

Using Gamification to Create an Educational Tool based on a Natural Deduction Proof System

Submitted by: James Farthing

COPYRIGHT

Attention is drawn to the fact that copyright of this dissertation rests with its author. The Intellectual Property Rights of the products produced as part of the project belong to the author unless otherwise specified below, in accordance with the University of Bath's policy on intellectual property (see <http://www.bath.ac.uk/ordinances/22.pdf>).

This copy of the dissertation has been supplied on condition that anyone who consults it is understood to recognise that its copyright rests with its author and that no quotation from the dissertation and no information derived from it may be published without the prior written consent of the author.

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:

Abstract

This project explores how the Natural Deduction proof system is currently taught and how this system can be integrated into a game. Current literature is examined to find out what traits in games can be effective in an educational tool. This project aims to teach Natural Deduction using the principles of Gamification. This allows a user to learn in an alternate way to which they are used to. The author has created software in which Natural Deduction has been incorporated into a game. Feedback from the software has been gathered and positive results were achieved for the educational effectiveness of a Natural Deduction game.

Using Gamification to Create an Educational Tool
based on a Natural Deduction Proof System

James Farthing

Bachelor of Science in Computer Science and Mathematics with
Honours
The University of Bath
April 2016

Contents

1	Introduction	3
2	Literature Review	7
2.1	Gamification	7
2.1.1	Uses of Gamification	8
2.1.2	Advantages	9
2.1.3	Current Educational Games	12
2.2	Education	13
2.2.1	Different learning styles	13
2.2.2	Current Natural Deduction Educational Resources . .	14
2.2.3	Online Natural Deduction Resources	16
2.3	Natural Deduction	16
2.3.1	History	17
2.3.2	Why using Natural Deduction is Suitable to Make a Game for	18
2.3.3	Mathematics of Intuitionistic Logic	19
2.4	Notes on the Literature	22
3	Requirements Analysis and Specification	24
3.1	Target Audience	24
3.2	Project Goals and Targets	25
3.3	Software Requirements	26
3.3.1	Functional Requirements	26
3.3.2	Non Functional Requirements	28
3.4	Hardware Requirements	29
3.5	Programming Language	30
3.6	Project Limitations	32
3.7	Changes to Existing Systems	32
4	Design	34
4.1	Data Design	34
4.2	Data Flow Diagram	36
4.3	Architecture Design	38

4.3.1	File Structure	38
4.3.2	Key Functions	38
4.4	Interface Design	41
4.4.1	Design Principles	41
4.4.2	User Interface Designs	42
5	Implementation	47
5.1	Beginning the project	47
5.2	Drag and Drop	48
5.2.1	Original Colour System	49
5.2.2	The Dots System	51
5.3	Data Structures	53
5.4	Drawing and rendering algorithms	54
5.5	The Game Play	56
5.6	Design Implementation	58
6	System Testing	60
6.1	Developer Testing	60
6.2	Functional Testing	61
6.2.1	Unit Testing	61
6.2.2	Acceptance Testing	62
6.3	Non Functional & Hardware Testing	63
6.3.1	Stress Testing	63
6.3.2	Security Testing	64
6.3.3	Compatibility Testing	65
6.3.4	Other Hardware Comments	65
7	Results	66
7.1	User Questionnaire & Feedback	66
7.1.1	Age and Mathematical ability	67
7.1.2	Prior Natural Deduction Knowledge	67
7.1.3	Tutorial	68
7.1.4	Hints and Tips	69
7.1.5	Levels and Difficulty	70
7.1.6	Fun	71
7.1.7	Educational Value	72
7.1.8	Usability	73
7.2	Improvements to the game	74
7.3	Further Work	75
8	Conclusion	77
A	User Testing Results	83

Chapter 1

Introduction

Natural Deduction is a logical proof system that has been in existence over 80 years, founded by Gerhard Gentzen, and has been a staple in the teaching of Mathematical Logic for the last half century. Natural Deduction is a proof system which is made up of basic logical rules with the goal of proving that some reasoning is correct. Laboreo (2005) Natural Deduction is commonplace in current undergraduate degrees in Mathematics and Computer Science. Teaching logic has traditionally been delivered in a formal format such as a lecture or a classroom session but other ways of teaching can also be effective. The advancement of technology over the last 20 years offers alternative teaching options. The increased usage of computers has led to a greater number of resources accessed via the internet that are available to learn from. Methods of teaching using a computer can be utilised to provide more varied ways of learning.

In this project an educational game will be created for Natural Deduction. The software will be tested on users to see if it would be useful for learning. This project investigates Natural Deduction in an educational environment and aims to help bridge the gap between teaching Natural Deduction and playing games. The software created in this project explores techniques that can be used in games to teach users. It also puts the Natural Deduction proof system in a more accessible and engaging environment than traditional learning. The vision for this project is to take Natural Deduction and try to make it easier for older secondary school and university students to understand. The literature explores whether gamification is an effective way of doing this. The overarching aim is to engage students in a new and different way.

One motivation for this project is the author's interest in gaming and how principles from games can be applied to an educational context to help students and other interested individuals to learn. Literature discussed in

Sections 2.1 and 2.2 suggests that games can be used to help people learn new concepts. This provides evidence that using educational concepts in games can be effective.

Games are popular in the everyday life of young people (16-21 year olds), which is the age group this project is aimed at. Mobile phones, tablets and computers are all capable of playing games which people play on a daily basis. Creating a game to be played on a digital format such as a computer means the game will be easily accessible to users. It will also be located on a platform which the majority of the target audience are familiar with.

Gamification is the process of taking a task that is not usually associated with gaming and turning it into a game. Gamification is already used in a variety of situations such as in Industry and Marketing sectors. These are described further in Section 2.1. Gamification principles will turn the task of teaching Natural Deduction into a game. Using motivational gaming elements in this software such as hints and instant feedback (Section 2.1.2) will provide a different and more entertaining Natural Deduction learning resource.

Some existing software already effectively uses gamification and replicating their best features will help deliver a quality project. Current educational games offer in-game rewards and a number of different levels of increasing difficulty. Integrating these elements into this software will be necessary to make it effective.

People in education benefit from different learning styles. Not every child learns in the same way so using unconventional methods such as games to help people learn may be effective for some pupils. This project aims to create software that users feel has benefited them educationally, whilst still enjoying themselves. Gamification for Education is built on the premise that the concept of a game motivates users to play. Researching key educational principles that need to be included in such a game is an important step to make sure the game teaches users as well as letting them have fun. How Gamification can be used for education and an explanation of learning techniques is detailed in Section 2.2.

What makes this project unique is its approach to teaching Natural Deduction. The software uses Intuitionistic Logic which is natural to understand and easy to learn.

Logic is rearranging arguments in a way which we desire. Mathematical logic represents this in a formal way. Assumptions are made and from these conclusions are drawn, creating a rule. These rules are represented in a tree structure where assumptions are on the top and the conclusions on the bottom. Rules can be combined so that conclusion from one rule

can become the new assumption for the next rule. This builds up the tree structure.

In this project, the user will create this tree structure and input all of the rules needed to create the proof. Representing in a tree format teaches the user not only how to form Natural Deduction proofs in the Gentzen style but also learning about intuitionistic logic. Common rules that exist for intuitionistic logic are “and”, “or” and “implies” and they work in a way which you would expect. Using these rules and helping to teach them in a mathematical context is central to this project as it allows the user to learn the logic system whilst using Natural Deduction proof system to create the proofs.

Once literature has been examined, the system development life cycle needs to be implemented to create a successful piece of software. This starts with the analysis gathering and the requirements setting. Whilst this project doesn’t have a customer in the traditional sense, a target audience is specified in Section 3.1 for where this project is positioned in the wider scope of educational games. The student target audience will make sure that the game is helpful and has a genuine use.

The goal setting process also needs to be carefully contemplated. Frequently in software products, the end user can give you detailed requirements for what they want delivered. This project is more of an investigative process in whether the software created can be useful in the future. Targets and requirements need to be formulated from the literature, drawing on the experience of similar educational games (Section 2.1.3) and from other key features examined in Section 2.1.

All of these requirements need to be present in the design of the system. The design should take all of these required features into account. Section 4.1 explains how data in the software will be stored and how it will be used in the game. The flow of how data moves round the game is also displayed, designing the groundwork in how the program will function. The architecture is then discussed at a high level in Section 4.3. The main functions are focused on and how each function links to achieving the requirements. Designing the interface is also an important step for any project which needs a user interface. Designing how the program will look and how users will interact with it is a key part of the design. Interface designs are shown in Section 4.4.2.

Next, the process of how the project was developed and the reason why certain decisions were chosen is outlined. The implementation of the project describes key features included in the game. Also explained is some of the changes made throughout the software development process and explanations for the change of direction for elements of the game. Parts mentioned

include how the drag and drop works, how the data is represented and how proofs are rendered on the screen. Implementation details can be found in Chapter 5.

The investigation in this project is finding out whether this software has the potential and is effective in teaching users about Natural Deduction through the medium of a game. A user survey was distributed to test an Alpha version of the software. Feedback was encouraging and suggests that further time should be invested in the project to make small improvements to make it into a more complete software project. Detailed results are analysed in Chapter 7.

The majority of users felt that this game delivered as an effective educational tool. Most of those surveyed also didn't understand Natural Deduction before playing the game and after playing the game, felt that they had learnt more about the topic. The testing suggests that this game is a good tool for learning and that people also think that it is more effective learning through the medium of this game rather than learning in a classroom. These positive results show the further potential for a Natural Deduction game and how people can enjoy it whilst still learning from it.

Chapter 2

Literature Review

2.1 Gamification

Gamification is the idea of making a task or situation use game mechanics that does not naturally use this context. It is theoretically possible to make many tasks into a game using game-like concepts Muntean (2011) and this is one of the reasons why gamification has become popular in a wide variety of topic areas.

It is important to understand the characteristics of a game to fully understand how a task can take on game like properties. A game is defined by a system in which players engage in abstract challenge, defined by rules, interactivity, and feedback, that results in a quantifiable outcome often eliciting an emotional reaction. Koster (2013) Gamification of this in an educational context would be to set rules for a user to follow, challenge them with different Natural Deduction concepts and give them constructive, instant feedback in the task which is not usually presented in this format.

There are many general benefits to gamification and why video games can be effective learning tools. Offering support to a user at the appropriate times can be helpful. A game providing information when the user needs it is much more useful than information that is unnecessarily given at the wrong time. Rewarding users giving them a sense of achievement is also something that games can naturally providing and can easily be offered in game. Balancing difficulty and having a correct difficulty curve gives a game a challenging yet achievable structure. Getting the correct balance of challenge and difficulty will keep the motivation of the users high and will drive them further on in the game. González & Area (2013)

2.1.1 Uses of Gamification

Gamification has been used in a variety of different situations in a number of industries and environments. Whilst it is important to focus on how gamification has been adapted and used for educational purposes, it has to be noted which other areas have used gamification to their advantage and what key concepts they have used to make gaming a successful platform for their product or service.

Marketing by companies has been a successful way of using gamification Zichermann & Cunningham (2011). They have been able to engage users in new ways by providing apps and websites with games and activities on and consequently consumers are spending more time around that brand. It is important to harness this way of being able to interest people using gaming principles when trying to teach my educational concepts. I think that people can enjoy and learn more effectively if they are engaged in the task they are doing. Games are a good way of achieving this.

Industry has also used gamification. Studies have investigated whether gaming concepts can increase motivation and morale in the workforce, leading to higher productivity. An article in *Frontiers in psychology* was positive stated in conclusion that there's "emerging base of evidence that suggests gamification as a promising strategy for promoting loyalty, productivity, and wellbeing in the workplace" Oprescu et al. (2014).

Whilst education and industry are two different things, there are common qualities and skills that are required in both areas. Increasing motivation to learn a particular mathematical formula, for example, could be very beneficial for some students. A higher rate of productivity would also be advantageous, as students often have many tasks ongoing at any one time. An increase in these skills would overall increase the enjoyment of learning and reinforce understanding so students can get a better handle on the key ideas.

Gamification has also been embraced by the educational sector. This is by far the most important sector for this project. It is important to understand how gamification has helps improve education and assists students in learning new material. The advantages of gamification and how these advantages can be used to benefit an educational tool will sculpture the direction that this project takes. Ideas discussed will be taken forward and help formulate requirements for the project.

2.1.2 Advantages

An abundance of educational games exist on the Internet. Educational games have many benefits that will help students learn. Instant feedback, social aspects, competitiveness, games being fun and giving the user hints and tips are all advantages that make games a better way of learning than other formats. All of these features have different reasons why they are effective in games and transferring these qualities into an educational resource will add value to it in a way that traditional educational resources do not.

2.1.2.1 Instant Feedback

Games offer instant feedback; the user can immediately know when they have done something right or wrong. This is a big advantage with games, it can offer advice and tips much quicker than a teacher can. The game in this project needs to use this feature to its advantage, because giving someone instant feedback can be rewarding and be much more helpful than waiting weeks for feedback, for example, after handing in a question sheet.

Research has been undertaken to examine whether giving students instant feedback has improved learning performance Wu et al. (2012). This study, which was published in the British Journal of Educational Technology, found that the enhanced way of presenting their concept maps which gave the students instant feedback “significantly improved the learning achievements of the students”. They recognised that giving instant feedback meant that the topic was fresh in the students’ minds and therefore could organise, collate and adapt to the information that was given in the feedback.

The paper also suggested that students which used the new system “gave significantly higher ratings” than those who used the standard approach. This questionnaire was asking about learning attitudes towards the course that they were undertaking. From this information, it can be deduced that the students who were questioned seemed to be enjoying their course more due to the enhanced system that was introduced in this study. It is important to recognise that instant feedback was not the only improvement to the enhanced system they were using in the study, but this was a key part of the new system, so must have contributed at least partly to the students’ improved learning performance and improved learning attitudes.

Instant feedback can be key motivation for the project, given that it can improve both student enjoyment and performance. As mentioned by Dempsey et al, instant feedback can more effective than delayed feedback when using “classroom quizzes and (other) materials” Dempsey et al. (1993). He goes

on to conclude that “immediate feedback should be prescribed unless the feedback is delayed systematically for a specialised purpose”.

It is crucial to bear this in mind when creating the game that giving a user instant feedback can be a powerful tool. This style of feedback shows that it will increase the learning capability of students. This project should take advantage of that and introduce the correct mathematical notation to the user. If the formal style of notation is included then students can start to learn how to use a Natural Deduction proof system with intuitionistic logic concepts straight away. If the game was too abstract, with no relation to the mathematical representation, part of the learning process would be lost when skills had to be transferred over to the classroom environment.

One way of offering this constructive feedback would be to offer a scoring system. You could get a certain number of points for getting parts of a question correct and bonus points for flawless levels completed. This would instantly tell the player of the game whether they were doing well or not. Players could repeat plays of the game to get higher scores which would increase understanding. This would be because doing a repetitive action multiple times reinforces your learning.

2.1.2.2 Social and Competitive Aspects

Another advantage to Gamification is the ability to offer a social or competitive aspect of your subject matter. This could be through online leaderboards displaying top scores, multiplayer aspects or social media connectivity. Keeping fellow users of the game connected in some way can be important for increasing enjoyment of gameplay and also the amount of time they spend playing the game.

In the current age where 90% of 16-24 year olds in the UK have smart phones, and being used for at least two hours a day, it can be said that people find that communicating via online methods important Ofcom (2015). In the same article, it is said that “around half (49%) of young people aged 18-24 check their phones within five minutes of waking up”. This just backs up the point that social interaction and social media via online devices is in the forefront of people’s minds throughout their lives. By incorporating this into the software and offering some kind of social aspect, people will enjoy it more.

A study undertaken which has looked into how to create engaging games in the health care sector discovered that communication was important for 13 to 16 year olds, which was their target market Suhonen et al. (2008). It also discovered that being in touch with others and interacting with them while playing games was also important to them. The report states that

“The company of friends motivates to play games and makes gaming fun”. These are two massively important points that should be not underestimated when creating my own game. Being able to socialize whilst playing a game may increase participation and fun. Social media may be a good way to implement this. Students can also talk to each and discuss ideas about what is the correct way to go about solving a problem.

2.1.2.3 Fun and Enjoyment of Games

Games being fun and engaging is one of the main reasons that people want to make a variety of different tasks into games. If people will enjoy what they are doing more when they are playing games, then games are a viable solution for all sorts of media, including educational resources. A game being fun will capture an audience more and they will be more likely to play again because of the enjoyment they have got whilst playing it.

Research published by The UK Interactive Entertainment Association stated that the UK games market is worth £3.944 billion in 2014 UKIE (2014). This much money would not have been spent on games if consumers did not enjoy playing games. As mentioned by Rosemary Garris, Robert Ahlers and James E. Driskell, users make a clear cut choice to whether they are enjoying a game based on their own feelings Garris et al. (2002).

2.1.2.4 Hints and Tips

Another advantage of gamifying education is the ability to be able to give users hints and tips throughout. Students learning a topic can often get stuck, and sometimes with nowhere to turn, may give up and try again another time. What interactive games can offer to educational topics that more traditional methods can not offer is that they can give the player hints and tips at that moment whilst playing. If a player is struggling with a particular topic, a prompt could be the only motivation a student needs to carry on. This hint could help the student continue working through the problem instead of giving up. This can aid further learning and a longer time will be spent playing the game because the user will be able to complete the levels.

Chen discussed this in his 2009 paper on giving users prompts to online learners Chen, Wei, Wu & Uden (2009). He found that “the main factor affecting reflection levels is high levels prompts”. He went on to say that “We believe that effective reflection can be achieved by reflection prompts”. This means that prompts can be an effective tool for promoting learning performance through a process of reflection.

2.1.3 Current Educational Games

MyMaths is a popular subscription website that offers a number of mathematical puzzles, challenges and games for a student to work through. *My-Maths Website* (2016) From personal experience of using this platform when I was in secondary school, it was a much more exciting way of experiencing mathematics because it was more interactive than just completing exam style questions. It was understanding mathematics in a completely different way because of the innovative way it was taught. The key element to this platform being successful is that it is fun. It keeps students engaged by making a more fun way to learn.

A reason why I thought MyMaths was a good interactive resource is that it was available anywhere there was a computer and internet connection, which are becoming more easily available resources as connectivity improves. This means that the tasks on the website could be completed during dedicated lesson times, but also could be done during lunchtimes or at home. It was accessible to all and this will be one of the advantages of creating an online game.

A highly successful website that targets a wider audience to teach them how to code is Codecademy. Sims & Bubinski (2013) Codecademy is a website that provides the ability to learn the basics of certain programming languages. It does this by creating a variety of different levels with goals at every stage of the process. Some of the levels can contribute to the creation of a larger program. Using bite size chunks of code as levels is an idea that could be incorporated into this project. This splits the task into manageable pieces and makes it easier to understand.

Another concept that Codecademy uses is badges for completing levels, challenges and for having a streak of days where you complete challenges. Receiving badges makes the user gain a sense of achievement and will motivate them to continue with the website. This would be something that could be considered for the project, as it encourages people to continue playing your game due to the motivation in receiving rewards for completing levels.

Codecademy makes levels progressively harder with harder coding concepts as the user goes through the levels. The user needs to be eased in to very basic concepts so they will continue playing the game through the levels and not give up. Another reason to introduce the basic concepts first is that it is common that basic concepts will need to be used in conjunction with the more complicated topics, which can then be introduced in later levels.

2.2 Education

Learning is a process that comes naturally throughout life. Growing up, many skills are learnt without even realising we are learning them. When students are in education, they are consciously trying to learn new things to benefit themselves and to improve themselves. Learning could be defined as "A change in behaviour as a result of experience or practice" amongst other things. Pritchard (2013) Trying to learn things in the best way possible is our goal and there of many different ways of learning material. Similarly there are many different ways of teaching material. In this section some advantages of certain learning and teaching methods will be outlined. Gathering these different learning and teaching techniques will hopefully lead to the creation a more valuable game which can be used as an effective learning resource.

As summarised in Simulation and Gaming, games are perceived as more interesting than other ways of delivering material Garris et al. (2002). In a study by Cohen, also mentioned in Simulation and Gaming, he found that 87% of students were more interested in educational games than other classroom approaches. This highlights a want for educational games by students and an enthusiasm that might not be there for other approaches to teaching.

2.2.1 Different learning styles

Every individual is different in which way they learn best and it would be rare that a person can only learn from one style. The different ways of learning below are broadly the main ways people can take in and understand information.

- Visual Learning

Visual learners like pictures and graphs. They would prefer to see something written down in front of them rather than being said to them. Maps and other visual learning tools can also be effective for these kind of learners.

- Auditory Learners

Auditory learners learn best by listening. They can absorb information well from lectures and reading out loud to themselves. Good techniques for Auditory learners would be to record themselves saying information or hearing the information being said by others.

- Kinaesthetic Learners

Kinaesthetic learners learn best by being active. They like to touch and feel objects as well as doing examples and practising. If information is presented to them they learn best by writing it down. A game where they were playing it would suit them well.

McDonald & Hull (2012) Montemayor et al. (2009)

From these different learning styles, we can already see that different people learn in different ways. Kinaesthetic learners would benefit the most from playing the game and this should be the audience that should be targeted. Getting users to get interactive with the game naturally aligns itself with Kinaesthetic Learning. From personal experience in the UK educational system, different learning styles are used at different ages. At university level, very little learning is Kinaesthetic because of a lecture format which suits Visual and Auditory learners more. In a classroom, students are encouraged to practice exercises in the class. This is an example of Kinaesthetic learning and university students are encouraged to do this kind of learning in their own time.

By making a Natural Deduction tool that helps users learn Intuitionistic Logic, it will make Kinaesthetic learning methods more accessible and more entertaining. This will be because of the gamification elements previously discussed that will capture the attention of the user more. This should encourage university students to spend their own time learning through the game and hopefully benefit from a different teaching approach. Using the Natural Deduction proof system again and again means any user should feel comfortable with how it works and its structure by playing levels of the game. The repetitive nature of the rules means that every novice user is capable of learning and benefiting from using the software which will be created. Gamification elements that help guide the user should make them also feel comfortable with the logic rules by playing the game.

2.2.2 Current Natural Deduction Educational Resources

Natural Deduction has been taught in schools and universities for a long period now. With the more recent creation of the internet, there are now a number of different resources online which teach people the basics of Natural Deduction. All of them teach it in slightly different ways and each method of teaching the material will benefit a different group of learners better. In the game, the aim is to try and grasp some of the better characteristics of these ways of learning material and include them all within the game.

- Notes

Notes for Natural Deduction are widely available online written by

different groups of people. Enthusiasts and academics all have materials available which help for different levels of understanding Natural Deduction. One of the main advantages of notes are that they are clearly laid out and organised. Many sets of notes are written in LaTeX which gives them a professional and good aesthetic appearance. Laboreo (2005) Notes will be effective for visual learners who can see things written down and learn from them. One of the main advantages that I would like to incorporate into my game would be having easily displayed information available to view and also to print off for reference.

- Lectures

Lectures can be an invaluable resource for auditory learners. Academics in their field can really convey information effectively and explain it well because they are experts in their field. Lectures typically include a visual aid, which could be a slide show or writing on a blackboard to help students understand the concepts. npTELhrd (2015) This will benefit visual learners too, as they can see what is being explained. The important thing about lectures that could be included in the game is the audio. Audio is important to develop understanding and to explain concepts in a different way to the visual material.

- Videos

Online videos exist for teaching Natural Deduction, such on websites like YouTube. PhilHelper (2013) These provide a different style of teaching than online tutorials which are typically just powerpoint slides. The audio over the top of them explaining what is happening gives videos an extra dimension to the teaching experience. This particular video, uploaded by the user PhilHelper on YouTube, uses graphics and examples to clarify some of the mathematical points. The mathematical notation is written next to the example, which keeps the mathematical notation in the forefront of the viewer's mind. This will appeal to visual learners. He is also speaking through what is on screen, so will benefit auditory viewers also. Some of the materials that he has in his videos would be effective in the project between levels of the game, to reinforce the learning.

From these common techniques of delivering material, visual and auditory learners are well catered for through Notes and Lectures, as well as Videos catering for both. Kinaesthetic learners may miss out somewhat by these traditional ways of learning. Creating software that will benefit Kinaesthetic learners may fill this current gap in educational resources.

2.2.3 Online Natural Deduction Resources

Whilst not strictly an educational resource, Automated Theorem Provers are computer driven systems that can automatically work out a proof tree from a given end point Sutcliffe (2009). These have their advantages as they automatically create a proof, so once the theory has been learnt by an individual, it can be checked by the automated theorem prover. What this project wants to achieve would be to have an automated theorem prover which can guide a user through creating a proof for themselves. The game will have all of the logic ingrained in the code, but the objective is to make the user learn as they go along.

One similar project that has been undertaken is a Proof Assistant for Natural Deduction logic Gasquet et al. (2011). This tool has many good points that are worth highlighting. The notation is in keeping for what will be included in this project. It is also a very interactive system that will definitely benefit kinaesthetic learners. Things that will be focused on in this project is the aspect of becoming a game. Utilising levelling systems, fun and giving hints and tips to benefit the user. Additional features of some sort of social aspect would also improve engagement in the game.

By creating this project, it aims to target this type of learner through the interactive process of getting the player to 'do' things. If Natural Deduction can be taught effectively as a game, all the different learning types will have some benefit from playing, but especially kinaesthetic learners.

2.3 Natural Deduction

Natural deduction is a mathematical logic system. It is a system for proving statements of formal logic. It is used to show if certain statements are valid and to prove them. Pelletier (1999) Natural Deduction has a typical mathematical notation system that could create a barrier between the majority of general public understand how Natural Deduction works. Through this project, I hope to break down these barriers and make Natural Deduction more accessible to a wider range of people.

Natural Deduction is used in a way that a large problem that needs to be proved is broken down into singular steps which are easily provable in their own right. Each individual step is proved, and then that information can be used going forward in the proof. Halbach (2014) The small parts of the proof are verified and their validity means that can then be assumed in later parts of the proof. These small steps can then contribute to a larger Natural Deduction proof which is not trivial at the start. This lends itself well to levelling up system that will be implemented in the project, where the user

could complete easier tasks in early levels and end up joining all of their findings together to prove a more complex theorem.

It is important to make the distinction between Natural Deduction as a proof system and the types of logic that can be used with it. Natural Deduction is the system used for the logic to prove a statement and applying proof rules derived from conclusions of other rules is part of the Natural Deduction system. Many types of logic can be represented by Natural Deduction such as Classical Logic, Intuitionistic logic and others.

Other proof systems are available. The main other type of proof systems are Axiomatic systems where basic assumptions are axioms that are used and theorems are defined from the axioms. The Frege System is an example of an axiomatic proof system. Natural Deduction systems- as in the name- represents a more common sense way of describing logic. These systems have no axioms and lots of rules instead. Smith (2010)

Intuitionistic Logic is the logic that will be the logic system that will be used in this project. It is a formally defined set of rules that is naturally suited to be displayed using the Natural Deduction proof system. Van Dalen (1986) Other logic systems could be displayed using Natural Deduction such as Classical logic with slightly different rules. One of the most accessible logic systems to teach is Intuitionistic Logic and was the main reason it was picked for this project.

2.3.1 History

Natural Deduction as an idea formally came to be in 1934/35 when Gerhard Gentzen and Stanisław Jaśkowski both produced papers independently of each other. Jaśkowski (1934) Gentzen (1964) No content had been published on this topic before and this led to the start of what is now known as the Natural Deduction Logic system. Both papers published in the same year had a very similar topic area of taking basic logic as a given- assumed to be true- and then creating a method of exploring what can be done with these basic axioms to prove more complicated theorems. Pelletier (1999)

Whilst this was the true start of where the ideas of Natural Deduction came from, it did not pick up widespread appeal until educational textbooks begun to publish this logical system in the 1950s. Pelletier (1999) Natural Deduction has not changed greatly since the original concepts of Gentzen. Many proofs are solved using this proof system throughout all of Computer Science and usually first taught during an Undergraduate degree. Many logic systems are used with Natural Deduction including Lambda Calculus of which I have had personal experience with.

Natural Deduction is not only important for solving practical proofs. Over the 1960's and even continuing now, Natural Deduction has been used as a theoretical tool, and lots of research has gone into it to improve how Natural Deduction is applied. The aim of this is to make practical Natural Deduction problems easier to solve and quicker, possibly using computing resources to do so. The way Natural Deduction proofs are used has expanded, partly due to the work of Prawitz and RagGIO on Normal Proofs. Indrzejczak (2016) More work continues to be done on many streams of Natural Deduction today to improve the way people use the logic system.

2.3.2 Why using Natural Deduction is Suitable to Make a Game for

Natural Deduction is a logical proof system which is thought of as closely aligned to people's natural reasoning patterns. Arthur (2011) It should, in theory, be easy to teach a system that is closely related to people's intuitive reasoning patterns. In reality, it might be that the mathematical notation could be one of the main things standing in the way of people learning this logical system. By creating something that is easier to grasp and understand, users should be able to connect with their intuition once again and understand the concepts behind Natural Deduction.

Having being taught since the 1950s, this way of teaching logic has been one of the primary methods for many years now. Its ongoing reputation as a primary way of teaching mathematical logic in both mathematics and philosophy means that it is still very popular today. An easily accessible way to learn this new 'language' for the first time can only be beneficial in my opinion.

Natural Deduction is a core system of logic that has been used regularly over the last fifty years. This importance has been shown through the number of educational resources produced and the amount of material that has been written on the topic of Natural Deduction. The lacking of a fun, easy way to learn Natural Deduction is a shame and is why I feel it is necessary to create a game based on this logical system.

This project provides academic merit because of the alternative way that Natural Deduction logic will be taught in. Whilst primarily benefiting users and students who use the software for their own educational benefit, many academic experts and educational leaders in logic fields should find this paper interesting due to different ways of conveying traditional logical information to people who are not familiar with the concepts.

2.3.3 Mathematics of Intuitionistic Logic

This section will cover the basics of mathematics that will be implemented into the project. Most textbooks use similar notation but Fitch notation written in lines is sometimes used. For the purpose of this project, tree-like presentations will be used.

2.3.3.1 Notation

- And (Conjunction) \wedge

A and B both have to be true for $A \wedge B$ to be true.

- Or (Disjunction) \vee

Either A has to be true or B has to be true or both for $A \vee B$ to be true.

- Implies \Rightarrow

The statement $A \Rightarrow B$ says that if A is true, then B is true. Similarly you deduce A is true if you know B is true which would be written as $B \Rightarrow A$

- For all \forall

If all elements in a set or in the domain of a function holds a certain property then the for all symbol can be used appropriately. $S = \{1, 2, 3\} \forall x \in S, x < 4$.

- There exists \exists

This means that at least one item in a particular set or domain exists and holds true. $S = \{1, 4, 9\} \exists x \in S, \text{s.t } x > 7$.

- If and only if \Leftrightarrow

$A \Leftrightarrow B$ means that if we know that A is true then B is true also holds. This also means that if B is true then we can deduce A is true.

- Not \neg

The opposite of being true. A being false implies $\neg A$ is true.

Laboreo (2005)

2.3.3.2 Rules

Everything on the top line is assumed to be true, with each statement separated by a comma. What is deduced is below the line. In proofs, there will be multiple lines on top of each other.

Conjunction introduction Introducing the 'and' symbol into the proof means that given two things that are deemed to be true, we can conclude both together are true. In formal mathematical notation, it would be written as shown in Equation 2.1 below.

$$\frac{A \quad B}{A \wedge B} \wedge I \quad (2.1)$$

A real life example of this would go as follows:

1. It is sunny.
2. It is hot.
3. Therefore it can be concluded: It is sunny and hot.

Conjunction elimination Whilst the introduction takes two pieces of separate pieces of information and makes it into one new piece of information, conjunction elimination takes one piece of already assumed information and from this we can deduce two separate pieces of information. Equations 2.2 and 2.3 show what can be achieved.

$$\frac{A \wedge B}{A} \wedge E1 \quad (2.2)$$

$$\frac{A \wedge B}{B} \wedge E2 \quad (2.3)$$

Similarly to the previous example we use conjunction, but instead of deducing the conjunction, we already have it, so we can deduce the individual components.

1. It is sunny and hot.
2. Therefore it can be concluded: It is sunny.
3. It can also be concluded: It is hot.

Implication introduction This takes two statements. The first of which is assumed to be true. Now if we can deduce from this another element is true then trivially one must imply the other.

$$\frac{[x : A] \quad B}{A \rightarrow B} \rightarrow I/x \quad (2.4)$$

Implication elimination (Modus ponens) In implication elimination we are given an implication and (at least) one other piece of information and from this we can eliminate the implication to conclude that another element is also true. In Equation 2.4 below, we see that $A \rightarrow B$ is true but this is not enough to conclude that A or B is individually true. By being given the extra information that A is true, we can then use this rule to deduce B must also be true.

$$\frac{A \rightarrow B \quad A}{B} \rightarrow E \quad (2.5)$$

A practical example of this rule being used may be clearer to understand:

1. James eating chocolate causes James to be happy.
2. James is eating chocolate.
3. It can be concluded: James is happy.

In relation to Equation 2.4 above, James eating chocolate is A and James being happy is B. When the fact is given that James *is* eating chocolate, then it can be concluded that James is happy and this is true on its own because of the other information we have.

Disjunction introduction This introduces the 'or' symbol into an equation based on the fact that one of the variables is true. This is because only one of the variables need to hold true for the disjunction to hold true. This is shown in Equation 2.5 and then an example is given in text.

$$\frac{A}{A \vee B} \vee I \quad (2.6)$$

1. Football is a sport.

2. Football is a sport or A Lemon is a sport.

The second variable does not have to be true because the first variable, in this case 'Football', makes the statement hold true. The disjunction would still hold true if both the variables held true.

Disjunction elimination This rule is very slightly more complex and gives a flavour of what complexity is possible with natural deduction proofs. This is still a simple rule and this list is not exhaustive of rules. Disjunction elimination manages to remove the disjunction because of other information given. There are two implication statements, both implying the same thing. Both these implications hold true. There is then a third true statement, which is a disjunction between the first variables of each implication. Because of the disjunction, it holds that at least one of the variables is true, meaning that the third variable that is implied is true. We conclude that the implied variable is true. Displayed in Equation 2.6 is a much simpler way to explain the rule using formal mathematical notation.

$$\frac{A \rightarrow C, B \rightarrow C, A \vee B}{C} \vee E \quad (2.7)$$

An even easier way to understand this is to give a simple example.

1. James eating chocolate causes James to be happy.
2. James eating pizza causes James to be happy.
3. James is eating chocolate or James is eating pizza (*or both*).
4. It can be concluded: James is happy.

This is exactly the same to what is displayed above but using examples instead of complicated mathematical notation.

Halbach (2014) Laboreo (2005) Indrzejczak (2016)

2.4 Notes on the Literature

The three sections in this literature review: Gamification, Education and Natural Deduction have all touched on areas that will be brought together in this project.

The benefits of games and what makes people play them have been discussed. The advantages that a game could bring to an educational resource has also

been researched in detail. Current games that are successful in the real world show what games can do if created in the correct way. I conclude that a game can be a valuable resource for educational purposes where people can learn better. This, in my opinion, justifies why a game should be made of Natural Deduction.

Education, the second section, talked about different learning styles. There is a lack of kinaesthetic leaning in mainstream Mathematics and this is exactly what a game can provide. I looked at similar resources and projects to my own and how I feel they can be improved upon. Due to the lack of similar, helpful resources there is definitely a need for this kind of online platform that can help educate.

Finally, an introduction to Natural Deduction gave a background to the history and to reasons why Natural Deduction should be used to make a game. Natural deduction is still one of the core logic concepts and it is important to make sure people know about it.

All these things lead me to believe that a software project in this subject area would be extremely worthwhile which will have value to many different groups. Academics and Students alike should find benefit in a resource of this kind. This project, whilst mainly appealing to Mathematicians due to the content, should also appeal to other key departments such as Computer Science where Natural deduction is used extensively. Philosophers should also find this project interesting due to the ongoing research into whether gaming is an effective teaching resource.

Chapter 3

Requirements Analysis and Specification

3.1 Target Audience

It is important to recognise that the difficulty of game that will be produced will depend on what audience the game will be targeted at. A game for academics or for postgraduate students would have to be far more technical and complete in Natural Deduction theory for solving mathematical proofs than a game which would be made for undergraduate students or secondary school children, for example.

When deciding on what would be the suitable age group for the educational game, considering the age that someone would learn Natural Deduction logic highly influenced the process of picking the target audience. Whist most people are not aware of Natural Deduction before 18 years of age during Secondary Education, most people that are introduced to First Order Logic and Natural Deduction proof systems are taught about it during an Undergraduate Mathematics or Computer Science Degree. Other Students from other degree programmes are unlikely to learn the concept of Natural Deduction and therefore would also be suitable to play this game.

A game that introduces Natural Deduction concepts at a stage in life around the time you learn it formally in education seemed like the most favourable option. The Sixth Form/Collage and Undergraduate University age (16-21) is the age that a different style of learning may be most beneficial. Coming from a different point of view and using different techniques may be effective for some students. This project would be targeting people with A Level knowledge of Mathematics and some going on to study university mathematics, but all with a mathematical background.

At Sixth Form age, education is delivered traditionally in a School environment which can be very different to University education. At University, the basics get taught but students have to research the concepts in further detail and try examples independently. This is why this game could also be useful for University students where they can practise the logic concepts and become familiar with Natural Deduction by using the game.

A tool that tries to bridge this gap by giving you basic examples of Intuitionistic logic using the Natural Deduction proof system should help in making practise easier and therefore helps you to learn quicker. This game will provide a more entertaining way to learn than traditional methods of learning. This target age group of around 16-21 years old spends a lot of time on computers and mobile devices. A 5-15 year old spends an average of 8.7 hours a week gaming, Collett (2013) so using computers and gamification to help students learn could be very effective, as on computers is where a lot of time is spent currently.

3.2 Project Goals and Targets

When undertaking this project, the time researching literature associated with this topic has shown there are a number of things that this project must broadly achieve in order for it to be successful. Without a number of the following being met, the project will not be effective for the purpose it has been created for. Delivering the majority of these goals however, will ensure a project which displays ingenuity and has real use and value to the target audience. The goals and targets of the project are set out below:

- Create a new piece of software which successfully integrates Natural Deduction proof system and gamification concepts.
- Give users an engaging experience where they benefit educationally.
- Support users and give them help and advice when stuck in the game.
- Make the software accessible so little prior knowledge of Natural Deduction is known.
- Use proper logical notation so users can effectively transfer knowledge learnt in the game to the academic environment.
- Deliver a usable software project on or before 29th April 2016.
- Use Kinaesthetic Learning techniques to benefit users who learn in this style.

- Take current attributes in educational games to make sure my project encapsulates qualities of what makes games effective learning tools.
- Make users feel more confident and capable with the Natural Deduction proof system as a result of playing the game.

3.3 Software Requirements

Effective requirements set before implementation of software leads to a focused design process and a structured framework. There are functional requirements which is what the software should do Wiegiers & Beatty (2013) and non-functional requirements which describes how the system is expected to work. Chung & do Prado Leite (2009) Within the functional requirements there are Gaming Elements. These are specific functional requirements that will make my software play like a game, rather than just a drag and drop tool.

3.3.1 Functional Requirements

3.3.1.1 Provide an intuitive drag and drop interface

Making it clear what to do and how to do it. Making sure that the user knows how they can drag and drop items. An algorithm will have to be implemented where users can drag and drop onto certain elements of a proof. This could be that a user can drop onto a certain area of a proof, or could drop onto particular elements of a proof. This will need to be investigated in the design of the program.

3.3.1.2 Making all elements of the game drag and drop

This would mean no part of the game would need to be controlled by the keyboard. Everything being able to be dragged and dropped makes it nicer to use.

3.3.1.3 Have proofs rendered nicely so they fit on the screen

Proof trees can get very big so this would be a way of nicely scaling proofs on screen so a user can easily tell what the proof is. This will have a function that will render the proof tree.

3.3.1.4 Allowing the user to experiment with proofs and possibly create their own levels

Creating a flexible system that can be expanded upon is important for future development of the system.

3.3.1.5 An algorithm or a system that can successfully recognise whether a solution to a proof is correct

In an ideal world an algorithm would be automatically be able to determine whether a solution to a problem is a correct solution. If this is not feasible during the time period then a hard-coded system should implemented.

Gaming Elements**3.3.1.6 Provide hints and tips for the user**

Hints and tips enhance a user experience as if they get stuck, then hints and tips will prevent them from giving up. At least 75% of the people that used the hints should find them effective and that they made a difference.

3.3.1.7 Have levels that will increase in difficulty and guide the user through the learning process

A gradual build up in difficulty can guide a user through the game and keep them entertained and challenged. The process of a user being able to progress through the levels is an important gaming element that this project should implement.

3.3.1.8 Create a reward, scoring or recognition system for users to tell how they are doing

This could be the form of a score counter, letting the user know how many levels the user has got correct, or could be as simple as just letting the user know whether that individual level is correct or not.

3.3.1.9 It should be more fun than learning Natural Deduction logic in a traditional way

The key factor for this game to be successful is that users can learn the required material in a more entertaining and engaging way than they would

learn it in an educational environment. Expectations are that 75% of users should find this game fun to play. It is also expected that 75% of users should prefer learning through this game compared to learning in a traditional way like pen and paper.

3.3.2 Non Functional Requirements

3.3.2.1 Testing must be undertaken to make sure the software is working as desired.

- Continuous testing must be done by the programmer to make sure that the code is continuing to react as expected and there are no obvious bugs present.
- User testing must be completed before the end of the project to get feedback on improvements to the system.
- Some unit testing should be completed to check that the code is solid and nothing unusual should take place when in use.

3.3.2.2 The system must react consistently with 100% reliability.

- Every time elements are dragged and dropped, the system must react the same so everything happens as expected.
- Algorithms in place are reliable and executed as expected every time.

3.3.2.3 The system must be robust and secure.

- The system must be able to cope and should not be able to be changed permanently by anyone.

3.3.2.4 The code should be written in a way which is easy to maintain.

- The code should be written in small functions which can easily be read by other developers.
- The code should be sufficiently commented so that other developers can easily understand what is happening in the algorithms.
- All function and variable names should be sensibly named so that other developers can easily grasp the gist of the code.

3.3.2.5 The program must be written in a way in which it could be scaled up in the future.

- The code must be written in such a way that it could be expanded on in future. This can be achieved with a clear layout of the code, where the maintainability requirement is also achieved with similar approach to how the code is written.
- The software will be open source to allow for the continual development of the platform.

3.3.2.6 The user should feel like they have a better understanding of Natural Deduction on completion of the game.

- The goal should be to have at least 75% of people who do not know about Natural Deduction prior to the game to have a better understanding.
- Another goal that at least 80% of users feel more comfortable with Natural Deduction proof systems following finishing the game.

3.4 Hardware Requirements

All software projects require hardware for them to run on. It is important to outline what the capabilities of the software should be so it can run on the correct hardware.

3.4.0.1 All code should run by the client.

- Using a language which uses client side increases the speed of load times, reduces the reliance of a server.
- By deciding to run the code client side, it is important that the code is not too computationally demanding so that it runs on a variety of architectures.

3.4.0.2 The software should be able to run on a variety of browsers.

- The language chosen must be picked so that it can be easily integrated with web development.

- It must be able to work on a variety of web browsers so that the software is available to use by a wide range of users across different platforms.
- The code must work on different architectures and different operating systems.

3.4.0.3 The software should run comfortably on a PC or Laptop and be played with a mouse.

- This software project is designed for use on a PC or Laptop. It shouldn't necessarily be made to work on mobiles or tablets with a touchscreen due to time restrictions.

3.5 Programming Language

When considering the structure of this project, thought needs to put in about the format of the game, how the game would be accessed and played. These factors directly influence what language should be used to write the software, as different programming languages have different strengths. The Software and Hardware requirements also needed to be given thought to assess which language would be most suitable.

Initially Java was considered for the development of the project. Java is an object orientated language which would be an important feature in this project for creating proof trees. Java is the language where I have had most experience and so therefore seemed a good choice. A major disadvantage of Java was that it does not lend itself nicely for in-browser development which therefore could lead to problems when it was time to test the software and distribute the game to users. Using Java means the a web application would have to be created on a dedicated server which runs the program. Vogel (2009) This is something that this project was aiming to avoid.

Considering accessibility requirements and software requirements, a browser based game was decided on early in the process. This led me to a choice where the project could either be created in a language that is native to being online, or create it in a different language and then allow a user to download and install the game. The application would be quite lightweight and as such there is no need for external databases or dealing with large amounts of data. It was therefore decided that a game that could be played in the browser would be the easiest way to move forward.

Javascript and PHP are both widely used for website development. Javascript and PHP both have their advantages and disadvantages and personal pref-

erence was the main reason Javascript was picked for this project. PHP also makes it easy for quick web development so would have been another sensible choice. PHP however is used more for server side programming, something that was felt that this project did not require. Nixon (2012) To keep this project as simple as possible for both the programmer and the user, server side programming was avoided. There is nothing in this project that would actively need the processing power of an external server, and everything that is created in this project should be able to run on the host's computer, as it is not very demanding.

Due to the mathematical nature of the project, the logical programming language Prolog was also considered. Whilst this would not naturally lend itself to web development, it would be a strong contender for the best language to code the mathematical logic of the project in. Research into proof checking using prolog exists meaning it is a suitable language for proof verification. TSUKADA (2001) This would be the logic of the game could be verified by Prolog for a back end and another language would need to display this on the front end. The main reason that Prolog wasn't chosen was that it would add significant complication to the project. Most of the logic concepts to be used in this project are fundamentals and Javascript could be used to verify these proofs.

I settled on a HTML 5 canvas for the playing area and Javascript for the majority of the logic. Javascript seemed a sensible option because of the way it runs. Javascript is an interpreted language which is suitable for in-browser execution, unlike Java.

It is fast and has small overheads because it is a client-side programming language. It doesn't need to contact a server so there is little latency, meaning that any code can be executed straight away. Another reason that Javascript was chosen was because of its ease to learn and simplicity. There is lots of documentation and tutorials available online and this makes Javascript accessible to new web developers. Crockford (2008)

The fact that Javascript can be easily integrated with HTML was another big reason that it was picked for the project. A HTML 5 canvas was the easiest way to get a drag and drop interface working and made it easy to show a user which area is clickable. The logic is implemented by Javascript with the HTML canvas being the area that the user sees on the front end. Using these two languages together makes it easy to publish a web page, with no servers running continuously storing data. All that is needed is a website host.

Javascript is generally accepted as an object orientated programming language, where objects can contain other objects. Crockford (2001) This is important with the data structure of the project, where a tree structure

needs to be implemented to correctly draw natural deduction proofs. By having a structure where objects can be placed within objects, this tree structure can be achieved.

3.6 Project Limitations

Completing the core requirements for the project are an absolute minimum its success. There will be room for further improvement that will be outside the scope of this project.

Due to the limitations of web based development, storing information to file would need to be done on the server side. As this project is client side, no information will be able to be stored. This project does not plan to use a database to store information so users will not be able save levels for future use. Levels could still be created by users and then the data structures could be emailed so they could then be incorporated into the project. An idea could be for a user to press a button to submit an proof suggestion.

Another limitation of the software is that there will only be a finite set of things that the project can do. Whilst the basic elements of conjunction, implication and negation should be implemented, it may be more challenging to implement other operators that work in a different way. By getting at least the basics of Natural Deduction in there and working correctly, it will still be a functioning and valuable learning resource for an introduction to Natural Deduction. Designing the code with scalability in mind will make sure that more complicated operators can be added in later versions of the software.

3.7 Changes to Existing Systems

Of the number of other systems that are currently in existence, it is important to recognise what are their strengths and what should be improved upon. Pandora is "a tool for supporting the learning of first order natural deduction". Broda et al. (2007) It does a number of things successfully that this project would like to take on board. It has a series of exercises which users can work through which is what this project is aiming to create in the form of levels in a game. Pandora successfully teaches the user by increasing the difficulty of levels and giving hints and pointers as the user works their way through the game. Hints and tips is an important feature in the learning process so implementing this correctly and taking inspiration from Pandora will be useful in sculpting this project.

The main thing that will be done differently in this project will be the style of the proofs and how they are laid out. In Pandora, proofs are displayed using the Fitch notation. This is a series of lines where each one displays the next line of the proof. Whilst this notation is perfectly valid, in British Universities Natural Deduction and other logic systems are traditionally taught using the tree structures where all premises are above the line and the conclusion is below the line. Conclusions can also be premises for other logical statements and this is how the tree like structure is built.

Panda is an alternative current Natural Deduction tool that currently implements a tree structure. This is what this project does successfully well. Gasquet et al. (2011) This software displays the proofs graphically in a way I would like to replicate. Panda has a number of buttons for introducing a variety of different rules. This is also a nice way to do things and the user can easily understand what is happening.

One thing that this project will aim to do that Panda does not do is make the complete interface drag and drop. Using Panda, there is still a fair bit of typing that needs to be done, which could be perceived as a bit clunky as the user needs to switch between the keyboard and mouse to undertake the tasks. If this software is created so it is all click and drag, it not only makes the process of playing smoother, it could be easily created for other devices like touchscreens in future development iterations.

Both of these tools do an effective job of using Natural Deduction in an unconventional way by creating software that people can learn from. What this project should build on is implementing more features to make a user feel like they are playing a game. If some of the key gaming attributes could be introduced to software which can teach you natural deduction logic, then users should be more engaged and will not even realise that they are learning when they are playing.

Chapter 4

Design

4.1 Data Design

Outlined here are the different data structures that will be used to create the project and to make sure that all the data structures that are required are accounted for and that everything that the project will need to work successfully is outlined prior to implementation.

The user interface will consist of a number of drag and drop elements, each one can be individually selected by the user and moved. Every time a new element that can be dragged and dropped is created, a new object should be created that will have a number of variables that are individually associated with that object.

The x and y position of the object on the screen will need to be stored and these values will be updated every time that the object is moved. The x and y coordinates stored will correspond to the centre of the object and this will be the point that the object will be moved from.

A piece of data that needs to be included in every object made associated with the x and y coordinates is the width and height of the box. The width and height of the box will vary automatically based on how much of the proof needs to be drawn on it. The width and height of the box will determine the external structure where the border is drawn, with the idea that all of the proof should always fit within the structure.

The structure of the object should be stored. This will be how the proof is interpreted and the drawing function will recognise the proof structure and use an algorithm to draw the proof correctly. The structure of this will be an array of four elements. Each element will be one of three types, a string, a number, or another array. If text for a proof needs to be shown then that

particular element will be a string. A user can drag on another part of the proof where a dot is present. In the proof structure, this is displayed as a number. The drawing algorithm will recognise the number and render it as a dot which can be dragged on to. Thirdly, the array element could be another array. This would represent further proof structure that needs to be drawn. The recursive process would then start again where this nested array would also have four elements.

The colour of the border will need to be stored. This will indicate whether this object is hovering over another element. If the border colour is changed when the user lets go of the mouse, then an action will happen. A different border colour could be used for different actions.

There will be dots in the proof where users can drag statements onto. Each dot will need to be identified by a unique number. These dot IDs will need to be stored in an array unique to the object. This will be because when a proof structure is deleted from the canvas, all the associated dots need to also be removed from the canvas so that users cannot drag onto a point on the canvas that no longer exists.

The object elements which will be represented as drag and droppable squares on the canvas. These elements will all need to be stored in an array so they can be iterated through and all be drawn on the canvas. The draw function will continuously update what is drawn on the screen. By storing all the elements in an array, any new elements can just be appended onto the end of the array, because in Javascript there does not need to be a fixed sized array. When elements are deleted, they can be removed from this array and then they will not be rendered on the canvas.

Every dot will also need to be its own object. Each time a proof is created, the correct amount of dot objects will also need to be created. This is because dots need to be dragged on by other elements so the individual position of every dot is needed so the software knows whether the user is hovering over the dot. This means that the x and y coordinates need to be stored as variables of the object.

Each dot object will have the ID number that that is represented in the structure of the proof and will act as the link between where the dot is rendered on the screen using the proof structure and the dot object which will hold all of the positional information.

The colour of the dot will also need to be stored. This will be the same colour as the colour of the object border that is being moved, so if the dot and object being moved are on top of each other then the colour of both will change. This will allow the user to clearly tell which dot the proof will go onto if the mouse button is released. Black will signify no relationship

with that dot. When the dot goes purple, there is a relationship between the moving object and the dot.

A final field will be needed which is a variable. This will be for if the dot gets replaced by a statement or a proof structure. Then whatever the structure is will be put into this variable. Next time the dot is due to be drawn, the algorithm will be able to recognise the presence of data in this variable and it will replace the dot with the structure. The normal drawing procedure will then continue and correctly draw the correct shaped proof.

Like the draggable objects, the dots will also need to be stored in an array. Whenever new proof structures are created, the correct of amount of dots will also need to be created and these can be added to the dots array. Dots can then be iterated through to find out if any positions of objects coincide with the position of the dot.

For each level of the game, data needs to be stored so when the user believes they have completed a level, they can check against the stored answer to verify. Stored answers to levels should be in the same format as the proof structure that will be created in the objects. When the proposed solution needs to be compared to the model solution, the two solutions can just be compared to see if they are the same. Solutions for each levels can either be stored as separate variables or as elements of an array. A unification algorithm should be implemented so that all valid solutions for each level are accepted.

4.2 Data Flow Diagram

Figure 4.1 displays how data is stored and used throughout the dragging and dropping process. It includes the actions taken by the user and the processes that happen after the user has executed an action. Boxes in blue indicate user actions, brown circles indicate processes happening and white rectangles indicate data that is stored. Chen et al. (2009)

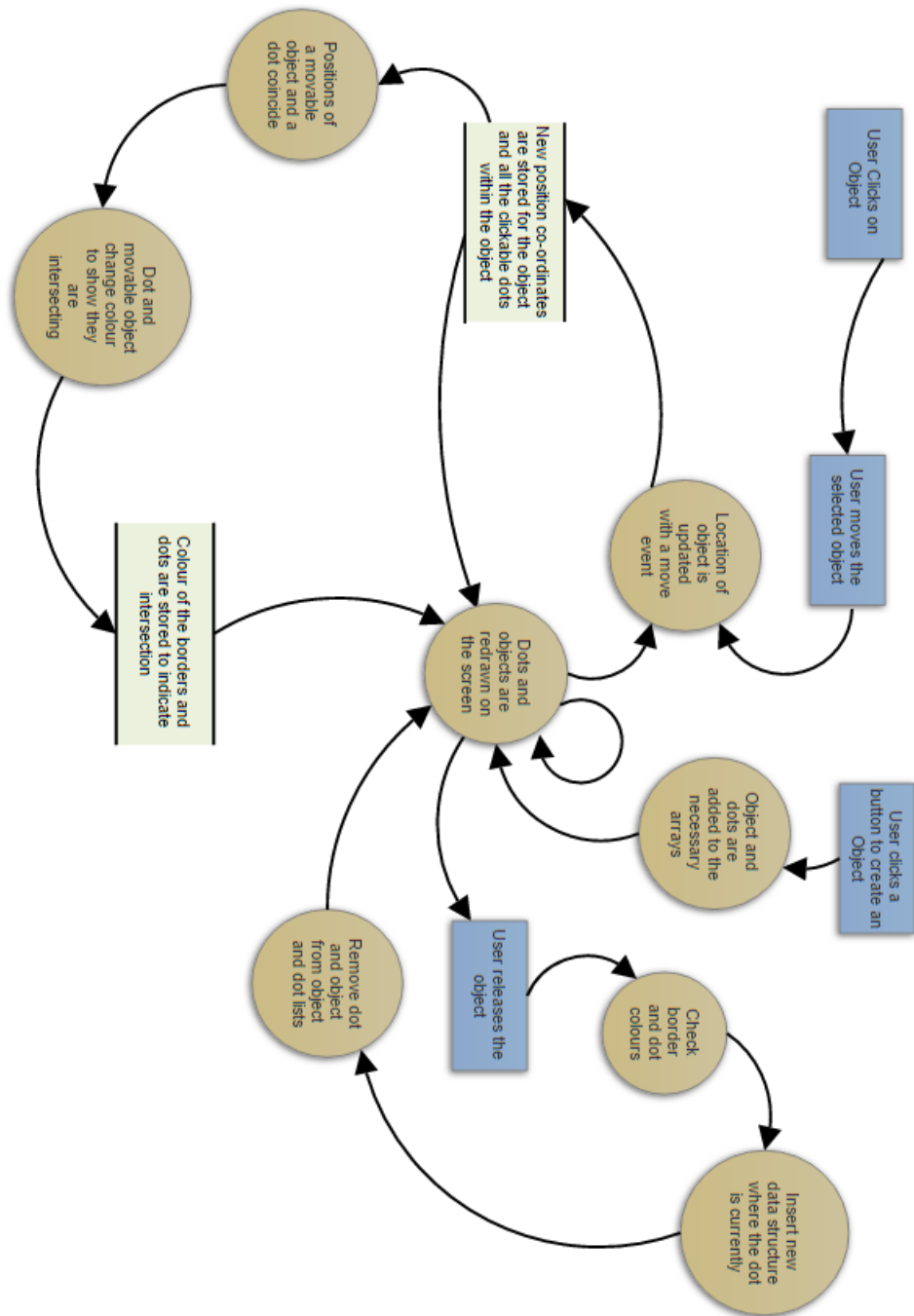


Figure 4.1: Data Flow Diagram on how Drag and Drop works

4.3 Architecture Design

This section describes the structure and layout of the program. It describes in detail how things will be done in the software and how all of the requirements will be met. It will outline how different sections of the program will interact and work together to create successful software without going into all the details of individual algorithms.

4.3.1 File Structure

There will be a very simple file structure to the program. The site will be run from a HTML file. Embedded in this will be 2 Javascript files, one which will be a linear tutorial for users to go through and a second file that will allow the user play the game. A Cascading Style Sheet (CSS) file will be in charge of the design of the program which will make it look visually appealing and align elements in the correct places. CSS files describes the presentation of a HTML file. These four files will contain everything that is needed to run the project. The two Javascript files will contain all of the logic of the program and have all the algorithms needed to draw the objects and to work out all of the dragging and dropping.

The tutorial file will provide the interface where the user presses a button to reveal more of the tutorial. This will explain the basic techniques to the user so they can gain skills and knowledge about the game. This will have the same layout and style as the main game which the user can interact with, so that the user will be familiar with what they must do when they have to make their own solutions. Each part will be hidden or visible as the tutorial progresses. As the user finishes this tutorial, they will automatically be transferred to the page where the user can play and interact with the game and fully play it.

The main Javascript file will contain many functions which all have specific tasks. All of these functions will link together to make the key logic structure of the software. Keeping the functions small and specific will make the code more maintainable and easier to read. It will also help when scaling the software and increasing its capabilities.

4.3.2 Key Functions

The three basic functions needed for dragging and dropping will be for when the mouse is clicked down, when the mouse is moved and when then mouse is released. Whilst these are the main functions, other tasks will be in separate functions and called from these functions.

When the left mouse button is pressed down, the function needs to determine whether the location of the mouse is over any of the drag and droppable objects on the screen. If it is over an object then this should be the current object selected.

When an object is selected then the function for moving the mouse will be active. This function will need to continuously update by using a timer. This will make sure the location of the dragged object is correctly represented when the mouse button is clicked down. This function must also recognise if the active object that is being moved has the same position as any of the other objects on the screen. If it does, then this function must react accordingly.

Finally, when the mouse button is released any appropriate actions needed will actually be confirmed. This could be the joining together of two objects, the deletion of objects or possibly other interactions. It will also release the current object so the software recognises that no object is now selected to be moved.

Creating these functions with the requirements in mind mean that all objects on the screen will be able to be moved using drag and drop. This would fulfil Requirement 3.3.1.2 and every action would be able to be performed with a mouse. This will make it a more usable interface for the user because they will not have to switch between a keyboard and a mouse.

Some of the main functions that will need to be called from these drag and drop functions are as follows:

- A deletion function which will remove both the object that can be dragged and the dots associated with it
 - By having a deletion function, the user will be able to make proofs and get rid of them as they choose. This allows the user freedom and the ability to experiment with proofs and delete them at will. Creating a deletion fixture will help the accomplishment of Requirement 3.3.1.4.
- A function that will check whether an object is being dragged onto another one
- Functions that will concatenate formulas and proofs based on what area of the proof was dragged on to

While these functions will be called by an event, (a mouse click or movement) there also needs to be functions that will render the proof continuously so that the user gets a good experience from dragging and dropping items and it is without lag.

A drawing function must be called periodically. This can be done that from an initial function that will set up the environment, create the canvas and set the correct level. The initial function will be called on startup so the canvas is always present when the application is first loaded.

The re-drawing function will have to cycle through all the objects and draw them individually. If the proof is a statement which will be represented as a string, it will need to be rendered differently to if it is a proof structure, where multiple layers will need to be drawn and the box size changing respectively. Requirement 3.3.1.3 states that proofs will be rendered nicely on the screen and this is the function that will achieve this requirement. Within the drawing function there will be a number of library functions used to draw text, align the text correctly and change the colours of text and borders. These library functions all work with the HTML5 canvas.

Users will need to create their own proofs from scratch and they will do this by having a number of buttons available to them which will then call a function in the Javascript to create the desired object. The buttons will be created in the HTML file and will call a function in the Javascript which is connected to that button. There will be Javascript functions for different proof shapes, variables and connectors such as implication and negation.

The last set of functions that will be important for the game to work successfully are functions that determine the level information and that will verify whether answers are correct. A function will hide and show certain information depending on the level that a user is on. Removing certain elements that is not needed for a level makes it easier for a user to know what to do. Also, certain hints and tips can be shown on various levels depending on what the user has learnt to do. This function will necessarily make sure the user is not overwhelmed with content and only the parts they need to use are included in each level.

A final function will be a verification check to see if what the user has created is a valid solution, and whether this solution is correct. This will be a comparison algorithm between an ideal, valid solution for a particular level and the user's attempt at the level. This algorithm should help the user and tell them when the solution is incorrect and prompt them to the part of the proof they should look at if they are wrong. The correct implementation of this algorithm will satisfy Requirement 3.3.1.5.

4.4 Interface Design

4.4.1 Design Principles

Designing the visual representation that the user will be presented with is crucial to the design process. Making something that is aesthetically pleasing to look at and also easy to navigate can be just as important to the user as the functionality of the software. Software which has the capabilities to do a lot and is computationally good but has a poor user interface can really be detrimental to user experience so thinking about how users interact with the software needs to be considered.

The target users will be teenagers and young adults in their early 20s. Being aware of the users who will play the game will affect how it is designed. This age group are technologically savvy and are used to frequently using touchscreens where swiping and dragging is commonplace. While this project will not be using touchscreens, it will be using PCs which this target audience are also familiar with. This audience does not need explaining how to use a computer. Creating an intuitive drag and drop with a good design as put forward in Requirement 3.3.1.1 will make it obvious what the user needs to do. By keeping a simple and straightforward design, the users can focus their time on learning how to use the Natural Deduction proof system and less time navigating menus. Users will achieve the levels quicker with a decluttered screen with only the necessary features displayed.

Consistency is also crucial when designing a user interface. Each level should have the same layout, the canvas in the same place, the buttons in the same place. By creating this consistency, the user will familiarise themselves with the structure and layout, and they will know what to do, for example, create a new variable object, or what is required when they need delete everything on the screen. It is especially important that the tutorial has the same format as the main game, because this will be the first time a user will see the game and the interface. This will be when most of the learning of the interface will take place.

The interface should be designed so if the user has not learnt the correct way to use the interface, they should be notified what they have done incorrectly. This is likely to be in the form of pop up boxes. This instant user feedback will not only help the user how to use the interface correctly, but also stop future similar mistakes. Users should not be in a position that they are puzzled in what is going on and therefore should be constantly told if an action they have performed is invalid. Designing the interface in this way will go some way to achieving Requirement 3.3.1.6.

Even if an action is valid, sometimes the user is mistaken in their action.

Users ideally would not want to start again if they make one mistake. Some way of rectifying a mistake will be beneficial for overall user experience. Creating a user interface which takes into account mistakes will hopefully be used much more that requires a user to go through a long process of creating their proof again. An undo button or a flexible system that lets a user replace parts of a proof will help improve usability for the user and therefore hopefully increase fun (Requirement 3.3.1.9)

Crucially, the user must feel they are in control of the system at all times. They will feel comfortable when what they expect to happen does happen and this is helped by the consistent clear design. If the user doesn't encounter any surprises and is not confused then the design is fit for purpose.

Spolsky (2001) Wood (1997)

4.4.2 User Interface Designs

Taking into account all of these attributes, it is important to envisage and plan what the ideal user interface design to look like before implementing it. Three designs for a potential user interface were suggested for different elements of the game. Explanations of the designs and analysis is outlined with the designs.

4.4.2.1 Basic Level Design

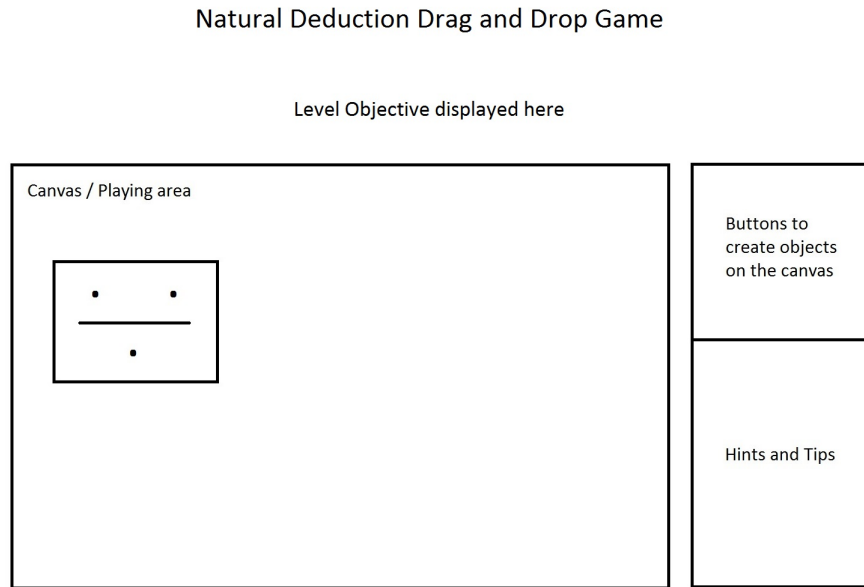


Figure 4.2: Design for User Interface

This first design is the basic layout of the game shown in Figure 4.2. It keeps a clean, simple design where every box has a very different purpose. The canvas is detached from the rest of the game to indicate the area you can drag and drop from. This will be outlined in the tutorial page. The box that is placed on the canvas is a proof structure which appears once a proof structure button is pressed in the top right box where the buttons are contained. Hints that the user can use to remind themselves what they have learnt are placed in the bottom right box and there may be an option to toggle hints on and off, so the user only gets the help if they need it.

Giving users convenient actions in the corners of the screen that will be used regularly will improve user experience. A waste bin and brackets will be placed in the corners of the canvas so users can drag there elements to either the bin or brackets, depending on what the user wants to do. These will be common actions that will probably need to be used on every level.

Usability is an important element of the game A duplicate button will also be placed in the top right corner and this will allow a user to copy the current statement into a brand new object. The aim here is to make it

quicker for a user to create a proof and reduce the repetitiveness of creating the proofs.

By keeping the core design to these simple elements, it will make the user feel in control because there are no complicated menus to navigate, making the user experience smooth.

4.4.2.2 Pop up boxes for level introduction

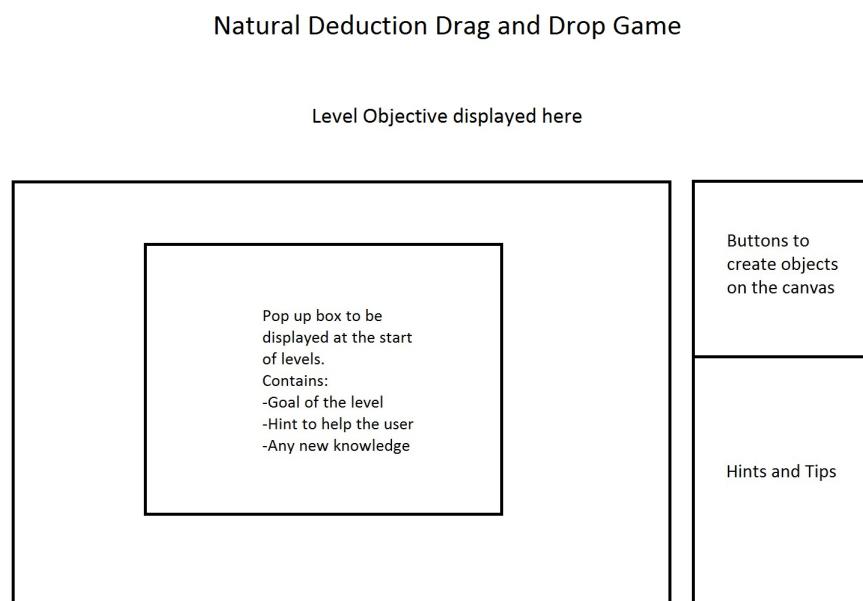


Figure 4.3: Design for User Interface with Pop up box visible

The design screen in Figure 4.3 takes a very similar format of the first screen. All of the boxes are drawn in the same place which keep continuity. This screen is to demonstrate what is presented to the user when they start a new level. All of the boxes including the canvas and the buttons will remain visible but not clickable whilst the pop up is active on the screen. The pop up will show the proof that the user needs to make. Pop ups will also be used to introduce new content to the user that they can then use in future levels. The user will have to click a button on the pop up to dismiss it and then they can play the level, like shown in the first design.

Users need to have instant feedback and one of the main ways they will receive it will be through pop up boxes. When a user believes they have

completed a level, they can verify it. A pop up box can be shown which will tell the user if they are correct or not. By providing feedback in this way, as soon as the user wants it, they will receive it. This takes on board this important gamification advantage and that instant feedback in games can be beneficial to the user by improving performance. Wu et al. (2012)

4.4.2.3 Multiple lines of proof and other objects on the canvas: Original Design

Natural Deduction Drag and Drop Game

Level Objective displayed here

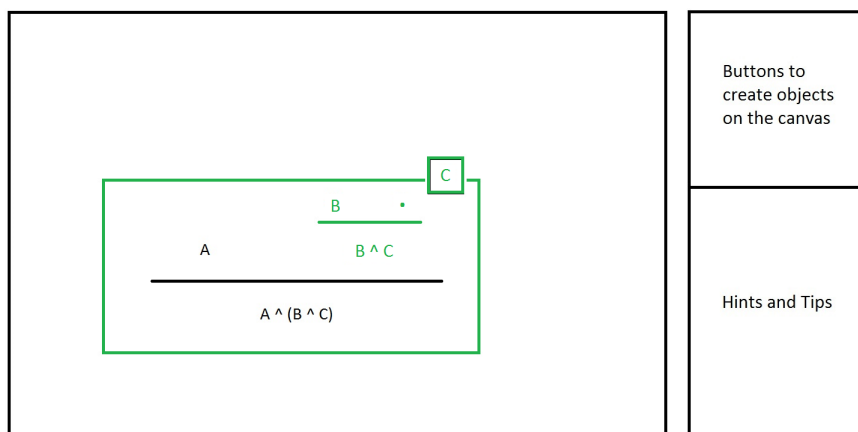


Figure 4.4: Design of how multiple lines of proof will be represented using the original Colour System

The original plan to drag and drop proofs was through a corner system. Dragging over certain corners will replace the proof over that area. In Figure 4.4 displayed above, the border of the objects and the part of the proof that will be replaced is highlighted. The 'C' will replace the proof structure in the top right corner that is highlighted green. Highlighting proofs makes it clear to the user about what part of the proof is replaced. The user will just have to drag over the correct corner of the proof within the square and doesn't need to actually drag over the text. In this design, for multiple levels of the proof proofs cannot be drawn from the root to the leaves because dragging over one corner of the proof will replace everything over that corner. This

may not be seen as intuitive as Natural Deduction proofs are commonly created from root to leaf. The next design solves this issue and goes further to fulfilling Requirement 3.3.1.1.

4.4.2.4 Multiple lines of proof and other objects on the canvas: New Design

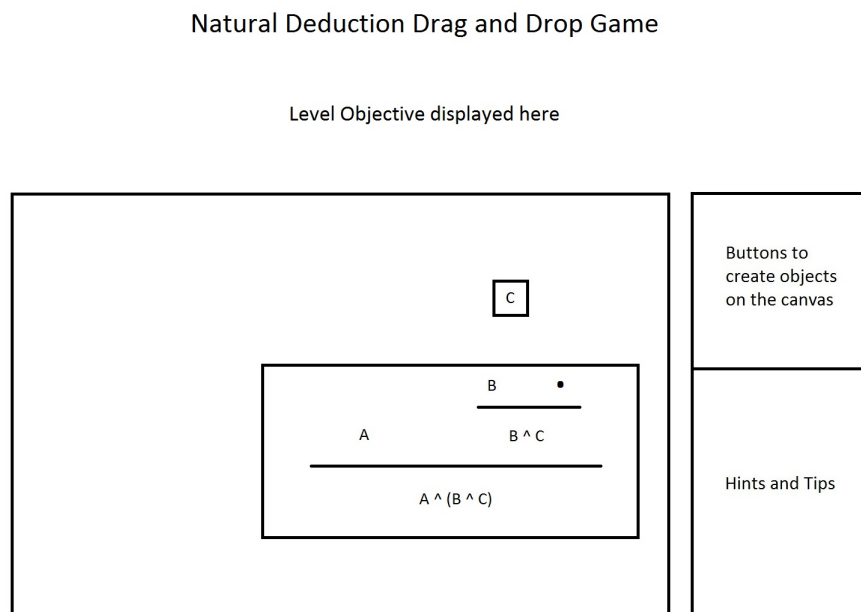


Figure 4.5: Design of how multiple lines of proof will be represented

Figure 4.5 displays how a user will build a proof of multiple layers. Once the user drags proofs together then the software will automatically detect the new structure the proof needs to be drawn in and will be rendered differently. The only positions that users will be able to drag onto the proof will be where the dots exist. In the design, the user would drag the 'C' element onto the remaining dot to complete the proof. Similar to the previous design, the border of the objects will highlight, so the users knows what objects will be connected. This solves the issue from Design 4.4 as proofs can no be built intuitively from root to leaf. A disadvantage of this system is that none of the proof can be replaced once one of the dots has been filled.

Chapter 5

Implementation

5.1 Beginning the project

As a new Javascript developer, the first few weeks of coding was spent learning the language and becoming familiar with how it all worked. This included knitting Javascript together with HTML and CSS. By using this combination of programming languages, Requirements 3.4.0.1 and 3.4.0.2 will be automatically met and this was the reason why these languages were chosen in Section 3.5.

For any software project, keeping a good record of progress is important. In this case, GitHub was chosen as the version control. GitHub was simple to set up and use. As well as it being used to maintain an archive of the project I also used to store notes as the project progressed. If a bug has been introduced in the version under development it is straightforward to revert to a previous committed version and carry on again from then. This project is stored in a public repository meaning that anyone can view the code if they want to. This makes the code open source and anyone can download and edit the code if they choose. This supports Requirement 3.3.2.5 and hopefully interested parties will contribute to the project and take it further.

Learning how to use the HTML 5 canvas as well as the methods associated with this that were available took some getting used to. I started off by creating a small game that was played on a HTML canvas. The user would use a button to create numbers. They could drag and drop these numbers on top of each other. If they dragged an item over the top half of another item then the two numbers would subtract from each other and if over the bottom, then the two numbers would add.

Addition and Subtraction Drag and Drop Game

Drag numbers to the bottom half to add them together (when it goes green). Drag a number of the top half of another one and it will subtract the one you are moving from the static one (when it goes red). Click create to begin. Numbers are created in order, starting with 1.

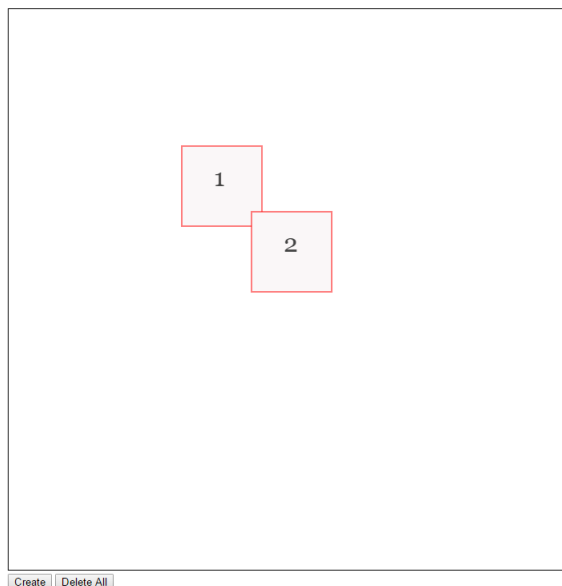


Figure 5.1: The first game I created with Drag and Drop

This game helped to understand how drag and drop worked, as well as understanding how items can react differently on a screen depending where the mouse is released. Most of this code is carried forward and used in the main Natural Deduction game.

5.2 Drag and Drop

With no prior experience of implementing a drag and drop system, an online tutorial was followed and ideas were expanded from these foundations. Litten (2010) This was the tutorial that was used for both the initial addition game and also for the main software project.

The three functions created in drag and drop are all event based. These events trigger when the left mouse button is clicked down, when the mouse is moved and when the mouse is released. All three functions have important roles and do different jobs that make the drag and drop successful. These functions were implemented according to the way it was described in the design.

```
1 function myDown(e){ //When the mouse is pressed down
2   move = 0;
3   for (j = 0; j < r.length; j++) { //Discovers which item is
4     //pressed down.
5     if (e.pageX < r[j].x + (r[j].w/2) + canvas.offsetLeft && e.
6       pageX > r[j].x - (r[j].w/2) +
7       canvas.offsetLeft && e.pageY < r[j].y + (r[j].h/2) +
8       canvas.offsetTop &&
9       e.pageY > r[j].y - (r[j].h/2) + canvas.offsetTop){
10      move = j;
11      r[j].x = e.pageX - canvas.offsetLeft;
12      r[j].y = e.pageY - canvas.offsetTop;
13      dragok = true;
14      canvas.onmousemove = myMove;
15      break;
16    }
17  }
18 }
```

Figure 5.2: The function executed when the left mouse button gets pressed down

When the left mouse button is pressed down, an event handler runs, which runs the function in Figure 5.9. The function iterates through all of the objects on screen. These objects are stored in the array 'r' and this function sees if the location of the mouse click is in the same position as any of the objects. If this is the case then the variable 'dragok' is set which indicates that an element needs to be moved.

The moving mouse function 'myMove' is constantly executed on mouse movement, but the majority of logic within it is only run when an object on the screen is selected to drag and drop because 'dragok' is set to true. The logic within the moving function went through different phases of development as the project progressed. Explained in detail below is both the legacy system and the new, improved way that it now works.

5.2.1 Original Colour System

This first drag and drop algorithm implemented was simpler and based on the design in Figure 4.4. This worked effectively and was based on the positioning of all of the objects on the screen. Thought had to be given to how the system would react when items were dragged on top of each other. The initial thinking was to base what happened to objects on the location of other objects.

The tree structure of natural deduction proofs means that the visual representation naturally splits into four different parts. Top left, bottom left, top right and bottom right are all possible locations where a part of a proof can be located. The original system worked out the location of all the objects on the screen. If another object was dragged over one of the objects, then depending which corner it was dragged over, the algorithm will set the borders of the two objects of a certain colour. Different colours would be set depending on what type of object is being highlighted.

Variables and conditionals can be dragged onto other objects of the same type. In Figure 5.3 below, dragging the conjunction onto the right hand side of the 'A' outlines both objects in red. Dragging the conjunction to the left hand side of the 'A' sets both of the borders green.



Figure 5.3: How to join statements together

The same logic applies to proof structures as shown in Figure 5.4 below, but all four of the corners could be dragged over to give different coloured borders. Releasing the mouse button when the borders are highlighted removes the object that is being dragged and places whatever is on the moving object to the static element in the correct corner.



Figure 5.4: How combining statements to proofs worked in the original version

The advantage of this method is that it makes it clear which part of the proof the user is dragging to. The use of colours makes it clearer to the user and it means once a user is familiar with it, they will learn what to do in future. This may be considered a disadvantage because it does not come across as intuitive because the colours are unknown to the user before they

start the game. There are also no obvious points on the proof where the user can drag to, and without any kind of prompt it can be difficult about the possible valid moves available.

Another issue occurs when multiple layers of proof are dragged together. This system of associating colours with the area user drags the objects to does not work intuitively when building proofs. Natural deduction proofs are often built from the bottom up. With this method, proofs cannot be built from bottom up because the dragging and dropping is all associated with the first object that is dragged onto. Any new proof dragged onto the top left or top right of the proof will remove the existing structure that is there. This leads to a peculiar way of forming proofs of multiple layers. Due to these reasons and reacting to feedback echoing this, the way users dragged onto objects was changed.

5.2.2 The Dots System

A redesign of the drag and drop algorithm made the software fulfil Requirement 3.3.1.1 more effectively, so it was a positive step, even if it was more technically challenging. The idea behind this was to indicate to the user what moves were valid. The goal of the new system was also to provide a more flexible way of creating proofs, where users could drag onto any part of proof that was missing. This was achieved by adding dots to the proof structure.

Dots are used to indicate to the user where they can drag onto on the proof. It makes it much clearer visually and it is more intuitive. Locations of each dot needed to be stored. This meant that every dot had to be a separate object with its own unique ID number and location information. These ID numbers would link to every proof object, so each proof object that can be dragged and dropped knows which dots are associated with it.

Also in the dot object is the colour of the border. If one of the objects is dragged over the location of the dot, then both the dot and the object being dragged over will turn purple. This is to let the user know which elements are being highlighted and what is going to happen if the user releases the mouse. This is a great improvement on the previous system and will reduce errors from the user because it is less likely that the user will make mistakes.

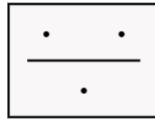


Figure 5.5: The new representation of the “2 Top/1 Bottom” proof structure

Another improvement from the first iteration of the code is that this system is more natural in the way proofs are created. Proof structures can be dragged onto the dots at the top of the current structure which leads to building a proof from bottom up. This is the conventional way to create Natural Deduction proofs in the tree structure and will offer a more realistic experience to the user.

In this way of dragging and dropping, the border of the object does not change colour when the user drags onto it. Only the dot of the static object changes colour. Dragging to any corner of the object no longer has any effect, it is all based on where the dots are located. Because of this, multiple layers of proof can be built up with lots of dots contained in them. The user can then individually drop variables or statements onto the dots. This was something that was not possible in the previous way of dragging and dropping.

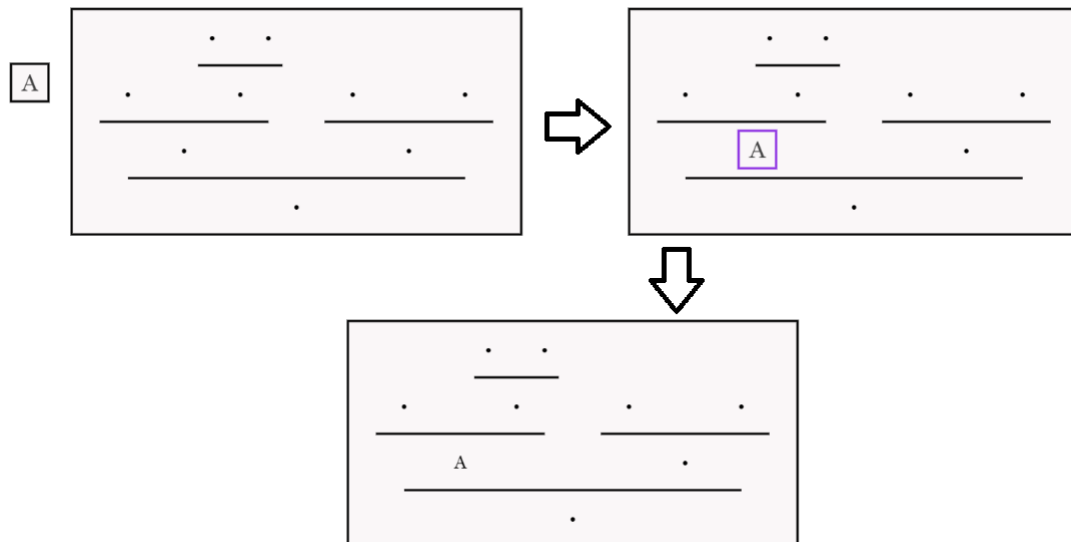


Figure 5.6: Example of how a user can drag onto a dot

When another structure is placed on top of a dot, the dot object is replaced with the structure that the moving object has, whether that is another proof structure or a statement. This method of dragging and dropping proofs solved both of the main issues encountered in the original colour system. This is based on continuous development and unit testing as well as constant dialogue with interested stakeholders such as the project supervisor Willem. Constant testing and improvement is a key part of any software project and will be covered in further detail later.

5.3 Data Structures

Due to the visual representation of the proofs as trees, storing the data in a similar way made sense as it made the design scalable. Basic objects are stored as a list. Each draggable object structure was the same, whether it was a proof structure or a variable. The core structure had four elements. The first element represented the top left corner, the second element represented the top right corner and the third and fourth elements represented the bottom left and right corners respectively.

If an object is just a statement, like a variable, connector or a mixture of these, the string representing the statement would be placed in the first element and the remaining three elements would contain “%”. A percentage string was one that let the program know that this element will never need to be used for this part of the proof. For the statement $A \wedge B$, the data structure would be represented as $["A \wedge B", "\%", "\%", "\%"]$.

These percentage strings were also used for determining the layout of a tree structure. Two elements on the top and one on the bottom is used as well as one element on the top and one on the bottom in this game and they are represented with the following data structures:

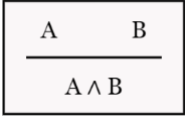
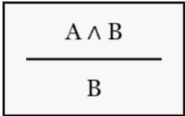
2 Top / 1 Bottom	$["A", "B", "A \wedge B", "\%"]$	
1 Top / 1 Bottom	$["A \wedge B", "\%", "B", "\%"]$	

Figure 5.7: Simple data structures

When dots are represented in the proof, the ID number of the dot number is the element in the proof structure. The drawing algorithm will recognise a number and render it in a dot. The ID numbers of the dot will link to a dot and the dot object with the position coordinates associated with it.

The way larger proof trees are stored is in the form of nested lists. If an element of the list is stored as another list, then a proof structure of the same format can be contained there. This system works recursively to create proofs in tree shapes, which can then be easily be drawn by the software.

Proofs with dots and multiple layered proofs are represented as follows:

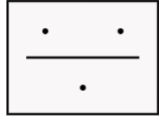
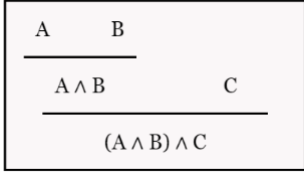
Dots in a proof	[1, 2, 3, "%"]	
Multi-line Proof	[[["A", "B", "A ∧ B", "%"], "C", "(A ∧ B) ∧ C"]]	

Figure 5.8: Additional data structures

A simple, unified proof system allows both ease when drawing and when assessing the proof to check whether it is valid. As the structure is consistent, comparison is possible.

5.4 Drawing and rendering algorithms

The drawing algorithm is another important part of the software to make sure that the proof is presented to the user in the way that they expect. Presenting information to the user correctly and reliably is a key feature that this project must achieve. The drawing algorithm uses *setInterval* which is a Javascript function that calls a function at certain intervals. The function called contains the drawing algorithm that redraws what is on the canvas, giving a user instant feedback to whatever moves they are performing.

The drawing algorithm has been written in a way that it interprets the data structure to correctly render the proof in a tree shape. The drawing algorithm takes into account whether each structure is “2 Top/1 Bottom” or

“1 Top/1 Bottom” and draws it appropriately. It also takes into account any dots in the system, which are represented as numbers in the proof system. The algorithm recognises these numbers and renders them as a dot to the user.

Figure 5.9 gives a flavour of the complexity of drawing the proof. This Figure shows whether a dot or text should be rendered in the top left of a proof structure. If neither of these things are here because the proof is “One up/ One down” or that there is a proof structure above this point, then it will go into different logic of the algorithm.

```

1  if(text[0] != "%" && text[1] != "%" && (typeof text[0] === '
    string' || typeof text[0] === 'number')) {
2      //'%' in text[1] would mean one statement at the top of the
    proof.
3      //A string means some statement can be written, a number
    represents a dot.
4      ctx.fillStyle = "black";
5      if(typeof text[0] === 'number'){ //Dot
6          if (dotsArray[text[0]].formula != ""){ //If a
            statement has been dragged on
7              temp = dotsArray[text[0]].number; //to an object,
            change the array to
8              text[0] = dotsArray[text[0]].formula; //represent that
            statement in the object
9              if(typeof text[0] === 'string'){
10                 ctx.fillText(text[0], x-(w/4), y-dist);
11             }else{ //Proof structure dragged on
12                 if (typeof text[1] === 'string' || typeof text[1] ===
                    'number'){
13                     r[i].proofHeight = r[i].proofHeight + r[move].
                        proofHeight - 1;
14                 }
15                 r[i].dots = (r[i].dots).concat(r[move].dots); //
                    Update the dots an object holds
16                 r.splice(move,1); //remove the old object from the
                    canvas
17             }
18             dotsArray[temp].number = "deleted"; //ignore the dots
                in the old object
19         }
20         else {
21             dotsIterate(text[0], x-(w/4), y-dist); //update
                position of dots
22             ctx.fillText(".", x-(w/4), y-dist);
23         }
24     }

```

Figure 5.9: A code snippet from the drawing algorithm

The main point to understand from this code is that dots are represented by objects and the position location of them needs to be updated. The visual representation of the dot object needs to be a dot string, even though it is represented as a number in the array structure. This code gives an example of the complicated logic going into drawing canvas items correctly.

Multi level proofs need to be drawn in a different location of the screen because they are on top of the first level of structure. Due to the potentially different proof widths that need to be drawn, the first 4 levels of proof are individually accounted for. If more than four levels of proof need to be drawn, a recursive function is used. Another box linking to the proof is drawn in the same format as the original proof. This is done for ease of readability for the user and with Requirement 3.3.1.3 in mind. The user will still be able to read all parts of the proof easily.

In retrospect, a better drawing function could have been implemented. One that offers the whole proof in one object, no matter how tall the proof. As long as it remains readable, it would be ideal. An algorithm that uses recursion more would reduce repetitive code. Tree like structures lend themselves well to recursion, so this could have been utilised more.

A feature that works well is the resizing of boxes. When the height of a proof is increased, the size of the box will automatically increase as well. As the height increases, the width will also increase proportionally as well. This is to make sure that items on the top row on the proof will still be rendered clearly and that the user has a good user experience.

5.5 The Game Play

Although making sure the mathematical logic was complete and works successfully, it is just as important to provide the user with an enjoyable game in accordance with Requirement 3.3.1.9. There are many tools already available that will aid a user in learning Natural Deduction, but as previously discussed, many of these do not engage the user and lack other key elements that make these tools into games. Part of the implementation of this project was putting in key gamification attributes that would make this tool a game.

One of the main features that was important to include was the concept of levels. (Requirement 3.3.1.7) These are set up with guidance to help the user through the game. The implementation of levels in this game prevents the user from using certain elements and this helps the user by guiding them in what buttons they can press. In Figure 5.10 below, which represents the buttons in Level 1 of the software, only the buttons needed to create $A \wedge$

B are available to press. Buttons unavailable are greyed out and when the user hovers the mouse over the button, a 'not allowed' symbol is displayed. These buttons use bootstrap and are toggled in the Javascript by setting a property whether they should be enabled or disabled.



Figure 5.10: How Buttons are represented in the game

Requirement 3.3.1.4 set out to let the use experiment with creating proofs. This is implemented in part by the implementation of a Freeplay button. Figure 5.10 displays the Freeplay button which the user can turn on and off at any time. Pressing this button makes all of the other buttons available to the user (they can be pressed) apart from the 'Check Proof' button. The level objective is also removed. A user can still drag and drop everything but the proof cannot be compared to anything. Unfortunately the user cannot save their proof so it is currently unavailable for a user to create their own levels. This would be something I would consider in the future work as it would let the software be more scalable.

An important requirement for this software project was to recognise whether a proof is valid and correct. In Natural Deduction logic you can't use distributive, associative or communicative properties. Laboreo (2005) Clicking on 'Check Proof' runs the checking algorithm. This will either complete the level and start the next one if the user is correct. If the user is wrong, it will let them try again. The pop up that appears when a user gets the proof incorrect displays the part of the proof where it is first incorrect. This helps the user and gives them a guide of which part of the proof they should look at. The implementation of this checking system gives the user hints, gives them instant feedback which are both requirements and fulfils Requirement 3.3.1.5 which explicitly states the need for a checking system.

A user gets a score of 10 for every level successfully completed with no mistakes. Every mistake they make (that is when they click 'Check Proof' and the proof is incorrect) is a deduction of two points. A user successfully completing a level correctly will always receive 1 point, no matter how many attempts are taken. Scores for all the levels are accumulated and a final score for a user is given. Having a score was one of the key gaming elements set out in Requirement 3.3.1.8 and this offers a competitive aspect of the game. The user can then play the game again and hopefully the user can keep trying until they get a perfect score of 10 for every level. Scoring is included to motivate the user to play again and hopefully subconsciously solidify the knowledge the user is learning about Intuitionistic Logic and the Natural

Deduction Proof System.

Additionally, further help is provided via a hints button that has been implemented. This is to aid the user in the learning process. Clicking the hints button reduces the score by 2 points, so it encourages the user not to use the hints, but they are available if needed. Having hints available to the user lets them continue with gaming process without getting frustrated. This will hopefully stop the user from giving up and they will persevere more by using the hints. (Requirement 3.3.1.6)

5.6 Design Implementation

The first design principle was to make the game look aesthetically pleasing to the user. To help with this, Bootstrap was used, which is a responsive front end framework. Bootstrap (2016) Using Bootstrap meant that whatever the size of the window, the buttons, title and the level objective were always visible to the user. Due to the drag and drop functionality of the canvas, this is a fixed size, so if it is not suitable then the user needs to resize their window or zoom in or out to successfully play the game.

All of the levels have a consistent layout, with the canvas in the same place. The buttons are always below the canvas and the level goal is always above the canvas. Having this familiarity means that the user will find the software easy to navigate. This consistent look to the software means the user will quickly pick up what they need do and then will be more comfortable as they progress through the levels.


Testing that has gone on throughout the development process means the user should feel in control of the system because there will be nothing unexpected. This resonates with the consistent design, so the user knows that what they expect to happen does happen. Figure 5.11 shows the user interface that has been created in accordance with the design. It shows the main gaming page with the hints visible.

Natural Deduction Drag and Drop Game

Level 1: The goal is to prove $A \wedge B$.
This uses Conjunction Introduction.

Score: 0

Premise



()

2 Top / 1 Bottom
1 Top / 1 Bottom
A
B
C
 \wedge
 \rightarrow
Delete All
Check Proof
Free Play
Hints

<p>How to Drag and Drop</p> <p>Move any proof structure onto dots that are at the top of a branch of the proof.</p> <p>A proof statement can be dragged onto any available dot.</p> <p>When the proof structure you are moving and a dot turns purple then these two elements will join when the mouse is released.</p>	<p>Conjunction Introduction</p>	
--	---------------------------------	--

Figure 5.11: The interface of the game

Chapter 6

System Testing

6.1 Developer Testing

Whilst the software was being written, testing needed to be carried out to make sure that the system responded in the desired way. With a relatively short turn around time for the system development, an approach of Continuous Delivery was taken. This approach puts the user first and make sure that the software project is always in a presentable state to the user and a demo can be given. By writing code in this way, it meant that if time didn't allow to complete certain features, then it was not critical. This is because the code is always in a working state where it can be delivered to a user. This is important for a software project that needs to be tested not only by developers and experts, but also by casual users from the target audience.

The Developer Testing consisted mainly of making a change to the system and directly testing that feature in the new version of the software. While testing in this way it is difficult to make sure that everything is working simultaneously. Responsibility lies with the developer to take the necessary procedures to make sure the software is working as required.

Constant sanity checks were made to ensure that the software maintained all the core functionality. This included making sure that the rendering is correct, making sure the data structures were being represented correctly and making sure there was no runtime errors in the Javascript.

Some of the requirements set out were to do with the code and the expansion of the project. Requirement 3.3.2.4 set out to make sure that the code was easily maintainable. This is obviously down to personal opinion but the effort has been made to fulfil this requirement. All of the function and variable names have been sensibly named, making the code easier to read. As

well as this, the important pieces of code have been sufficiently commented, again giving guidance in how the code runs. Every function in the code has specific tasks to perform, like drawing the proof on the screen, but this results in some functions get large. Even though this has happened, because of good commenting of the code, it shouldn't be too hard to follow.

Requirement 3.3.2.5 set out that the code should be scalable. As features have been added to the software, the code has got more and more complex, leading to little intricacies. Whilst it not that difficult to add new levels to the game, significant logical changes will be difficult because of the way the code is structured. One of the main areas for improvement here that would allow for a more natural and aesthetically pleasing way to display larger proofs. The current way is perfect up to a proof height of 4 but finding a way to display larger proofs nicely would take a large step towards the software being scalable as it would comfortably allow much larger and complicated proofs to be displayed on the screen.

6.2 Functional Testing

6.2.1 Unit Testing

Unit testing is difficult to achieve in Javascript because of the nature of intertwined HTML and Javascript. Even so there are some functions which are stand alone and can be run without interference from any other function. These functions that run stand-alone can be tested to ensure that their behaviour is correct. To do this I created a new Javascript file and created a number of unit tests. These included tests for when objects are created and when objects are deleted. It analysed if objects were being created correctly and did a number of tests on the data structures to verify everything was working correctly. Results of the tests were printed out to the console, so the developer could analyse the results. The result of every subtest was logged to the console and if every subtask passed then the test passed. At the end of the testing, a summary was given saying how many tasks have passed and how many have failed. This gives a clear indication on whether all of the tested functionality is working correctly. When changes were made to the system, this would be the first check to make sure that all the tests have passed before undertaking the developer testing. Unfortunately, most of the other functionality of the code is difficult to be tested and therefore needed to be handled by the developer testing.

6.2.2 Acceptance Testing

Acceptance Testing is where testing is undertaken to decide whether the functional requirements set have been met by the implementation of the software. Asking the end users if they have had a successful and rewarding experience playing will go some way to deciding whether the requirements have successfully been implemented. Most functional requirements are straightforward and do not require user opinion on whether a certain targets or features have been implemented. This will be obvious to the developer, so analysis can be done internally to decide which functional requirements have been met.

Functional requirements were set in Section 3.3.1 and these will be dissected, with the aim of explaining whether the requirement is met or not. The requirement 3.3.1.1 is set out to 'Provide an intuitive drag and drop interface'. This has been achieved. Speaking to interested parties led to a change in how the drag and drop system works, meaning it has been improved and become more obvious. As previously mentioned in Section 5.2.2, the drag and drop now works with a dots system. Being intuitive is obviously user dependent and in the event that users do not find the drag and drop intuitive, a tutorial has been created to explain to the users how everything works.

The HTML 5 creates a clear area where the user can drag and drop items. Users can drag all the elements that are created on the screen. The keyboard is not needed to be used for any part of the game which was prioritised in the implementation of the software. This successfully achieves requirement 3.3.1.2. By making everything drag and droppable, it lets the user do everything with the mouse, making it straightforward and simple.

Requirement 3.3.1.3 was to have 'proofs rendered nicely so that they fit on the screen'. This was important so that users could see all parts of their proofs and view them clearly. For the levels that the user have to complete, all of the levels are rendered nicely. It has not yet been implemented that users can create proofs of infinite size all within the same box. The way the algorithm was structured did not make it possible to draw an infinite size proof in the same square. For large proofs, connecting squares are drawn which connect the leaves of the Natural Deduction tree together.

A free play mode has been implemented so that users can experiment with proofs and create them themselves. It is not possible for a user to create their own levels and any new levels need to be implemented by a developer. Future work could be to allow users the options to send the developers the levels they have created. The data structure could be captured and these could then be used to create more levels. Allowing users to experiment with

proofs and letting users create their own levels was requirement 3.3.1.4 so this is partially fulfilled.

Requirement 3.3.1.5 states that there should be a system that can successfully recognise whether a solution to a proof is correct. This has been achieved and the system will let the user know what part of the proof they have got wrong if they are incorrect. The ideal solution needs to be specified by the user and then the algorithm will go through each section of the proof to make sure it is correct.

Whilst facts can be given about what the software is capable of, recognition of whether the users feels gamification requirements have been successfully met will be shown in the results section of the project.

The game has a hints box which successfully fulfils the requirement of 3.3.1.6. This hints box is available via a button at the bottom of the screen. Pressing this button loses the user 2 points from their score, but it is there to be helpful if possible. As well as this helpful hints box, learning prompts are displayed at the start of certain levels which teach the user how to accomplish certain tasks. A box containing the level goal clearly is located at the top of the screen. This tells the user the assumptions and the conclusions. This guides the user through levels as stated in requirement 3.3.1.7. Whilst all of these things aren't hints, they all give users clues about what they should be doing, contributing to the requirement of giving hints and tips to the user. There is also a tutorial at the beginning of the software which guides and helps the user through how to navigate the game.

There is points scoring system which gives the user points for each level depending on their achievements. This encourages the user to think for themselves about what they know before automatically going to the hints. This scoring system also informs the users in how well they are doing. This fulfils requirement 3.3.1.8 and adds a competitive nature to the game where users can compare scores with each other.

Requirement 3.3.1.9 is that the game should be fun. This needs to be decided by the user so this and other requirements will be set out in the results chapter where user feedback will be received.

6.3 Non Functional & Hardware Testing

6.3.1 Stress Testing

Making sure that software can handle a number of users simultaneously is important for this project, as it is feasible that multiple users can log on to the webpage and play the game at the same time. The game is hosted on

the University of Bath's servers, which process and hold a lot of data. Due to their industrial nature, I would assume that it would be able to deal with a lot of traffic and supporting users accessing the website at the same time wouldn't be an issue.

To test this theory, I used a load testing website that loaded 100 users onto the website and simulating them accessing the webpage simultaneously. Impact (2016) The results tested for load times and registered the number of successful virtual users on the site. Figure 6.1 shows that 100 users successfully accessed the site simultaneously and that no load time for the webpage was above 1.5 seconds. This is a good achievement as a user will not be waiting a long time for the page to load. This shows the advantage of having all of the code client side, because it does not need to contact a server to retrieve the information once it has loaded for the first time, meaning that that waiting time will be the only waiting time in the entire game. Server side code may have a similar delay throughout the program.

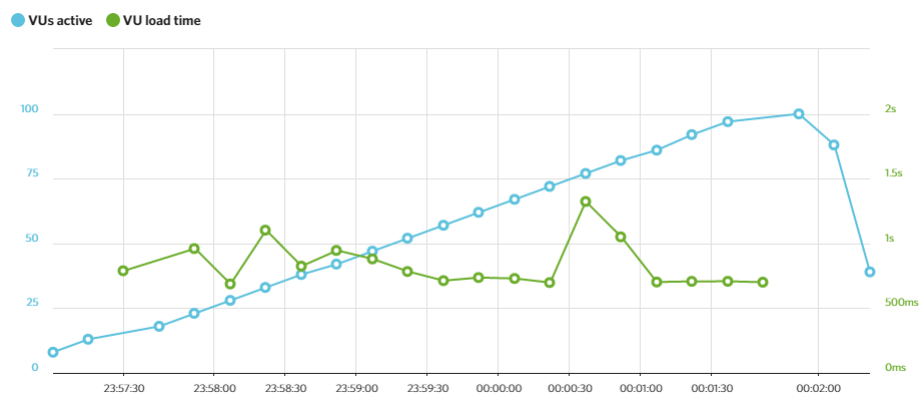


Figure 6.1: Load times and number of virtual users accessing the website over a period of 5 minutes

6.3.2 Security Testing

Security can be important to a lot of projects but in this project it is not too much of an issue. Sufficient security is in place that Javascript or HTML cannot be uploaded to the server or change permanently. This is another benefit to the whole of the code not needing to feed any information back to the server. If information was need to be uploaded to the server then there is some scope for inroads into the system.

Whilst it is not possible for the user to change the code in place that is stored on the server, because Javascript is being used it means that users can have access to the whole code base. This means that if the users really wanted

to then they could find the answers that are scored in the code to complete the levels. Whilst this against the spirit of the game, as it is meant to act as an educational tool, it is possible to circumnavigate the system in this way. This level of security that the code cannot be changed permanently fulfils requirement .

6.3.3 Compatibility Testing

As this is web software, requirements to make sure that it will work on a variety of different browsers, different operating systems and different architectures. This requirement was set out by 3.4.0.2. The code is not very demanding so it should be capable of working on even basic architectures. The only problem that may be encountered would be if the web browser doesn't support a HTML 5 canvas. Research was undertaken to make sure this requirement was fulfilled. The code was written on a Windows laptop running Windows 10 OS and an AMD processor. On this laptop, the website was loaded on Chrome, Firefox and Internet Explorer and it all worked fine. An online resource outlines which web browsers support a HTML 5 canvas stated that a HTML canvas is supported by all current versions of Microsoft Edge, Google Chrome and Mozilla Firefox. It is also supported by Internet Explorer 9 onwards, but the canvas not supported by IE8. The website goes on to state that the HTML5 canvas is accessible to approximately 94.38% of UK users, based on statistics about the browsers that are used in the UK. Deveria (2016)

6.3.4 Other Hardware Comments

By using a combination of HTML and Javascript for the software, it naturally achieves Requirement 3.4.0.1. This means all the code is loaded onto the client's web browser and as shown in reffig:stress, means that loading times are fairly small. These are the only loading times in the game because the server doesn't need to be contacted for more information. This leads to quick response times and smooth gameplay.

This software is naturally compatible with a PC or Laptop. Currently, it is not available to play on a touchscreen and this would be one of the main features that could be implemented in improvements to the software. Making the software native to a PC and Laptops realises requirement 3.4.0.3. Because of what was set out in the requirements, it was always planned to implement the software for mobiles at a later date.

Inflectra (2015)

Chapter 7

Results

The basis of this project is determine whether a game can improve a user's understanding of the Natural Deduction proof system and whether this game has the potential to be a beneficial way of teaching, whether stand alone or in conjunction with other methods of teaching. Testing needed to be undertaken by real users as this is what this software project was designed for. This is to see whether they like the interface, like the concept and whether they think the game is effective.

7.1 User Questionnaire & Feedback

A user questionnaire was decided to be the best and most effective way to get feedback on the game from users. Because the game is a website, an online survey seemed most appropriate. The user could play the game and fill the survey in straight away, both online. The survey was opened up mainly to my peers on my Computer Science degree. This reached my target audience and thought they would still be suitable candidates for testing because of their mathematical ability. This made me thought they would be capable of learning about Natural Deduction and Intuitionistic logic. Over a three day period I received 15 responses highlighting the positives of the software as well as improvements that can be made to the software. Whilst this is a small sample for making any research conclusions, it will help guide the future direction of the project and initial conclusions can be formed. All of the surveys were done anonymously.

7.1.1 Age and Mathematical ability

As mentioned in Section 3.1, the target audience would be mainly aged 16-21 and people with at least A Level Mathematical experience. These two groups shared the fact they are likely to have no or little prior knowledge of Natural Deduction style proof and Intuitionistic logic. This survey was answered by people with mainly aged 18-30. 14/15 people were aged 18-30 and the other participant was under 18. From the results, all of the participants in the 18-30 age group have at least A Level Mathematical backgrounds and 5 users have degree level mathematics. The under 18 user still had GCSE mathematics and therefore may have found the concept slightly harder. This age group with a largely strong level of mathematical ability shows that they are capable of learning mathematics and that they are all suitable targets to test the first version of this game on.

7.1.2 Prior Natural Deduction Knowledge

Users are expected to have little or very little of Natural Deduction prior to playing the game. It would therefore be the objective of this game to reinforce and teach the basics of Natural Deduction so even the most novice of users would be able to take part in the game and be able to solve some of the levels. If participants already knew something about Natural Deduction, it would be expected that they would get further in the game and would find it easier.

The results of the survey showed that two thirds of users didn't know anything about Natural Deduction prior to playing the game. The other third had a basic understanding of Natural Deduction. No one who completed the survey was confident with Natural Deduction. This meant that users learning, as well as playing, was an important factor to make sure that they came away from the game in a stronger position than when they went into it. The mixture of the Tutorial, Level Prompts and Hints were designed to do this and help the user learn whilst also assisting them through the game.

The question 'Do you think you have learnt more about Natural Deduction as a result of playing the game?' was a key one, due to the fact that this was one of the main project aims as set out in 3.2. By making participants feel like they have learnt something about Natural Deduction, the hope is that they will come out with a positive experience and that they will feel more comfortable if they have to come across Natural Deduction again. The survey revealed that 86.7% (13/15) thought that they have learnt more on the result of playing the game. Of the two that did not think that their knowledge of Natural Deduction had improved, one of them had a basic

knowledge of Natural Deduction whilst the other one did not know anything about Natural Deduction. It is a shame that one user stated that they did not learn anything about Natural Deduction and that they do not feel they benefited from the game whatsoever, but it is still encouraging that 8/9 people who knew nothing about the topic felt they learnt something and even 4 out of 5 people that had a basic understanding felt that they had a better knowledge of Natural Deduction following the game.

7.1.3 Tutorial

The tutorial and hints were two of the main sources of instant feedback to the user. By going through the Tutorial, the user should be able to navigate around the game and make their way through to a certain point. Requirement 3.3.1.6 made it explicit that the hints not only had to be present but they had to be effective with over a 75% success rate. As mentioned in 2.1.2.1, giving students instant feedback improves the learning achievements of students. Wu et al. (2012) It also mentioned that instant feedback can be more effective for a student Dempsey et al. (1993), so including hints and prompts was a key attribute to the success of the project.

The tutorial was designed to give users a first impression on how to use the system. This tutorial gives users an gentle introduction to the game and teaching them to using the drag and drop system. It lets the user complete all the actions that they are asked to do on screen and instant feedback is displayed on whether their action is correct or not. This was supposed to get them familiar with how to construct proofs and how to use the software. To start with, 100% of people that played the game decided to start with the tutorial. There is an option to skip the tutorial if you have played the game before but none of the users chose to do this. 100% also found the tutorial useful, which meant this was good technique to teach people how to do drag and drop in the game. Every user completed the tutorial so it was obviously explicit and clear enough. The hand holding nature of the tutorial which supported the user and gave them an instant reaction to every action they made was clearly appreciated and further supports the need for some instant feedback in the game.

One user commented that they liked the Tutorial most about the game, saying that the “Tutorial explained the concepts of the game well”. This ties in with the unanimous statistics that were received for the usefulness of the Tutorial.

7.1.4 Hints and Tips

After the tutorial, the main game begins. Here are where hints are available to the user via a button at the bottom of the screen. Hints, described in Section 2.1.2.4 are another crucial element to a successful game where the user can be prompted on how to make progress. Everyone that played the game used the hints. The response was mixed to whether the hints were useful or not. Two thirds (10/15) found the hints useful by using the hints button. This is a fairly low percentage and ideally wanted it to be higher. Requirement 3.3.1.7 set out a minimum benchmark of 75% of people finding the hints useful. Not hitting this figure shows that the hints were not helpful enough and didn't guide the user enough through the level.

Along with this result, there were some qualitative comments relating to the hints. One user commented in response to if they had problems playing the game "Couldn't answer it, still don't know how to proceed and the hints are just confusing". Another user gave a similar view "I don't think the hints were really "hints" they were more "these are the definitions for the techniques", I think they should be displayed on the screen the entire time and hints should tell you how to go about completing the level". In this alpha prototype version of the game released to the users, the distinction between the learning objectives and having worthwhile hints was merged.

<p>Conjunction Introduction The two statements on the top are assumed true. This means the statement on the bottom is true.</p> <div style="border: 1px solid black; padding: 5px; width: fit-content; margin: 10px auto;"> $\frac{B \quad C}{B \wedge C}$ </div>	<p>Conjunction Elimination The conjunction on the top is assumed true. This means A and C are both individually true and this is shown by eliminating the conjunction</p> <div style="border: 1px solid black; padding: 5px; width: fit-content; margin: 10px auto;"> $\frac{A \wedge C}{C}$ </div>	<p>Implication Introduction Implication Introduction means something assumed to be true, in this 'A' leads to further down the proof something else to be true, in this case 'B'. The following line can then be written as 'A \rightarrow B'</p> <div style="border: 1px solid black; padding: 5px; width: fit-content; margin: 10px auto;"> $\frac{\begin{array}{c} A \\ \vdots \\ B \end{array}}{A \rightarrow B}$ </div>	<p>Implication Elimination A implies B being true means that if A is also true then B is true by Implication Elimination.</p> <div style="border: 1px solid black; padding: 5px; width: fit-content; margin: 10px auto;"> $\frac{A \rightarrow B \quad A}{B}$ </div>
---	---	---	--

Figure 7.1: Format of hints given to the user on Level 6 when the questionnaire was completed

For a game, it is important to have sufficient and helpful hints for the user so that the user feels that they are really benefiting by looking at hints and that it gives them enough help when looking at them. In hindsight, the hints need to be genuinely different and helpful, rather than just the repetition of the rules that were already displayed at the start of the level. From an educational perspective, hints can be crucial in the learning process. Chen, Wei, Wu & Uden (2009) Good hints could make the difference between a user carrying on with playing or giving up, so prompting the user with useful hints that they can then make progress with, may help them get over

a barrier which otherwise they wouldn't be able to overcome and therefore give up and stop the game.

A response from a user gave a nice suggestion on how to improve the hints system "a better hint system that gives you a hint in to how the structure should be". The idea of displaying an empty proof structure with dots displayed is one that will be taken forward into further iterations of the code. As well as this, the algorithm that works out whether the proof entered by the user is correct could help to provide more support to let the user know which part of the proof is wrong. Helping users so they can overcome the problems they are facing during the game is one of the key qualities that a good game needs. All of this written feedback is helpful and a reworking of how the hints section works will be a suitable improvement to the system and help users overcome the difficulty they faced during the game.

Another suggestion was that the answer can be revealed if the user was stuck on a particular level. This would hopefully mean a user would be able to understand how the proof has been formed from seeing the answer and then the user can proceed onto the next level. By giving the user the answer, they would get no reward in the form of points for completing the level, but it will give them added knowledge of knowing how the level should be proved correctly.

7.1.5 Levels and Difficulty

Six levels were created for the users to work through on the game and it was interesting to see what stage people got to, how they thought that the difficulty increased as the levels progressed and if they didn't finish it, what the issue was. It was important that people could progress through the levels rather than giving up. If the user gave up, then they weren't learning so the game was not serving its purpose correctly. By getting users to 'do' things to learn, in this case formulating levels, kinaesthetic learning principles were being implemented, fulfilling a goal in Section 3.2. Each level started by introducing a concept to the user, which they would then have to create a similar example, using the rules that had been displayed and taught.

5 people out of the 15 only managed to get to Level 2 of the game. This figure is a third of the people who played the game and one of the reasons that people didn't finish the game reveal that they didn't feel that they knew enough about the topic of Natural Deduction. This is the responsibility of the game to help guide the user and through a mixture of lack of support to the user and difficult concepts, it made the game unachievable for some.

Responses for why users didn't complete the game included "Could not work

out how to complete the level correctly” and “Couldn’t answer Question 2, as I have no idea what proof it is wanting or how to prove it.” Both of these statements allude to the same thing that the Natural Deduction theory wasn’t explained clear enough and this greatly affected the progress of more novice users. This reaffirms the fact that a more comprehensive package of support is needed for users to make sure that everyone is capable of learning the material and successfully completing the levels.

More encouragingly 6 out of 15 users managed to get to Level 5 or further and 3 people completed the game. This shows that if users have the knowledge then it wasn’t the gameplay and usability that was stopping them from progressing. The main thing that was stopping users that struggled with the game was a lack of understanding.

Difficulty of levels was a mixed issue from the participants of the survey. When creating a game it is important to get the level of difficulty correct between levels Aponte et al. (2009). Some users commented with “Levels increased nicely” and “Good curve” whilst another user made the comment that “The difficulty increased quickly for a novice” and also that “Thought level 2 got too difficult too quickly”. If users have had previous experience with Natural Deduction then it will probably be the case that they didn’t think the difficulty curve was as difficult as to some beginners to Natural Deduction. This fact further highlights the lack of support for users that have not had experience of Natural Deduction logic and the need for more support to be given.

7.1.6 Fun

Gamification and Fun go hand in hand with the aim of the principle of Gamification for the game players to learn something through playing a game which as a consequence is more fun than learning in other environments. Requirement 3.3.1.9 highlighted the fact that this game should be more fun than learning in a traditional way.

If survey participants did not enjoy playing the game, or not realise it for what it is, which is to deliver education in a more exciting way, then the software has not delivered on key requirements. Enjoyment should be had, even though it is learning maths.

Gaming concepts such as a score system, multiple levels and interactive drag and drop have all been implemented and should all contribute to a fun experience for the user. The user was asked outright ‘How fun did you find the game?’. The mean score for the game was 2.79/5 with a median score of 3. This is unexpectedly low but comments associated with the enjoyment of the game explained key reasons why it didn’t fully entertain them.

One telling comment was that “There’s only so much fun you can find maths”. This reignites the consideration that the game should have been created without the proper Natural Deduction notation. During this project, the importance has been highlighted that the software should be a valuable educational tool. Without proper notation or structure, the difficulty would be transferring the skills learnt in the game to more academic environments such as a classroom. By having the correct notation in place, it allows users to easily take what they have learnt away from the game and use it in other Mathematical or Computer Science disciplines such as the previously mentioned Lambda Calculus.

Another observation from a user was that “It was quite good but it’s still doing proofs which are boring”. I think it’s important to realise that this is a game that is suppose to teach Natural Deduction so although this particular user may find this boring, they may still find it more engaging and entertaining than learning Natural Deduction out of a book. For the first version of the game that was released, fun was not the most important priority and this can be improved with fine tuning of the user interface.

7.1.7 Educational Value

One of the most important results of this project is whether the users thought they were learning and whether they thought there was value to playing the game and educating themselves in this way. Educational value of games was part of the reason this project was undertaken. If people thought that learning mathematical logic through gaming was valuable then this project will be worthwhile.

Of the 15 people that answered the survey, it was asked about how effective the game was as an educational tool. It was encouraging to discover that the mean average score was 3.93/5. Over 70% of participants ranked this a 4 or 5 for educational effectiveness which shows how people value this game as an educational resource. It shows that the majority of people thought the software was a positive educational experience. Along with the result that 86.7% of participants in the survey thought they learnt more about Natural Deduction than they did previously, the users seemed to value the educational learning benefits that the game offers. Users suggests by the results that this game has great educational potential.

Yet more evidence points to the users feeling that they benefited educationally. 11 out of 15 users responded believed that this game is a better way to learn than learning by traditional methods like pen and paper or learning in a classroom. Also seven out of nine participants answered that they had no prior experience of Natural Deduction and that this game was a better way

to learn that via traditional learning methods. This achieves Requirement 3.3.1.9 which was to achieve a minimum threshold of 75% of users preferring to use this game than learning manually. By having a high percentage of users that prefer to learn by games, it suggests that this game could be a feasible and more enjoyable way of learning.

7.1.8 Usability

Questions were asked to assess whether the user had a good experience and enjoyed the game they played. A user stopping playing because of usability issues is serious and harms the true reflection of the game's mathematical content. Some comments were received to that effect. This questionnaire is to pick out what does and doesn't work in the game and improve those parts.

All of the drag and drop worked correctly and no users commented on unreliable drag and drop which is an important usability feature that was worked on extensively to get correct. A smooth drag and drop interface was important for the usability of the game. Also, all the buttons were sensible colours, meaning that actions remained intuitive for the user, with red displaying the 'Delete All' button and green having the 'Check Proof' button.

When asking users whether they thought the game was fun, one user stated that "I found it really engaging because I wanted to keep going and reach more levels and complete the proofs and learn more rules, but the usability issues were really frustrating and it took me a long time to construct the proof for level 5 which I then did wrong because I misplaced one part of it. Starting again would have been frustrating so I stopped then". To have a user stop because of usability is just as bad as a user stopping because they are stuck. For whatever reason the user stops, this is the point that they are not learning, so the experience needs to be as pleasant as possible to make sure that the user doesn't give up.

A number of users complained about other usability issues. A popular topic was the lack of an undo button. "An 'undo' button to take back the last step" was a common criticism from users. This meant that if users made a mistake on the proof then instead of being to undo that move, they had to start again and create a new proof structure. Whilst an undo button may be hard to implement, other solutions may be explored to solve this issue.

The game being time consuming was a reason that users didn't complete the game. A slightly more usable and slick program may have improved perceptions of the game and users would have not given up as easily. One strong usability point is that the game only used a mouse. Removing the

keyboard made it easier for the user to navigate the game as they wouldn't get confused by a mixture of keyboard and mouse input.

7.2 Improvements to the game

Following feedback received from the game, there were a number of small bugs which were fixed. This included temperamental display of the hints and the 'Not Correct' boxes displaying unhelpful tips on how the level can be solved. Both these bugs have been fixed and will improve the support that is given to the user.

Other minor bugs that were fixed had usability in mind. A slight movement when the users picked up an object where 2 or more objects were on the same point has been fixed. The software needed to pinpoint which object was being picked up to stop this issue. Another small feature that has been improved which will enhance user experience is that the outline of the proof structure or statement will go turn red when they drag the shape to the bin and it is hovering over it.

Since the conclusion of the survey, a number of issues have more substantially changed to improve user experience. Firstly the hints have become more useful. Now one of the boxes displaying hints contains an empty proof structure, giving the user a structure which they can recreate on screen and then input the correct statements. This was important to add to the game because one of the key sticking points was that the user did not know what to do. By giving the user more prompts, it makes it more accessible and helps the user complete the game. Another one of the hints boxes gives the user more information about the level, helping them further.

The rules are now permanently displayed. This was done with education in mind. One of the strengths of the game needs to be that they are learning all the time. By presenting the rules to the user that are available at all times, the rules are always in front of them. This should reinforce the knowledge that the user is learning because of the ever presence of the rules. This are the educational parts that the user has received at the start of each level, so it is not new material for the user. Displaying this at all times will hopefully improve knowledge that is retained because of the repetition of the user seeing the rule.

7.3 Further Work

Taking into account the users' comments as well as my own thoughts on the project, here are some features that I'd like to implement in the future.

Users expressed an interest in an undo button so that they could alter proofs once they have connected them. Users commented that when making mistakes, the process was tedious to create a whole new proof structure when an undo button could be implemented. Implementing an undo button would be hard as it would take a lot of space to store all of the individual moves the user made. It shouldn't be too difficult to implement an undo button for one move though, where a snapshot of the previous move is stored which the user can revert to. It should also be made possible that users can drag statements over existing statements to replace them. This will reduce the need for an undo button in a number of cases and give the user an alternative option. There would still be the need for an undo button when two proof structures are dragged together as this cannot be undone. An undo button would be suitable for this purpose and should be implemented for this reason.

An additional improvement that could be made to the usability of the game is when users have acquired a particular bit of knowledge and are practised at it, parts of proofs are automatically entered for the user. Once a user has mastered a skill, they shouldn't have to go through the process of creating this every time and therefore variables should filter through the structure of the proof. This would reduce the time it takes for the user to complete levels. This would be useful, especially for longer levels. Manual structuring of basic moves when solving complicated levels would get repetitive and dull. By making sure that the user doesn't have to perform every step, it will keep enjoyment high as simpler moves will be automated.

Some users commented that dragging the buttons straight away would be a more natural approach rather than clicking the button and then having to move the cursor to pick up the item. A future feature could be that the buttons are on the canvas and that this does indeed happen. When they click down on the button, users will be able to drag straight from that location to give them a more intuitive feel.

By having the Natural Deduction proof system in place, it allows further opportunity for a larger variety of logic systems. Firstly, more complicated levels could be created with the intuitionistic logic. Once a solid set of levels has been created for this, the software has the potential to introduce other logical systems like Classical logic. A step to realising this feature is allowing users to create their own levels. If the user can create and save their own

levels, it makes the software more open and allows greater use of the software by more experience logicians. Only a finite number of logical symbols would need to be permitted for a large range of levels to be created in a large amount of logics systems. Symbols like the existential quantification (there exists) and turnstile (provable) would needed to be added so that the user can create more complicated levels.

Chapter 8

Conclusion

This project has demonstrated how users can learn mathematics using a game. The learning tool created fuses together ideas from Mathematical logic and Gamification principles. The Mathematical theory is central to the project and throughout it encourages users of the software to learn. The aim of the game is to teach users about the Natural Deduction proof system. Educational values were held important throughout the duration of this work and therefore it was necessary to ensure that Natural Deduction knowledge was absorbed through the medium of game. The author wanted to create software which offered something that was not currently available elsewhere and that users which had little knowledge of Natural Deduction proof systems found educational and fun.

The culmination of research studied in Chapter 2, in topics of gamification and Mathematical education, has inspired the software that has been designed and implemented in this project. The inspiration has mainly come from how Kinaesthetic learning techniques can be effective in the learning process and how gaming principles can be integrated with education.

There is no current software available which offers the balance of learning and Natural Deduction Mathematics that this project offers. Some current Educational games are mentioned in Section 2.1.3 and also Natural Deduction resources are looked at in Section 2.2.2. The software created in this project is unique as it takes aspects from both to create a brand new educational game. The game delivered has improved the educational Natural Deduction resources available and explored whether demand is there for a engaging Natural Deduction learning tool. The preliminary study suggests that the software can be effective at teaching users about the Natural Deduction proof system.

Software has successfully been created to a point where user testing has

been done and the results analysed. A Javascript game is currently hosted online which users can make use of to learn about Natural Deduction proof systems.

The initial feedback from the results of the Alpha testing have been highly encouraging. User feedback suggests that the software achieves its main aim of being an effective Natural Deduction educational resource. In general they thought that game was a better resource to learn from than learning through traditional methods like classroom sessions.

Also found in the survey was that users felt they knew more about Natural Deduction after playing the game, giving encouraging signs that this software can be used as a valuable learning tool for students.

Usability of the game received a mixed response. There were few comments on the drag and drop feature suggesting that this was effective but there was comment that some users stopped because of usability issues to do with the lack of ability to undo moves made.

One of the other key results is how fun the users found the game. This is important because the software is supposed to be a more enjoyable way to learn about Natural Deduction than current resources available. Users indicated that they were having an average amount of fun. I believe that an improvement in usability will have a positive effect on the amount of fun the user will have.

Overall I am very proud of the achievements I have accomplished during this project. I am proud of the software created and felt it was good enough to let users test it. I have found, in conjunction with the results, that the game created has a solid framework which only need minor tweaks to make it more effective as a Natural Deduction learning resource. This project has been a great learning experience both personally and in the topic that has been studied. In my opinion, the link between games and education will grow stronger as people grow up with electronic devices and so if educational tools can be effectively provided in the form of games, then they can have a great effect. Whilst this study is not broad enough to draw any research conclusions, I am confident that this software is suitable for enabling users to learn from it.

Bibliography

- Aponte, M.-V., Levieux, G. & Natkin, S. (2009), Scaling the level of difficulty in single player video games, in 'Entertainment Computing-ICEC 2009', Springer, pp. 24–35.
- Arthur, R. T. (2011), *Natural Deduction: An introduction to logic with real arguments, a little history and some humour*, Broadview Press.
- Bootstrap (2016), 'Bootstrap · the world's most popular mobile-first and responsive front-end framework.', <http://getbootstrap.com/>. (Accessed on 04/04/2016).
- Broda, K., Ma, J., Sinnadurai, G. & Summers, A. (2007), 'Pandora: A reasoning toolbox using natural deduction style', *Logic Journal of IGPL* **15**(4), 293–304.
- Chen, N.-S., Wei, C.-W., Wu, K.-T. & Uden, L. (2009), 'Effects of high level prompts and peer assessment on online learners' reflection levels', *Computers & Education* **52**(2), 283–291.
- Chen, Y.-L. et al. (2009), Data flow diagram, in 'Modeling and Analysis of Enterprise and Information Systems', Springer, pp. 85–97.
- Chung, L. & do Prado Leite, J. C. S. (2009), On non-functional requirements in software engineering, in 'Conceptual modeling: Foundations and applications', Springer, pp. 363–379.
- Collett, N. S. . K. (2013), 'Video games and wellbeing', <https://mindfulnessinschools.org/wp-content/uploads/2013/09/video-games-and-wellbeing.pdf>. (Accessed on 04/05/2016).
- Crockford, D. (2001), 'Javascript: The world's most misunderstood programming language', <http://www.crockford.com/javascript/javascript.html>. (Accessed on 04/05/2016).
- Crockford, D. (2008), *JavaScript: The Good Parts: The Good Parts*, "O'Reilly Media, Inc."

- Dempsey, J. V., Driscoll, M. P. & Swindell, L. K. (1993), ‘Text-based feedback’, *Interactive instruction and feedback* pp. 21–54.
- Deveria, A. (2016), ‘Can i use... support tables for html5, css3, etc’, <http://caniuse.com/#feat=canvas>. (Accessed on 04/10/2016).
- Garris, R., Ahlers, R. & Driskell, J. E. (2002), ‘Games, motivation, and learning: A research and practice model’, *Simulation & gaming* **33**(4), 441–467.
- Gasquet, O., Schwarzentruher, F. & Strecker, M. (2011), Panda: a proof assistant in natural deduction for all. a gentzen style proof assistant for undergraduate students, in ‘Tools for Teaching Logic’, Springer, pp. 85–92.
- Gentzen, G. (1964), ‘Investigations into logical deduction’, *American Philosophical Quarterly* **1**(4), pp. 288–306.
URL: <http://www.jstor.org/stable/20009142>
- González, C. & Area, M. (2013), Breaking the rules: Gamification of learning and educational materials, in ‘Proceedings of the 2nd international workshop on interaction design in educational environments’, pp. 7–53.
- Halbach, V. (2014), ‘Introduction to logic: Natural deduction’, <http://logicmanual.philosophy.ox.ac.uk/vorlesung/logic6p.pdf>.
- Impact, L. (2016), ‘On demand load testing for developers & testers — load impact’, <https://loadimpact.com/>. (Accessed on 04/10/2016).
- Indrzejczak, A. (2016), ‘Natural deduction’, <http://www.iep.utm.edu/nat-ded/>.
- Inflectra (2015), ‘Software testing methodologies - learn the methods & tools’, <https://www.inflectra.com/Ideas/Topic/Testing-Methodologies.aspx>. (Accessed on 04/04/2016).
- Jaśkowski, S. (1934), *On the rules of suppositions in formal logic*, Nakładem Seminarjum Filozoficznego Wydziału Matematyczno-Przyrodniczego Uniwersytetu Warszawskiego.
- Koster, R. (2013), *Theory of fun for game design*, ” O’Reilly Media, Inc.”.
- Laboreo, D. C. (2005), ‘Introduction to natural deduction’, https://homepage.univie.ac.at/christian.damboeck/ps06/clemente_nat_ded.pdf.
- Litten, J. (2010), ‘How to drag and drop on an html5 canvas’, <http://html5.litten.com/how-to-drag-and-drop-on-an-html5-canvas/>. (Accessed on 04/05/2016).

- McDonald, M. & Hull, C. (2012), ‘Different ways of learning’.
- Montemayor, E., Aplaten, M., Mendoza, G. & Perey, G. (2009), ‘Learning styles of high and low academic achieving freshman teacher education students: an application of the dunn and dunn’s learning style model’, *University of Cordilleras* **1**(4), 1–14.
- Muntean, C. I. (2011), Raising engagement in e-learning through gamification, in ‘Proc. 6th International Conference on Virtual Learning ICVL’, pp. 323–329.
- MyMaths Website* (2016), <http://www.mymaths.co.uk/>.
- Nixon, R. (2012), *Learning PHP, MySQL, JavaScript, and CSS: A step-by-step guide to creating dynamic websites*, ” O’Reilly Media, Inc.”.
- nptelhrd (2015), ‘Mod-01 lec-42 natural deduction in predicate logic’, <https://www.youtube.com/watch?v=mJl9t1Pz5TU>.
- Ofcom (2015), ‘The uk is now a smartphone society’.
URL: <http://media.ofcom.org.uk/news/2015/cmr-uk-2015/>
- Oprescu, F., Jones, C. & Katsikitis, M. (2014), ‘I play at work—ten principles for transforming work processes through gamification’, *Frontiers in psychology* **5**.
- Pelletier, F. J. (1999), ‘A brief history of natural deduction’, *History and Philosophy of Logic* **20**(1), 1–31.
- PhilHelper (2013), ‘A crash course in formal logic pt 8a: Natural deduction in propositional logic’, https://www.youtube.com/watch?v=ZebpxExsY_0.
- Pritchard, A. (2013), *Ways of learning: Learning theories and learning styles in the classroom*, Routledge.
- Sims, Z. & Bubinski, C. (2013), ‘Codecademy website’, <https://www.codecademy.com/>.
- Smith, P. (2010), ‘Proof systems’, <http://www.logicmatters.net/resources/pdfs/ProofSystems.pdf>. (Accessed on 17/02/2016).
- Spolsky, J. (2001), *User interface design for programmers*, Apress.
- Suhonen, K., Vääätäjä, H., Virtanen, T. & Raisamo, R. (2008), Seriously fun: Exploring how to combine promoting health awareness and engaging gameplay, in ‘Proceedings of the 12th International Conference on Entertainment and Media in the Ubiquitous Era’, MindTrek ’08, ACM, New York, NY, USA, pp. 18–22.
URL: <http://doi.acm.org/10.1145/1457199.1457204>

- Sutcliffe, G. (2009), ‘The TPTP Problem Library and Associated Infrastructure: The FOF and CNF Parts, v3.5.0’, *Journal of Automated Reasoning* **43**(4), 337–362.
- TSUKADA, M. (2001), ‘Proof checking using prolog’, *Institute for Mathematical Sciences Kokyu proceedings* **1186**, 78–83.
- UKIE (2014), ‘The games industry in numbers’, <http://ukie.org.uk/research>.
- Van Dalen, D. (1986), Intuitionistic logic, in ‘Handbook of philosophical logic’, Springer, pp. 225–339.
- Vogel, L. (2009), ‘Introduction to java web development - tutorial’, <http://www.vogella.com/tutorials/JavaWebTerminology/article.html>. (Accessed on 04/05/2016).
- Wieggers, K. & Beatty, J. (2013), *Software requirements*, Pearson Education.
- Wood, L. E. (1997), *User interface design: Bridging the gap from user requirements to design*, CRC Press.
- Wu, P.-H., Hwang, G.-J., Milrad, M., Ke, H.-R. & Huang, Y.-M. (2012), ‘An innovative concept map approach for improving students’ learning performance with an instant feedback mechanism’, *British Journal of Educational Technology* **43**(2), 217–232.
- Zichermann, G. & Cunningham, C. (2011), *Gamification by design: Implementing game mechanics in web and mobile apps*, ” O’Reilly Media, Inc.”.

Appendix A

User Testing Results

All have 15 responses unless otherwise stated. Numbers in brackets are the number of participants with that answer.

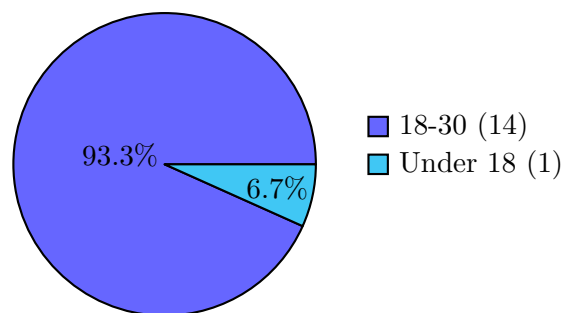


Figure A.1: What is your age?

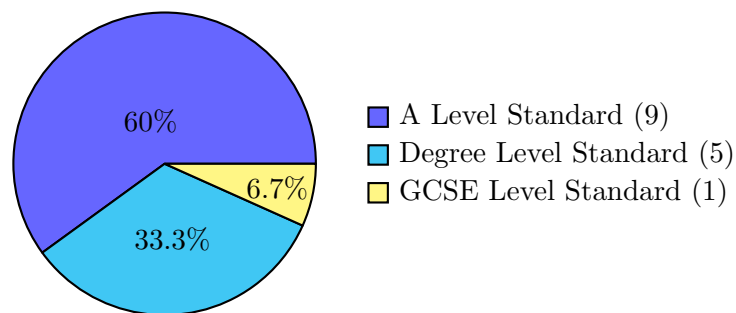


Figure A.2: What is your mathematical ability?

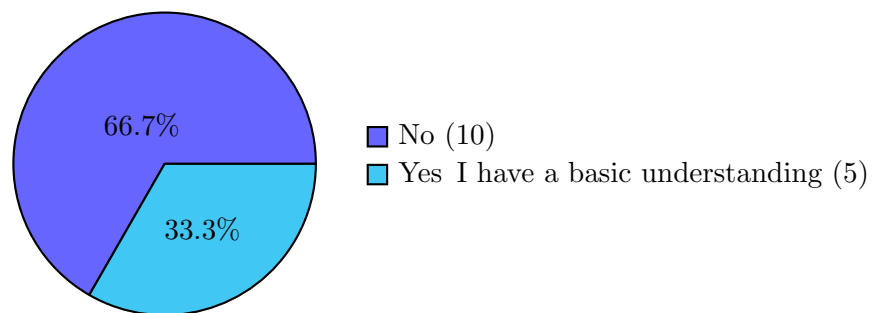


Figure A.3: Have you used Natural Deduction logic prior to playing the game?

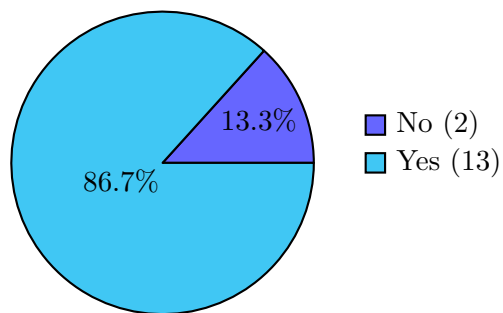


Figure A.4: Do you think you have learnt more about Natural Deduction as a result of playing the game?

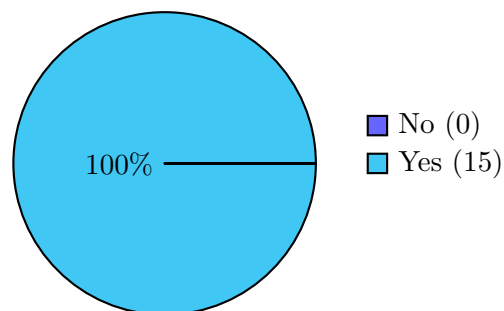


Figure A.5: Did you do the tutorial at the start of the game?

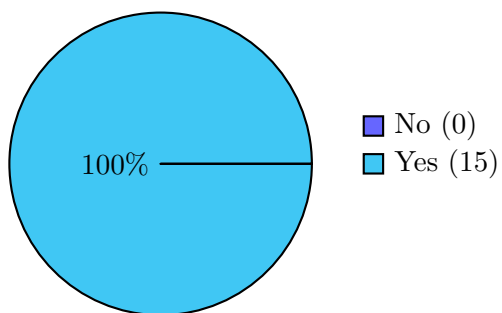


Figure A.6: If you answered Yes, did you complete the tutorial?

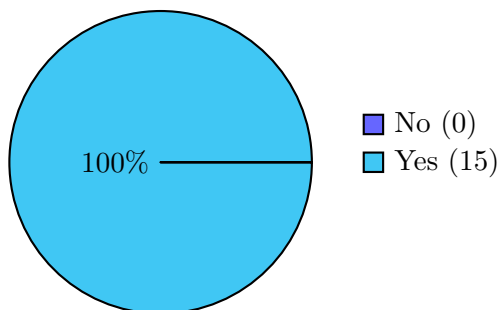


Figure A.7: If you answered Yes, did you find the tutorial useful?

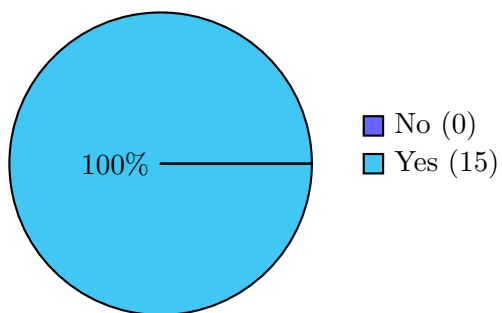


Figure A.8: Did you use the hints button during the game?

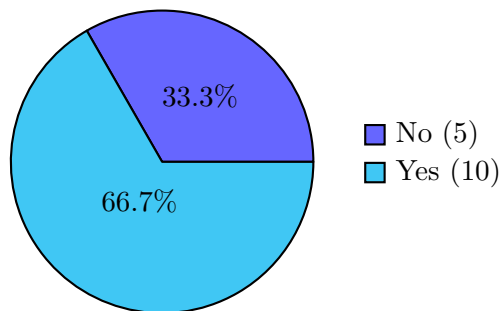


Figure A.9: If Yes to using hints, did you find them useful?

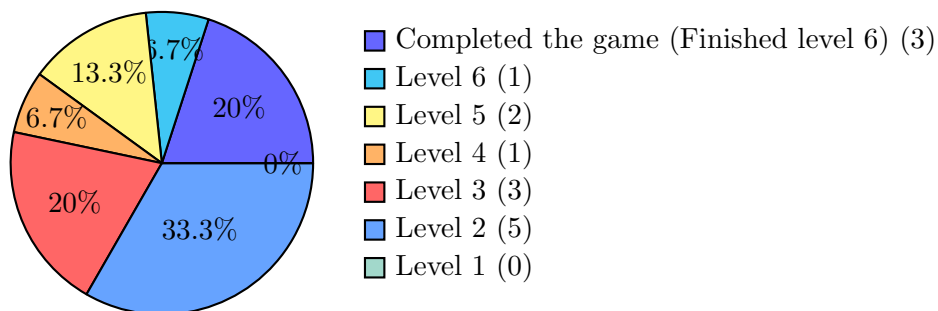


Figure A.10: Which level of the game did you get to?

I got frustrated with the proof! I knew what I wanted it to look like but I kept doing it wrong the proofs kept getting placed on the wrong dots from where I wanted them.
It took me a while to create a proof and then it was wrong and I couldn't be bothered to create it all again from scratch
could not work out how to complete the level correctly
Couldn't answer Question 2, as I have no idea what proof it is wanting or how to prove it
Because I was watching Big Bang Theory :)
Time constraints.
Not enough knowledge of proofs or time
Time consuming
Conjunction elimination is poorly explained, I dont understand what I should be doing what so ever

Figure A.11: If you didn't finish the game, why not?

Yes
Levels increased nicely
Thought level 2 got too difficult too quickly, there should have been an example to show how to put an extra 2 top 1 bottom in to prove the whole thing
Very well
Good curve
found level 2 a lot harder than 1
The difficulty increased quickly for a novice
Yes, the difficulty increased.
Yes checked internet for more info
Not really but maybe it would for the last 2
Exponentially
definitely
First 2 levels were easy, level 3 made no sense, so I guess there was a difficulty curve
Yes, difficulty increased as game progressed but not too quickly, just the right amount.

Figure A.12: How did you feel the difficulty curve increase as the game progressed? (Did the difficulty of the levels increase as the game went on?)

Minimalistic design
Drag and drop capability
The duplicate option! (when it worked)
Though provoking but logical
Colours
allows you to think logically and it is interactive
Good user interface
Tutorial explained the concepts of the game well
Simplicity
You can assemble your answer
It let you work things out yourself
The challenge
i distracted me from my work, like all good games
Nice drag and drop stuff that is mostly smooth, at times there are issues with item selection.
Mind puzzle / brain-teaser type game

Figure A.13: What did you like most about the game?

no
no
No major issues, but it caused chrome to hog a lot of memory and processing power
i couldn't undo an action when i did something wrong so i had to bin the whole thing and start again
For complicated proofs it was so frustrating when I had done somethign wrong there was no undo button / i couldnt drag the letter back out again!! I did this on every single level. Level 4 I put A on top of A, and it said proof incorrect- check A. I Know this is meant to help the user but it wasnt very helpful!! Maybe saying "the top A" would have been better. I didnt understand why a should be on top of b in the implication bit, it didnt seem to work like that in the game
Not really
No
Couldn't answer it, still don't know how to proceed and the hints are just confusing
Apart from my lack of knowledge, no
Size of game to fit the browser required zooming out as forewarned.
yes
It was annoying having to delete everything and start again if something was in the wrong place
My answer at the end wasn't accepted even though it should be right, I just didn't write it in the 'correct' way
Drag and drop issues, poor explanations. removing items from a structure would also have been useful, I found myself creating large structures and then having to bin them, rather than tweak them.
Small bugs

Figure A.14: Did you have any problems whilst playing the game?

No
No
No
no
no
Potentially fairly resource intensive, but nothing major
% signs in my incorrect message
Not yet
No.
see notes
I don't know whether this is a bug but it didn't accept the answer was right if the parts were in the wrong order
Didn't show me an image in the hints (some overlap of writing?)
"something wrong with undefined..?" on level six, the hint didnt appear for some reason, so i had to use the hint and lose point even though i hadnt actually seen it before the level as intended.
Not that I know of.
Some of the pictures did not show up

Figure A.15: Did you find any bugs in the code?

Mobile compatibility is a must, which this currently doesn't support well. Laptop/desktop personal computing is in decline - this really should be touchscreen compatible.
An 'undo' button to take back the last step. Dragging straight from the button set instead of pressing and then dragging in the canvas.
When you click a button make the object go onto the canvas at the bottom above the button to make it easier to use straight away. Also, brackets could be another object dragged onto stuff, dragging the object onto the brackets was a bit strange when the rest isn't like that. Red outline when deleting something might be clearer than blue? I didn't know how I was losing points, the user isn't told how points are given out at all. Delete all button next to check proof, nooooooo! My beautiful long proof! When hovering a proof over a dot, it would be useful for the underneath part to be expanded to fit my on top proof in- I ended up putting stuff in the wrong place because I thought it was over one dot when actually it wasn't at all
It would be nice if you could detach objects (like A, B, etc.) instead of having to delete it and start over everytime you make a mistake
Display the rules
be able to undo once you have put two parts together
It might be useful to have an undo button, make it all fit on the screen so you can't scroll
Zoom thing / Undo / novice tutorial
Reduction of the game size? As some potential users, younger or older, may be unaware on adjusting browser size.
see notes
The controls could be easier
More explanations
a copy button? less white space on terms the larger they get.
Better explanations, the ability to tweak current proofs, a better hint system that gives you a hint in to how the structure should be. For example, if the hint actually shown me how to do level 3, I may then have understood what was going on. I also found myself clicking the buttons at the bottom and trying to drag items from there, that was a possible usability issue.
Allow proofs (2 top/ 1 bottom etc) to be duplicated

Figure A.16: What improvements do you think could be made to the game?

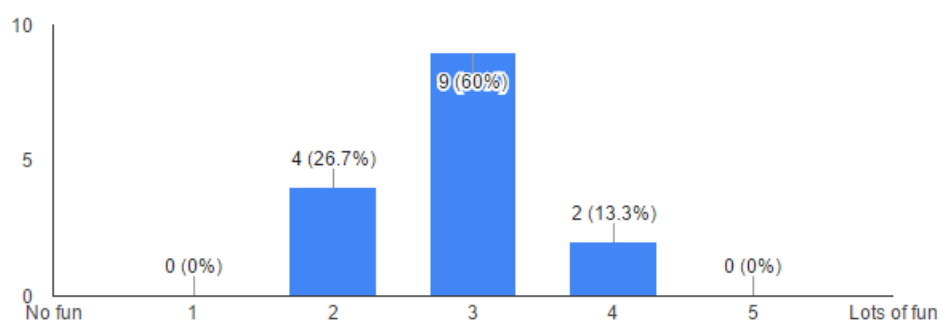


Figure A.17: How fun did you find the game?

I'm not hugely interested in maths, nor particularly good at it! I'm sure my opinion is no reflection on how good the game is for those who are!
There's only so much fun you can find maths.
I found it really engaging because I wanted to keep going and reach more levels and complete the proofs and learn more rules, but the usability issues were really frustrating and it took me a long time to construct the proof for level 5 which I then did wrong because I misplaced one part of it. Starting again would have been frustrating so I stopped then.
Trying to solve the problems was fun but it was tedious to have to scroll down the screen to create the buttons each time and remake everything from scratch when an error was made
Fiddly to input new formulas
found it hard
Ew, maths
Game seems education orientated.
Satisfaction when got the answer. Alot of trial and error
It was quite good but it's still doing proofs which are boring
ew maths
Didn't get far enough to appreciate it, it also took far too long to move items around, hindering the fun.

Figure A.18: Give a reason for your answer...

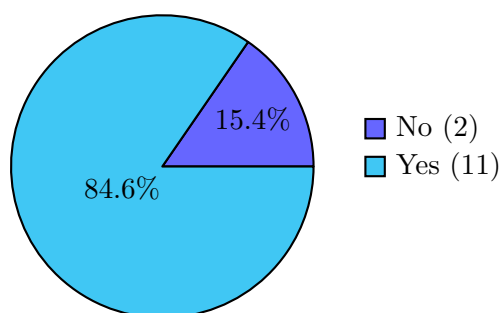


Figure A.19: Do you think that playing this game is a better way to learn than learning in traditional methods? (E.g. Pen and paper/ Lectures/ Classroom session) (13 Responses)

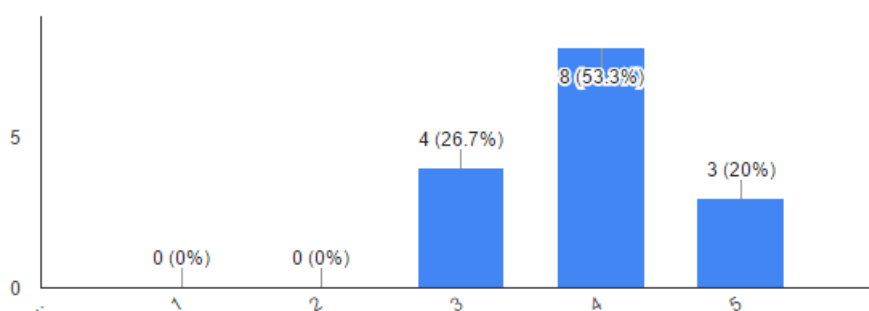


Figure A.20: As an educational tool, how effective do you feel the game is?

Perhaps add in a 'reveal answer' button if the question is too hard and the user can't figure out the answer.
It's much more fun than learning from a book or in a lecture!
I don't think the hints were really "hints" they were more "these are the definitions for the techniques", I think they should be displayed on the screen the entire time and hints should tell you how to go about completing the level
Still not sure what Natural Deduction is

Figure A.21: Any other comments...