place from the x,y coordinates of the image. This would mean that regardless of where the user clicked on an image they would experience dragging from the origin.

Function	Operations per second
setInterval	39
setTimeout	43
requestAnimationFrame	61

Table 5.1: Animation test

Every time the mouse is moved until it is released the x and y get updated and everything on the screen redrawn. There are three ways to do this, setInterval, setTimeout and requestAnimationFrame. Initially, setTimeout was used as it is supported by all browsers. It works by calling a function every specified number of milliseconds. In table 5.1 the value 1000/60 is used

34

as the interval at which to call the function as this represents 60 frames per second. However, there was often a noticeable amount of delay when lots of intensive tasks were being animated, especially on lower powered devices such as mobile phones.

Therefore, another solution needed to be found. requestAnimationFrame is an API provided by browsers that handle the drawing of animations for both Canvas and DOM-based changes. This allows the browser to perform a number of optimisations such as preventing unnecessary redraws, call pool multiple animations into a single redraw cycle and prevent animations from running on tabs that are not currently open. As table 5.1 shows, requestAnimationFrame is considerably faster than the other two options. Unfortunately, not all browsers support requestAnimationFrame meaning that a polyfill is needed that allows the browser to fall back to setInterval should requestAnimationFrame not be supported.

```
window.requestAnimFrame = (function(){
  return  window.requestAnimationFrame      ||
          window.webkitRequestAnimationFrame ||
          window.mozRequestAnimationFrame    ||
          function( callback ){
            window.setTimeout(callback, 1000 / 60);
          };
})();
```

5.5   Floodfill

In order to allow the user to fill in sections of the circles to represent ideas such as the empty set, a flood fill algorithm is needed. A flood fill behaves in the same way as a paint bucket tool in many image editing programs, filling an area of connected pixels with the same colour. There are a number of different algorithms that can achieve this, with recursive and stack based solutions the two most popular. The issue with recursive solutions is that given a complex image there can be a huge number of recursive calls made, potentially causing stack overflow. Therefore, the implementation chosen was an iterative one that uses a stack and a while loop [Mallone].

- First the x,y location of the mouse click is added to a pixelStack

- While there are elements in pixel stack

  - Pop top value from pixel stack and assign to x and y

  - Travel upwards one pixel at a time until boundary hit or different colour pixel is found

  - Now travel downwards one pixel at a time until boundary hit or different colour pixel is found. This time colouring pixels with new colour as we go.

  - Now check pixel to left to see if it needs colouring

  - If it does need colouring add pixel to stack and set a boolean goLeft to true

  - Now check pixel to right to see if it needs colouring

  - If it does need colouring add pixel to stack and set a boolean goRight to true

  - Move down 1 pixel

  - Look left and right

    * If left or right needs colouring but respective boolean set to true then do not add to stack

    * Else if left or right needs colouring but respective boolean set to false then add to stack

  - Keep moving downwards until boundary or different colour pixel is hit

  - One column is now filled in, but the stack still has pixels in it so go back to start, pop next pixel and run through again.

This algorithm works well but unfortunately has a few problems. Figure 5.5 shows that when using this solution it is not possible to get a *clean* flood fill. This means that the colour black is not filled all the way to the edge and there is a jagged outline due the inherent problem that flood fill algorithms have with anti-aliased edges.

36
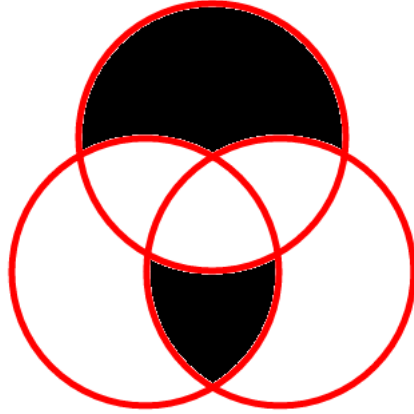
Figure 5.5: Stack based floodfill algorithm

A solution was found by using the library FloodFill2D which implements a more sophisticated floodfill algorithm to specifically deal with the anti-aliasing issue [Spronck, 2016]. This library allows a tolerance fade to be set that allows a soft edge to be created at boundaries. However, even that did not completely resolve the problem. In order to get a completely clean appearance to the floodfill more needed to be done. A new Canvas layer was created that sat on top of the one currently being used to draw the circles. This new layer would be used to draw identical circles with a thicker line width that would cover up the rough edges on the inner perimeter of the circle. This would not effect the floodfill as the algorithm only floodfills on one Canvas layer so has no knowledge of these new, thicker circles.
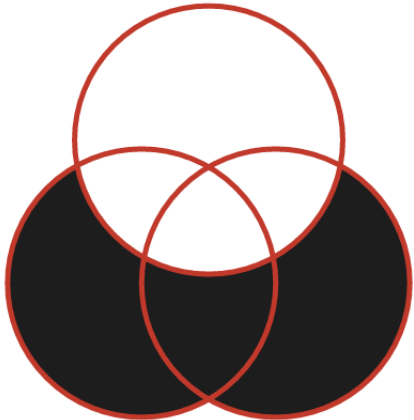
Figure 5.6: Final floodfill appearance

5.6 Visual Feedback

As one of the big advantages to using games to educate is that they can provide the user with immediate feedback it was important to try to incorporate this where possible. There are a number of ways that this has been achieved in this game. When placing items in the circle, a check is performed to ensure that the placement is valid and if so highlights the circle in green. This immediately shows the user that they have made a correct move. This is done by checking each premise is inside the same circle as another premise. If the premise is in its own circle then a new, green circle is drawn over the top to indicate a valid placement and remains highlighted until the user releases their mouse.

```
function checkPremiseAllInTheirOwnCircle() {
    var middle = whichCircleIsPremiseIn(level.
        movableTextArray[0]);
    var predicate = whichCircleIsPremiseIn(level.
        movableTextArray[1]);
    var subject = whichCircleIsPremiseIn(level.
        movableTextArray[2]);

    if (middle == null || predicate == null || subject ==
        null) {
        return false
    } else {
        return !(middle.equals(predicate) || middle.equals(
            subject) || predicate.equals(subject));

    }
}
```

Figure 5.7: Correct Placement with highlighting

Figure 5.8: Incorrect placement with no highlighting

Other feedback includes changing the font colour of the statements. When completing the syllogism levels there are 3 statements that must be made true; the major premise, the minor premise and the conclusion. When the user makes any of these three true the font colour of the sentence changes to green.
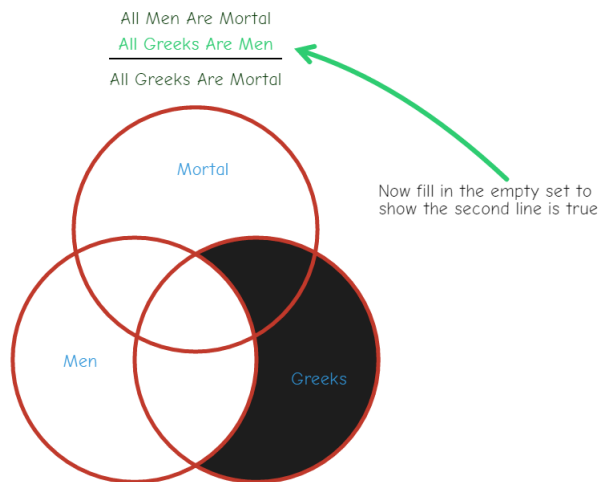
Figure 5.9: Premise Highlighting

5.7 Undo and Redo

As Canvas does not maintain any record of drawings on the canvas there is no in-built functionality that allows a user to undo or redo an action.

There are two ways this can be done, storing everything that is drawn on the canvas in a stack or capturing the screen as a image and using that image to redraw the canvas. Both options were investigated to see which was the best.

Tests were ran on the execution time of both options. Saving the entire canvas as an image took 16 milliseconds whilst adding the game state to a stack was 160 times faster, taking just 0.1 milliseconds. Despite 16 milliseconds being an acceptable amount of time for an action to take, when combined with other functions that must also run the performance can be impacted. Also as the number of moves increases the number of canvas images that must be stored increases leading to memory issues on certain browsers. Another issue with using the image method is that whilst the canvas will revert to its previous image, the underlying state of the game will be incorrect. This is not an issue when using the stack method as the entire state of the game is stored.

Every time a move is made, the current state of all the objects are stored in a data structure and pushed to the stack. Despite there being many different objects on the Canvas, the only changes the user can make to the canvas is the location of the movable text and the flood filled sections. Therefore the current state of the game can be represented as

```
var GameState = function (movableTextArray, clickedInArray) {
    this.movableTextArray = movableTextArray;
    this.clickedInArray = clickedInArray;
};
```

When the user presses the undo button, the top object in the stack is popped and everything is redrawn. Redo works in a similar manner, as time an undo is performed the GameState object is popped from the undo stack is added to a redo stack. The undo and redo buttons then can push and pop from their respective stacks allowing the user to go back and forth between their moves.

5.8 Recognising when level is complete

With the user being able to drag the premises into any circle of their choosing it means that there are multiple ways that a completed level can be represented. Therefore, it is crucial that the game is able to recognise any valid arrangement of a completed level.
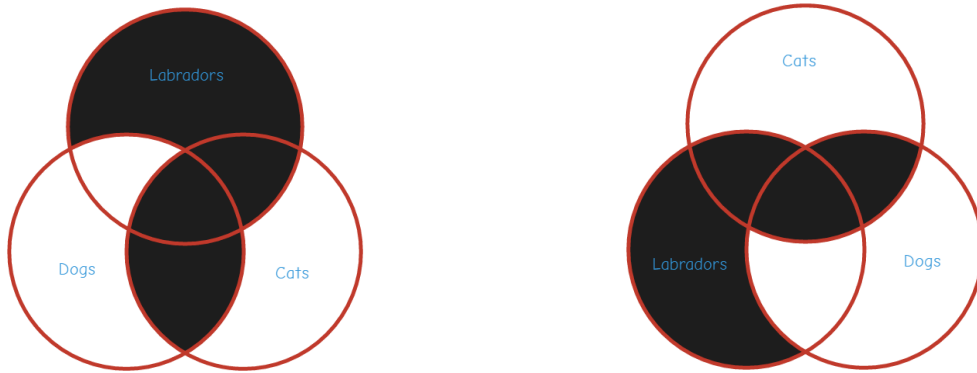
Figure 5.10: Two different ways to represent the same syllogism

The ability to recognise correct syllogisms in any form is done by first finding which circle each premise in located in:

```
var middle = whichCircleIsPremiseIn(level.
    movableTextArray[0]);
var predicate = whichCircleIsPremiseIn(level.
    movableTextArray[1]);
var subject = whichCircleIsPremiseIn(level.
    movableTextArray[2]);
```

The location of each intersection is then also stored based on the location of the middle, subject and predicate. The correct answer is for each level is stored in the JSON as so:

```
"correctSyllogism": {
     "subject": true,
     "middleSubjectIntersection": true,
     "subjectPredicateIntersection": true,
     "middlePredicateIntersection": false,
     "middleSubjectPredicateIntersection": false,
     "predicate": false,
     "middle": true
}
```

A new object is created with the same structure as the correct syllogism object. Every time the user clicks in a segment of the circles, which fills it black, the location of the click is stored. This array of user clicks is used in conjunction with the location of the text to set each location to true or false. From there the correctSyllogism object stored in the game level JSON can be compared with the new object and if they are equal then it means that the level has been completed.

5.9   Saving the game

One requirement was that it must be possible for the game to be saved so that the user could return at a later date and carry on playing from where they left off. There are a number of possible solutions for saving the state of a game in the browser.

Previously, any storage of data in the user's browser would have to have been done through the use of cookies. A cookie is a way to store a small amount of data, up to a maximum of 4kb, in a text file on a user's computer which expire after a certain amount of time.

With the introduction of HTML5 a new method for storing data in the browser was added, localStorage. This new mechanism allows 5MB of data to be stored locally on the browser and is then accessible through any JavaScript that is running on the same domain. It has been implemented here by saving the user's progression after every level that is completed [Mills, 2017].

In order to save the game an object is created that holds the current state of the game. This contains both the current level number and the name of the user. Every time a level is completed the object is updated with the new level number and that object is then saved. There is no need to remove the

old object as the localStorage API silently overwrites the existing object if the key already exists.

```
\\save object
localStorage.setItem('gameObject', JSON.stringify(game));
```

The next time the user returns to the web page this saved object is retrieved by the browser.

```
var retrievedObject = JSON.parse(localStorage.getItem('
    gameObject'));
```

The option to resume play is presented to the user through a 'continue' button on the home page. The continue button is initially hidden and by checking that the game object retrieved from local storage is not null it can be made visible.

```
if (retrievedObject != null) {
    $("#continueButton").visible();
}
```

5.10   Saving data to database

One of the requirements was for a leaderboard that provides users with the option to enter their name and be able to see how they rank alongside other people who have played the game. This also doubles up as a way to gain an extra layer of analysis by being able to take a deeper look at how long levels took and the number of moves that were made.

Initially a REST (REpresentational State Transfer) service was to be created to handle the transport of data. REST uses the HTTP protocol to manipulate data through requests such as GET, which returns data from a datbase and POST which persists data in a database. The REST service would act act as the intermediary between the client side and the database.
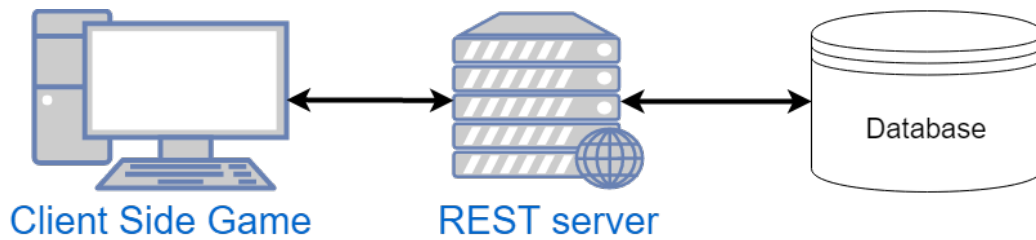
Figure 5.11: Basic REST service architecture

This would require writing REST endpoints to handle such data using a language like Node.js, a variant of JavaScript designed to run server side. The problem with this solution is that it requires not only the writing of a REST service but also setting up of a database and acquiring a server to host both of these on. This is overkill for the needs of this project as no sensitive data is being stored, only a few columns of data are needed and speed and number of possible connections that can be sustained are of no concern.

Therefore, for this project Google Sheets is being used as a cloud based database. Google sheets is a web based spreadsheet application, very similar in functionality and appearance to Microsoft Excel. Whilst Google Sheets does not offer the same depth of functionality as desktop based solutions it is perfect for storing a relatively small amount of data.

Google Sheets does not explicitly support the functionality of using it to act as a database, and as such there are a number of workarounds that must be carried out to make this possible.

Google Scripts is a cloud based scripting platform operated by Google that allows JavaScript code to be written that can interact with other Google products, such as Sheets. By using this, it is possible to create what is equivalent to a REST endpoint all hosted within Google's platform. Data can then be sent to this Google Script URL from the game's code. The Google Script will then take in this data, process and format it and insert it into a Google Sheet [Hawksey, 2014].

In the game's code an object is created that stores all the data that is to be saved. This includes the time and moves taken to complete each level, the player's name and the total time. This is then serialised, converted from an object to a stream of bytes, for use in a URL query string. This serialised object is then sent to the Google Scripts endpoint via an AJAX request. AJAX (Asynchronous JavaScript and XML) allows client side code to

44

communicate with a web server without needing to reload the page. The asynchronous nature means that the client side code will continue to execute without waiting for a reply, returning only once a reply has been received. This prevents the user from having to wait for the request to be returned.

Initially the game data was only saved once the whole game had been completed. This was because it was an expensive process to carry out, taking up to 1 second to save the data. However, during the initial user testing phase there were issues with the users either not fully completing the game or bugs that caused the game to enter a state that could not be progressed from. This meant the users never reached the final stage and the data was not saved. With the analysis of this data being such an important part of the project it was decided to save the data after every level.

```
request = $.ajax({
    url: "https://script.google.com/macros/s/
        AKfycbxVGVxoQxNFK7_nxKfglL8yLNUmdPwP2e9j8IMO6JY5wzLEdSE
        /exec",
    type: "post",
    data: serializedData
});
```

5.11   Reading from database

Saving data to the database is just one part of the problem, in order to have a functioning leaderboard system it must also be possible to read that data back in to the application.

The JavaScript library Sheetrockjs allows Google Sheets to be interacted with as if they are a traditional database through the use of SQL-like statements [Zarate, 2015]. For example, in traditional SQL the query to get the 5 fastest play through times would be:

```
SELECT TOP 5 time_taken, moves_made, levels_skipped
    FROM leaderboard ORDER BY time_taken ASC;
```

In Sheetrock.js a similar syntax is used:

```
$('#spreadsheet').sheetrock({
    url: mySpreadsheet,
    query: "select B,W,X order by X asc",
    fetchSize: 5
});
```

By providing the URL of the spreadsheet, the query and the number of results that are required the data is returned from Google Sheets. This data is then automatically used to populate a HTML table with the data in the form of a leaderboard.

5.12 Working across multiple platforms

As this game is designed to run across browsers and mobile devices, it is important that the experience is consistent across all potential screen resolutions. The way that has been chosen to do this is everything drawn on the canvas is relative to the height and width of the screen resolution. For example, when creating the circles to represent the venn diagram the following is done:

```
if (circlesNeeded === 3) {
    //split the screen in to 4 vertical segments
    var segments = canvasHeight / 4;
    //set the radius of each circle
    var radius = segments / 2 * 1.25;

    var overlap = radius / 2;
    circlesArray.push(new Circle((canvasWidth / 2), (segments
        + (segments * 2)) / 2 + (segments / 6), radius));
    circlesArray.push(new Circle((canvasWidth / 2) - overlap,
        (segments * 2 + (segments * 3)) / 2 - (segments / 6),
        radius));
    circlesArray.push(new Circle((canvasWidth / 2) + overlap,
        (segments * 2 + (segments * 3)) / 2 - (segments / 6),
        radius));
```

This allows the user interface to look exactly the same across all screen resolutions. Canvas is also supported by Android and iOS browsers but some changes must be made to actually make it work for touchscreen devices. Touch events must be defined to handle the user touching the screen and dragging their finger along the screen. This is relatively straightforward when using jQuery as event handlers for these type of interactions are provided.

6. Testing

6.1   Test Driven Development

Test Driven Development is a process that involves incrementally developing software by writing unit tests before any functional code is written. The following process covers how to use test driven development when writing unit tests. A unit test aims to take the smallest piece of testable code from an application and test it in isolation.

- Write a unit test

- Run the unit test to ensure that it fails

- Write the bare minimum functional code to make the unit test pass

- Refactor the code

There are a number of benefits to using test driven development when working on large software projects. Müller and Padberg [2003] found that whilst test driven development might result in more overall code being written for a project the total time to actually complete it can be shorter. One of the main benefits is that it provides security when making any major code changes, something which happened frequently during the development stage of this project. This is because the suite of tests can immediately be re-run and any breaking changes are quickly highlighted to the developer.

As the entire user interface is written exclusively in Canvas and the use of HTML and CSS is very minimal any unit tests would be explicitly testing the Canvas aspect of the code. Writing unit tests to ensure that certain UI elements are drawn, such as circles, text or images is not really possible given Canvas' 'draw and forget' approach. Therefore, all testing of the UI was done manually by hand.

There are a number of functions that do lend themselves to be unit tested. Functions that simply take data in and perform checks and operations on it are perfect examples. In the scope of this project this concerns functions that check to see when levels are complete, check to see if user clicks are

47

in a certain place and setting up shapes ready to be drawn. Jasmine, a behaviour driven framework for testing JavaScript code was used to write the unit tests.

Below is an example unit test that ensures the correct number of circles with correct sizes are setup for each level.

```javascript
describe("Given a number of circles to setup", function () {
    it("sets up that many circles images", function () {

        var x = 714;
        var y = 702;
        var canvasWidth = 1920;
        var canvasHeight = 935;

        var circlesArray = [];

        var expected =[];
        expected.push(new Circle(960, 467.5,
            155.83333333333334));

        var actual = setupCircles(1, canvasHeight,
            canvasWidth, circlesArray);

        expect(_.isEqual(actual, expected)).toEqual(true);
    });
});
```

By having a complete set of unit tests written for all testable functions it gives confidence that code will perform as intended.

6.2   Integration Tests

Integration tests are an extension of unit tests that combine multiple smaller parts of the code and test that they behave correctly when combined with each other.

The main advantage of writing integration tests are that it can find issues that only occur when two separate parts of the software interact, something that would not arise using solely unit tests.

An example integration test written for this project was to test that saving the game object to the database worked correctly, and it could also be retrieved to populate the leaderboard. This was an important test to write

as the participant data gained would be used in the evaluation. If there were any issues it would result in the data being lost which would in turn impact the analysis that could be carried out on the effectiveness of the game.

The library used to save data to the Google Sheets database can only send POST requests and has no support for DELETE. Due to this it was not possible to delete the entries made when running the tests. This means that a second test database was created as to not interfere with the production database that was used for the leaderboard.

Integration tests tend to be harder to write due to the interactions between the different parts of the code and this example was no exception. Due to the asynchronous nature of both adding to and reading from the database the assertions of the tests cannot be run until all the AJAX calls have successfully completed.

This is not completely straightforward to manage using the Jasmine framework, which supports asynchronous testing through the use of the *done* keyword. Once an asynchronous call has completed the function *done();* is called allowing the code to progress on.

The test below works by inserting a new record in the database with a randomly generated ID, then that same ID is used to ensure that the record can be retrieved. A check is then done to ensure that this unique record is the only one returned.

```
describe("POST data to Google Sheets", function() {
    var id;
    var returnedSpreadsheetResponse;

    beforeEach(function(done) {
        var testSpreadsheetURL = "https://script.google.com";
        id = generateGuid();
        var myData = {"name": id, ..., "totalTime": 100};
        postData(myData, testSpreadsheetURL);
        done();
    });

    beforeEach(function(done) {
        var myCallback = function(error, options, response) {
            returnedSpreadsheetResponse = response.rows.
                length;
            done();
        };

        $('#spreadsheet').sheetrock({
            url: "https://docs.google.com",
            query: "select B,Y,AD where B = '" + id + "'
                order by AB asc",
            labels: ['Name', 'Levels Skipped', 'Time'],
            fetchSize: 10,
            callback: myCallback
        });
    });

    it("Sheetrock.js can read data back", function() {
        expect(returnedSpreadsheetResponse).toEqual(2);
    });
});
```

7. Evaluation

7.1   Introduction

The next stage of the project was to carry out an assessment with users in order to test both the effectiveness of the system and see if the requirements had been met. The focus with this evaluation was on the effectiveness of the game to teach the player about syllogisms.

The experiment consisted of three parts that the participants were required to carry out in succession. The study included 18 participants made up of a wide variety of ages, academic backgrounds and mathematical experiences.

Two questionnaires were completed, one before and one after the game had been played. Across the two questionnaires, four riddle questions were asked to participants. The first riddle in each questionnaire closely resembled a syllogism in its sentential form, with the second being a more traditional style riddle. This provided another opportunity to try and find a link between performance in the game and how well participants would answer these questions. It was expected there would be more correct answers to the riddles in the post game questionnaire as participants would have just learnt about syllogisms.

The questionnaires laid out a series of statements that the participants specified their level of agreement or disagreement using the following scale.

- Strongly agree - 5
- Agree - 4
- Neither agree nor disagree - 3
- Disagree - 2
- Strongly disagree - 1

The rationale behind the questions was to build up a picture of the experience of each participant who played the game. The expectation was that those

with a stronger mathematical background and those who were doing a degree involving some form of maths would perform better.

7.1.1 Pre Game Questionnaire

The participants were were required to first complete a questionnaire to gauge how comfortable and experienced they were with different aspects of maths.

1. *Did you do A Level Maths?*

   8 of the 18 studied A Level maths, an optional qualification taken in the United Kingdom by pupils aged between 16-18. Conversely 10 did not study this subject for A Level. The people who studied A level maths will here on out be group 1 and those who did not study it will be group 2.

2. *Does your degree contain any of the following forms of mathematics?*

   - Statistics

   - Pure Mathematics

   - Computer Programming

   - Excel Spreadsheets

   - None of the above

     As is clear to see in figure 7.1, there is a significant divide between the two groups, most notable in the pure maths and computer programming section. Only three people from group 2 have done a degree that contains any form of maths.
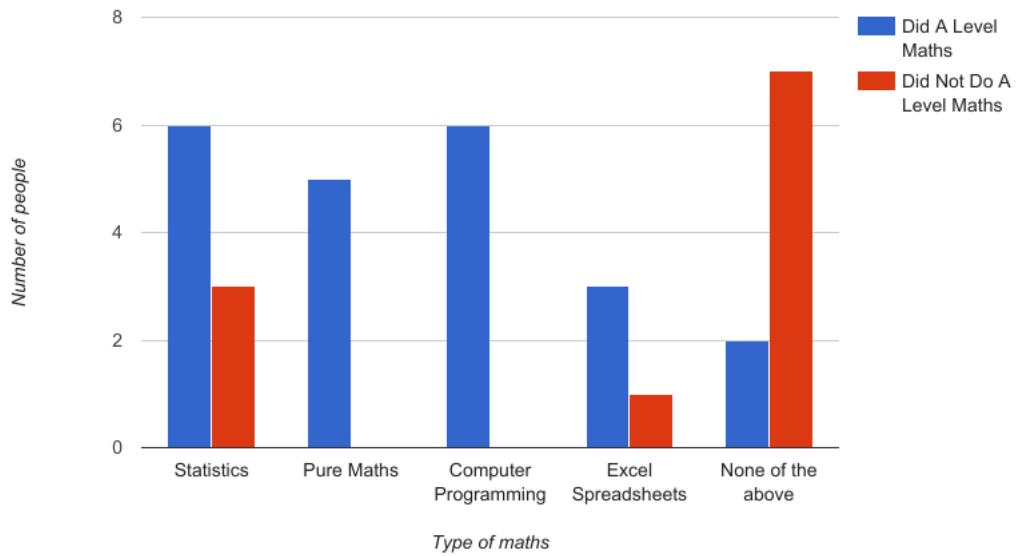
Figure 7.1: Types of maths in participants degree course

3. *I enjoy maths*

   Unsurprisingly group 1 participants liked maths whereas the opposite was true for group 2, with an overall average of 3.4.

4. *I enjoy puzzle games*

   A large correlation could be seen here between those who enjoy maths and those who enjoy puzzle games. Overall however, the participants prefer puzzle games to maths, with the average being 3.9.

5. *Riddles*

   All participants got the first riddle correct and only five, with a two and three split between the group 1 and group 2 respectively, got the second riddle wrong. With so many participants getting the answer correct there is little that can be learnt from this.

7.1.2   Game

The participants were then required to play through the game, completing as many levels as they were able to. As mentioned in the requirements, the game will save both the time each level takes to complete as well as the number of moves each participant makes.

Of the 18 participants, 11 people completed the game in its entirety without skipping any levels. Perhaps the most interesting observation from the results is the split in group 1 and group 2 in terms of how well they did at the game. All eight participants from group 1 completed the game with no skipped levels in an average time of 12 minutes and 29 seconds. On the other hand, only three out of ten people fully completed the game from group 2 taking an average of 25 minutes and 33 seconds.
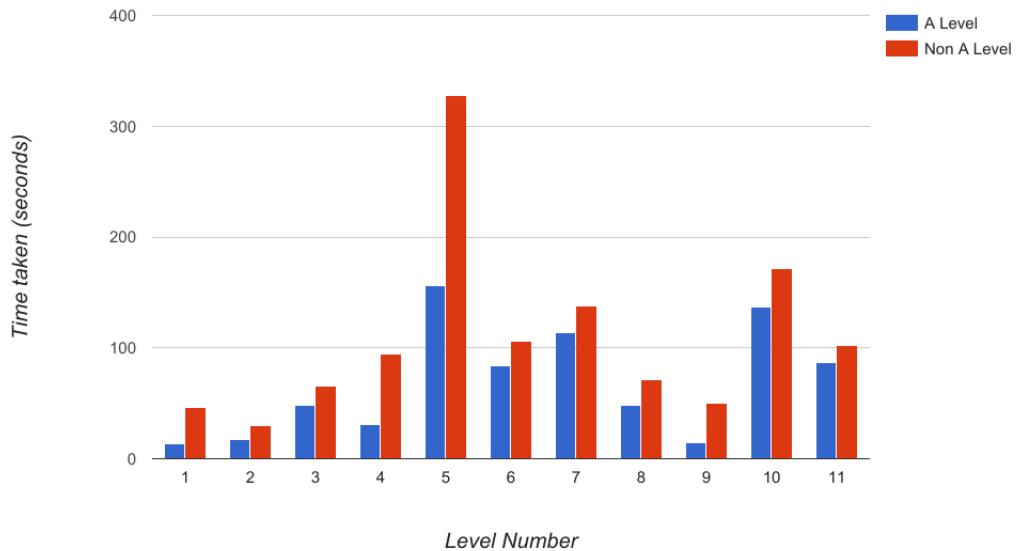
Figure 7.2: Average time to complete each level

Figure 7.2 shows the average time taken for each level but only includes times where the participant completed the level and did not skip it. As can be seen, the time was shorter for those who have done A level maths. The difference between average times varied significantly depending on the level. Level 5 sees a large increase in real terms, potentially as it is the first level

that introduces a totally new concept to the participants - the syllogisms. Interestingly, level 1 also also takes group 2 participants over four times as long to complete. This is surprising as level 1 is a simple level that just requires three pictures to be dragged into a circle. This substantial difference in time between the two groups suggests that group 2 was less comfortable with the interface.

A similar trend can be seen by the number of moves taken by the two separate groups, shown in figure 7.3. Again, the number of moves only includes participants who completed the level. The number of moves taken to complete level 11 was not recorded due to a technical issue. Level 5 again sees a massive jump in the number of moves it takes group 2 to complete it. With this group taking on average over 100 moves to complete the level, this in isolation might lead to the assumption that they progressed purely by guessing randomly until they got the correct answer. However the following level, level 6, was almost exactly the same question as level 5 with the tutorial system providing much less assistance. But as figure 7.2 and 7.3 show, both the times and number of moves made for level 6 drop drastically. This implies that whilst initially the new concept took longer for the non mathematicians to grasp, once they had got over the initial hurdle they improved massively. This suggests that they did actually understand the concept and ideas behind the syllogisms.

Similar jumps can be seen in level 7, 8 and 10 which can also be attributed to the fact that new concepts were being introduced.
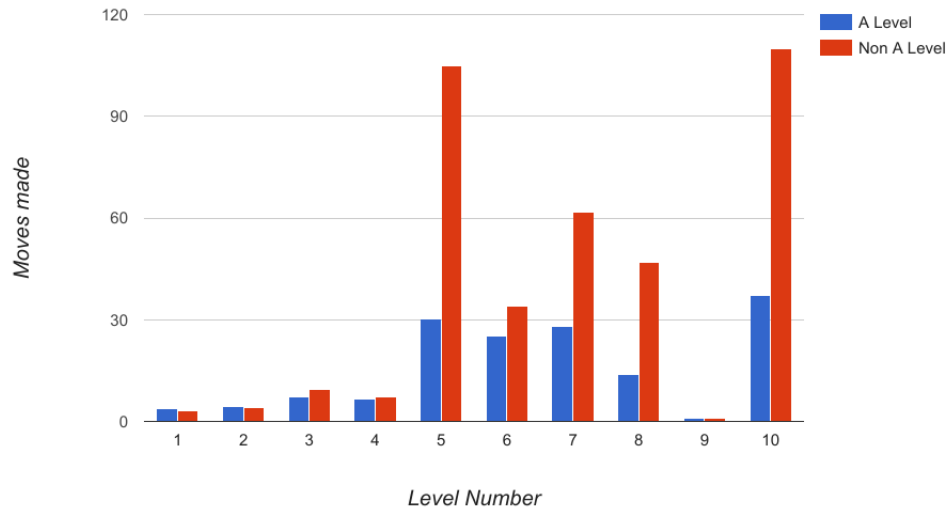
Figure 7.3: Average number of moves to complete each level

7.1.3 Post Game Questionnaire

Finally after playing the game there was a post-game questionnaire that asked how participants found the experience, if they had learnt anything and how usable the game was.

1. *I think I did well at the game*

    There was a clear divide in answers of this question between participants depending on which group they were in. Those in group 2 reported an average of 2.5 whereas in group 1 had a score of 3.6.

2. *I enjoyed playing the game*

    As enjoyment plays such a important role in the motivation to play a game this was an important score. The average showed 3.7 with a trend showing the better performing participants enjoyed the game more.

3. *I found the game too difficult*

    Again a large split could be seen between those with maths experience and those without. The average in group 2 was 4.2 showing that as a group they found the game very difficult. In comparison group 1 had

a score of 2.3 meaning they did not find the game too hard.

4. *I found the game engaging*

   The engagement the game provided was very positively received, with an average of 4. This reinforces the work of Abuhamdeh and Csikszentmihalyi [2012], as despite the fact participants found the game difficult they still were engaged by the game.

5. *I found the interface easy to use*

   This was another area where there was a large focus during development. The work on making the UI as simple as possible paid off as the score received was 4.1. A number of participants made comments that clicking and dragging items around the screen was too hard. This was due to the fact that the text, and therefore the hitbox, was slightly too small meaning the user had to be precise when clicking. This had a effect on the score for the interface usability. One participant also completed the game on a mobile device, giving the interface a score of 5. The participant remarked that it was slightly fiddly on a small screen but worked well.

6. *The game ran smoothly with no issues*

   The average for this question was 3.6. This was partially due to the fact that the first few participants to play the game ran into a number of issues. These problems were not found during the testing phase and therefore were present when the participants were playing the game, on one occasion causing the game to completely crash. The low score from the affected users reflects the bugs that they faced. Once the bugs had been fixed the score rose to 4.1.

7. *The game introduced concepts effectively*

   This was the strongest attribute of the game scoring on average a 4, with only two participants giving it less than a 4.

8. *I learnt something new*

   This question received an average of 3.7 showing that most people felt that they did learn something. The three best performers of the game based on completion time all felt as though they did not learn anything giving it a score of 1. However, those who did not perform as well provided an average score of 4 showing that even though they might have struggled they did benefit from the game.

9. *Riddles*

Similarly to the pre game questionnaire the vast majority of participants answered in the same way. The first riddle three people got wrong, including participants from both groups. The final riddle given to the participants was an incredibly challenging one that saw only three people get the correct answer, again with a mixture of the two groups.

8. Conclusion

Overall the game was a success with the high level of engagement, enjoyment and educational effect that it achieved. It is fair to say that it is more suited to those with more mathematical experience as they will both enjoy and learn more. However it does have its benefits for others, as the results show that nearly everyone felt that they did learn something.

The user testing carried out gave an excellent opportunity to see the difficulties that people faced when learning syllogisms through this game. The first few levels were easy but the learning curve reached at level 5, where the first syllogism was introduced, perhaps was too great. There was an attempt to mitigate against this, through both the tutorials and intermediate levels. The game would have worked better by increasing the number of levels, in order to more gradually increase the difficulty. These levels would have introduced the concepts at a slower pace and this build up would have greatly improved the learning experience.

If more levels were added, the system that is currently in place would provide a way to look more deeply at where the problem lies when learning about syllogisms. Currently, the data collected shows that participants began to struggle when the syllogisms were added, but if the game was changed to increase the difficulty more slowly through a greater number of levels it would be possible to more accurately pin point what people are struggling with. This is deeper level of analysis is something that would not be possible through more traditional learning styles, so is a huge benefit of using a serious game.

A surprising aspect of the questionnaire results was the number of participants who got the answers to the syllogism style riddle correct. As discussed in the literature review, both Larkin and Simon [1987] and Johnson-Laird [1980] both found that sentential representation of syllogisms was an ineffective way for them to be represented and reasoned about. This goes against what was found in this study, with participants being more capable of understanding about syllogisms in their text form. This could the result of a number of reasons, it is possible that the interface and tutorials did not make it clear what was being asked of participants. Participants might have been overwhelmed by being asked to both solve a

riddle in the game whilst at the same time using a system and representation they are not familiar with.

A more clear focus on the target audience of the game would have been of benefit. It could be said this game is trying to cater to both ends of the spectrum, in such a way that it is both too difficult for some and too easy for others. By having a more defined group of people for who the game is designed for then a more focussed approach to teaching could have been used.

From a technical perspective there are a number of things that would have been done differently if this game was developed again. Whilst at the time the decision to use pure Canvas with no use of any libraries seemed a good choice, on reflection this appears not to be the case. The time spent implementing dragging and dropping and recognising when certain items were in certain places added a significant amount of time to development.

If this project was done over there would definitely have been a more thorough investigation into the benefits of using the multiple Canvas libraries including trying a few of them out to see how they could have simplified the process.

As it stood, with the game not relying heavily on animations it was felt that they were not worthwhile but this decision was taken before there was a full understanding of how much work would be required to get the core functionality of the game up and running.

It is also possible that using Canvas in the first place was the wrong technology, and that SVG would have been far more suitable for a game of this nature. Canvas' strengths lie in its performance and versatility in creating 3d immersive games. These strengths were never hugely relevant to a game of this style so might not have been the most suitable technology to use. Perhaps the longest amount development time was creating the tutorials for each level. The work was far less challenging than the algorithms needed to actually make the game mechanics work but drawing both text and arrows on the Canvas was a surprisingly lengthy process. If SVG had been used instead it would have been far easier to structure the tutorial layouts with HTML and CSS which would have greatly reduced the time taken to implement them.

8.1 Further Developments

There are a number of further improvements that could be made to the game.

Level creator

A level creator could be added that allowed users to create their own levels. The infrastructure is already in place to facilitate this as levels are simply a JSON object. This means that it would be straightforward to create an online editor that allowed users to input their own JSON or modify existing JSON to add more levels. Adding levels through JSON might be too advanced for the average user so it could also be possible to use the existing graphical interface to add new levels. By providing three blank circles and 3 blank text boxes the user could build their own levels.

Alternative syllogism representations

For this project Venn diagrams were the chosen method to represent syllogisms. As a potential avenue to explore the game could be recreated using the alternative methods discussed in the literature review. The study could then be re-run with the multiple representations in order to determine which is the simplest and most effective. This would be done in a similar manner to the work carried out in this project, by measuring the length of time to complete the game and asking for participant feedback on the level of difficulty.

Bibliography

Sami Abuhamdeh and Mihaly Csikszentmihalyi. The importance of challenge for the enjoyment of intrinsically motivated, goal-directed activities. *Personality and Social Psychology Bulletin*, 38(3):317–330, 2012.

Joan Bagaria. Set theory. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, fall 2016 edition, 2016.

Peter C-H Cheng. Graphical notations for syllogisms: how alternative representations impact the accessibility of concepts. *Journal of Visual Languages & Computing*, 25(3):170–185, 2014.

Peter CH Cheng. Visualizing syllogisms: category pattern diagrams versus venn diagrams. 2012.

Roy B Clariana, Daren Wagner, and Lucia C Roher Murphy. Applying a connectionist description of feedback timing. *Educational Technology Research and Development*, 48(3):5–22, 2000.

Mihaly Csikszentmihalyi, Kevin Rathunde, and Samuel Whalen. *Talented teenagers: The roots of success and failure*. Cambridge University Press, 1997.

George Englebretsen. Linear diagrams for syllogisms (with relationals). *Notre Dame J. Formal Logic*, 33(1):37–69, 12 1991. doi: 10.1305/ndjfl/1093636009. URL http://dx.doi.org/10.1305/ndjfl/1093636009.

J. R. Erickson. Research on syllogistic reasoning. In Russell Revlin and Richard E. Mayer, editors, *Human Reasoning*, pages 39–50. Distributed Solely by Halsted Press, 1978.

Mark Griffiths. The educational benefits of videogames. *Education and Health*, 20(3):47–51, 2002.

John Hattie and Helen Timperley. The power of feedback. *Review of educational research*, 77(1):81–112, 2007.

Martin Hawksey. Google sheets as a database – insert with apps script using post/get methods (with ajax example), Jul 2014. URL `https://mashe.hawksey.info/2014/07/google-sheets-as-a-database-insert-with-apps-script-using-postget-methods-wit`

Hammer Hill. Existential import. http://cstl-cla.semo.edu/hhill/pl120/notes/existential%20import.htm, 2016.

Willem Homan and Kathleen Williams. Pc-based flight simulators: No more games. *Tech Directions*, 58(2):33, 1998. ISSN 10629351. URL `http://search.ebscohost.com.ezproxy1.bath.ac.uk/login.aspx?direct=true&db=bth&AN=1117015&site=ehost-live`.

Patrick Hurley. *A concise introduction to logic*. Cengage Learning, 2005.

Alastair Irons. *Enhancing learning through formative assessment and feedback*. Routledge, 2007.

Philip N Johnson-Laird. Mental models in cognitive science. *Cognitive science*, 4(1):71–115, 1980.

Paul Kawachi. Initiating intrinsic motivation in online education: Review of the current state of the art. *Interactive Learning Environments*, 11(1): 59–81, 2003.

Meggin Kearney. Measure performance with the rail model, Apr 2017. URL `https://developers.google.com/web/fundamentals/performance/rail`.

Anthony LaForge. Flash and chrome, 2016. URL `https://blog.google/products/chrome/flash-and-chrome/`.

Leslie Lamport. What good is temporal logic? In *IFIP congress*, volume 83, pages 657–668, 1983.

Jill H Larkin and Herbert A Simon. Why a diagram is (sometimes) worth ten thousand words. *Cognitive science*, 11(1):65–100, 1987.

Clare Lee and Sue Johnston-Wilder. Learning mathematics—letting the pupils have their say. *Educational Studies in Mathematics*, 83(2):163–180, 2013.

Oliver Lemon, Ian Pratt, et al. On the insufficiency of linear diagrams for syllogisms. *Notre Dame Journal of Formal Logic*, 39(4):573–580, 1998.

Mark R Lepper. Motivational considerations in the study of instruction. *Cognition and instruction*, 5(4):289–309, 1988.

Cher Ping Lim. Spirit of the game: Empowering students as designers in schools? *British Journal of Educational Technology*, 39(6):996–1003, 2008.

Keith Lumsden and Alex Scott. A characteristics approach to the evaluation of economics software packages. *The Journal of Economic Education*, 19 (4):353–362, 1988.

William Mallone. Create a paint bucket tool in html5 and javascript. URL `http://www.williammalone.com/articles/html5-canvas-javascript-paint-bucket-tool/`.

Thomas W Malone. Toward a theory of intrinsically motivating instruction. *Cognitive science*, 5(4):333–369, 1981.

Andrew McGivery. How many stacked canvas elements is too much?, 2013. URL `http://mcgivery.com/how-many-stacked-canvas-elements-is-too-much/`.

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

Chris David Mills. Window.localstorage, Mar 2017. URL `https://developer.mozilla.org/en/docs/Web/API/Window/localStorage`.

Matthias M Müller and Frank Padberg. About the return on investment of test-driven development. In *Edser-5 5 th international workshop on economic-driven software engineering research*, page 26, 2003.

Jakob Nielsen. *Usability engineering.* Elsevier, 1994.

Diana Oblinger, James L Oblinger, and Joan K Lippincott. *Educating the net generation.* Boulder, Colo.: EDUCAUSE, c2005. 1 v.(various pagings): illustrations., 2005.

Office For National Statistics. Internet access - households and individuals. Technical report, 2016. URL `http://www.ons.gov.uk/peoplepopulationandcommunity/householdcharacteristics/homeinternetandsocialmediausage/bulletins/internetaccesshouseholdsandindividuals/2016`.

Marc Prensky. The games generations: How learners have changed. *Digital game-based learning*, 1, 2001.

Adam Ranfelt. Optimize html5 canvas rendering with layering, 2012. URL `https://www.ibm.com/developerworks/library/wa-canvashtml5layering/`.

John W Rice. Assessing higher order thinking in video games. *Journal of Technology and Teacher Education*, 15(1):87, 2007.

Robin Smith. Aristotle's logic. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Winter 2016 edition, 2016.

Ian Sommerville. *Software Engineering: (Update) (8th Edition) (International Computer Science)*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2006. ISBN 0321313798.

Mark Spronck. Floodfill2d. `https://github.com/blindman67/FloodFill2D`, 2016.

W3Schools. Browser statistics, Mar 2017. URL `https://www.w3schools.com/browsers/`.

Edward N Zalta. Basic concepts in modal logic. *Center for the Study of Language and Information Publications*, page 2, 1988.

Chris Zarate. Sheetrock.js. `https://github.com/chriszarate/sheetrock`, 2015.

Appendices

Table 1: Time to complete level data

id	level1time	level2time	level3time	level4times	level5time	level6time	level7time	level8time	level9time	level10time	level11time	levelsSkipped	totalTime
1	7	14	18	11	15	22	35	28	7	20	21		199
2	13	18	61	31	143	113	154	39	7	180	67		826
3	16	15	56	43	73	41	92	39	9	114	61		559
4	22	16	39	54	126	99	112	46	26	93	52		684
5	14	15	27	20	125	35	255	45	16	276	101		928
6	11	25	98	36	411	118	96	78	17	241	311		1442
7	15	18	49	30	286	220	115	69	12	93	44		951
8	17	22	63	39	387	43	234	109	51	187	124		1274
9	30	39	69	68	747	377	0	0	0	0	0		1330
10	29	29	67	116	0	0	0	0	0	0	0	5, 6, 7, 8, 9, 10, 11	240
11	193	14	34	42	197	94	52	78	45	4	3	6, 8, 10, 11	755
12	12	17	59	42	262	49	284	174	40	425	112		1475
13	20	24	80	48	541	169	172	115	310	179	192		1850
14	22	17	38	232	328	72	110	113	47	66	23	5, 7, 8, 10, 11	1068
15	49	55	0	113	420	67	229	15	0	0	0	5, 6, 7, 8, 9, 10, 11	947
16	38	40	132	73	1201	84	64	1	19	91	0	6, 7, 8, 10	1743
17	55	41	51	171	0	0	0	0	0	0	0	5, 6, 7, 8, 9, 10, 11	318
18	14	15	40	18	67	25	53	39	19	76	38		403

Table 2: Moves to complete level data

id	level1moves	level2moves	level3moves	level4movess	level5moves	level6moves	level7moves	level8moves	level9moves	level10moves	levelsSkipped	totalMoves
1	7	6	8	6	7	8	13	12	1	25		93
2	3	5	6	6	9	11	11	17	1	37		106
3	3	4	11	7	9	8	10	19	1	32		104
4	4	4	6	11	62	58	62	11	1	37		256
5	3	3	6	2	13	8	60	7	1	41		144
6	4	8	7	24	23	31	16	15	1	39		121
7	3	3	6	1	119	68	39	20	1	49		309
8	3	3	13	7	58	7	48	23	1	25		188
9	3	3	6	1	159	91	0	0	0	0		263
10	3	3	6	1	0	0	0	0	0	0	5, 6, 7, 8, 9, 10, 11	13
11	4	5	9	3	163	82	36	53	1	1	6, 8, 10, 11	357
12	4	4	7	11	109	22	144	107	1	264		673
13	3	4	11	1	6	16	19	11	56	41		168
14	3	4	10	36	149	14	36	30	1	18	5, 7, 8, 10, 11	301
15	3	6	7	3	19	11	22	1			5, 6, 7, 8, 9, 10, 11	72
16	3	4	13	6	134	63	41	1	1	14	6, 7, 8, 10	280
17	3	5	13	5	0	0	0	0	0	0	5, 6, 7, 8, 9, 10, 11	26
18	3	3	7	1	11	9	14	10	1	37		96

Riddle 1

- All dentists are old

- All French people are dentists

1. All French people are old

2. No French people are old

3. Some French people are not old

4. Don't know

Riddle 2

You're not allowed to drink before 18. You see four people in a bar: one with a coke, one with a beer, and a child and his grandfather (but you can't see what they're having). You can ask one person for ID, and one what they're drinking. Select the the person you would ID and the person who you would check what they are drinking.

1. Person with coke

2. Person with a beer

3. Child

4. Grandfather

5. Don't know

Riddle 3

- All men are normal people

- All men are wine drinkers

1. At least a part of the wine drinkers are normal people

2. None of the normal people are wine drinkers

3. None of the wine drinkers are normal people

4. Don't know

Riddle 4

Suppose I glue a number of playing cards back-to-back, with a face card on one side and a number on the other. I claim that I have glued every red face card to an odd-numbered card. You want to know if this is true. There are

four cards on the table and you can only turn over two cards. Which ones do you pick?

1. Ace of spades

2. Jack of hearts

3. Nine of diamonds

4. Six of hearts

5. Don't know

Table 3: Pre game questionnaire

A-Level Maths?	Degree contains?	I enjoy maths	I enjoy puzzle games	Riddle 1	Riddle 2	Riddle 2 Choice 1	Riddle 2 Choice 2
No	None of the above	4	4	All French people are old	Person with a beer		Child
No	None of the above	4	4	All French people are old	Person with a beer		Child
No	None of the above	4	4	All French people are old	Person with a beer		Child
No	Stats	2	4	All French people are old		Child	Person with a beer
No	Stats	2	3	All French people are old	Person with a beer		Grandfather
No	None of the above	2	2	All French people are old	Person with a beer		Grandfather
No	None of the above	2	3	All French people are old	Person with a beer		Child
No	Stats, Excel	2	4	All French people are old	Person with a beer		Child
No	None of the above	3	4	All French people are old	Person with a beer		Grandfather
No	None of the above	1	3	All French people are old	Person with a beer		Child
Yes	None of the above	4	4	All French people are old	Person with a beer		Child
Yes	Stats, Pure maths, Computer programming	4	5	All French people are old	Person with a beer		Child
Yes	None of the above	4	5	All French people are old	Person with a beer		Person with a coke
Yes	All of above	4	4	All French people are old	Person with a beer		Grandfather
Yes	Pure maths, Computer programming	3	2	All French people are old	Person with a beer		Child
Yes	Stats, Pure maths, Computer programming	5	5	All French people are old	Person with a beer		Child
Yes	Stats	5	5	All French people are old	Person with a beer		Child
Yes	All of above	5	4	All French people are old	Person with a beer		Child
Yes	All of above	5	5	All French people are old	Person with a beer		Child

Table 4: Post game questionnaire

I think I did well at the game	I enjoyed playing the game	I found the game engaging	I found the game difficult	I found the interface easy to use	The game ran smoothly with no issues	The game introduced concepts effectively	I learnt something new	Riddle 3	Riddle 4
4	4	5	3	4	5	5	4	1	1, 2
4	4	4	4	4	1	2	5	1	2, 3
4	4	4	3	4	2	5	4	1	2, 4
4	5	4	2	4	5	4	5	1	2, 3
4	5	5	3	4	5	5	4	1	2, 4
3	4	4	3	4	4	4	5	1	2, 3
3	4	2	5	4	4	4	4	1	2, 3
1	2	3	5	4	5	2	4	2	2, 3
1	2	2	5	3	4	4	3	1	1
2	3	4	3	2	2	4	4	1	2, 3
3	4	5	4	5	3	5	4	1	1, 2
1	2	4	4	4	4	4	4	1	2, 3
5	4	4	2	4	4	4	2	1	2, 3
4	5	5	2	4	4	5	1	1	2, 4
2	3	3	4	5	5	4	4	4	2, 3
2	4	4	5	5	2	4	4	1	1, 2
2	4	4	2	5	4	5	4	1	2, 3
4	4	5	1	4	1	4	1	1	5
4	4	5	3	5	5	5	4	2	4

72

UNIVERSITY OF BATH

Department of Computer Science

13-POINT ETHICS CHECK LIST

This document describes the 13 issues that need to be considered carefully before students or staff involve other people ("participants") for the collection of information as part of their project or research.

1. Have you prepared a briefing script for volunteers?

   Yes

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

   No

3. Is there any intentional deception of the participants?

   No

4. How will participants voluntarily give consent?

   Verbally

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

   No

6. Are you offering any incentive to the participants?

   No

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

   Yes

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

   No

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

   No

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

Yes

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

    Yes

12. Will the participants be debriefed?

    Yes

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

    Yes

NAME: James Treasure

SUPERVISOR : Dr. Willem Heijltjes

SECOND READER : Dr Christian Richardt

PROJECT TITLE: The Development of a Serious Game to Teach Aristotle's Syllogisms

DATE: 26/04/2017