A Serious Game for Teaching First Order Logic to Secondary School Students

Amy Hooper

Bachelor of Science in Computer Science with Honours
The University of Bath
2016-2017

A Serious Game for Teaching First Order Logic

Submitted by: Amy Hooper

COPYRIGHT

Declaration

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

Signed:

❑ **Retain for Sample** **Mark**

UNIVERSITY OF BATH

Department of Computer Science

INDIVIDUAL COURSEWORK
Submission Cover Sheet

Please fill in both columns in BLOCK CAPITALS and post into the appropriate Coursework Submission Box.

for office use
Date and time received

This section will be retained by the Department Office as confirmation of hand-in

How to present your work

1. Bind all pages of your assignment (including this submission sheet) so that all pages can be read by the marker without having to loosen or undo the binding. Ensure that the binding you use is secure. Missing pages cannot be marked.

2. If you are required to submit part of the work on a disk, place the disk in a sealed envelope and bind the envelope into the submission.

You must keep a copy of your assignment and disk. The original is retained by the Department for scrutiny by External Examiners.

Declaration

I certify that I have read and understood the entry in the Department of Computer Science Student Handbook on Cheating and Plagiarism and that all material in this assignment is my own work, except where I have indicated with appropriate references. I agree that, in line with Regulation 15.3(e), if requested I will submit an electronic copy of this work for submission to a Plagiarism Detection Service for quality assurance purposes.

FAMILY NAME	FAMILY NAME
GIVEN NAME	GIVEN NAME
UNIT CODE	UNIT CODE
UNIT TITLE	UNIT TITLE
DEADLINE TIME & DATE	DEADLINE TIME & DATE
COURSEWORK PART (if applicable)	COURSEWORK PART (if applicable)
SIGNATURE	SIGNATURE

Abstract

This dissertation investigates the teaching of first order logic, which is normally taught at university level, to A-level students. A serious game, which is a game designed for a specific purpose, is developed to teach this topic. Serious games for teaching first order logic exist, but none are aimed at school-aged students.

This project describes the development of the serious game, and evaluates its effectiveness with a group of A-Level students. The game was found to be successful at teaching first order logic to adults with a range of mathematical backgrounds and to A-level students. This demonstrates the value of serious games and gamification in teaching abstract mathematical topics to this user group.

Contents

List of Figures

List of Tables

Acknowledgements

I would like to thank my supervisor, Dr Willem Heijltjes for his guidance during the project.

As always, I am grateful for the love and support of my Mom, my Dad, Matthew and the rest of my family during my studies.

Chapter 1

Introduction

1.1 Motivation

1.1.1 Problem Description

In 2014, the National Curriculum in the UK was changed to introduce Computing for the first time. Students as young of five are to be introduced to writing and debugging simple programs, and students are expected to be able to program in two languages before leaving Secondary School (Department for Education, 2013). These changes were welcomed by many people, but concerns have been raised on how Computing should be taught in schools by teachers with little or no experience teaching the subject (Brown et al., 2014).

Many pedagogical applications exist for teaching Computer Science, but most of these focus on programming (Brown et al., 2014). There are also successful web-based applications for teaching mathematics in schools. Despite this, there is little research into Computer Science education within schools, possibly because the discipline is much younger than most school subjects (Armoni, 2011). With the changes to the Curriculum, it is likely that there will be more demand for teaching resources aimed at teaching the theoretical side of Computer Science, including logical reasoning.

1.1.2 First Order Logic

First order logic is one system of formal logic that students may learn. It uses predicates, relations, objects, quantifiers and variables to express facts that can be evaluated as true or false. Logical thinking and abstraction is an important part of learning Computer Science (Computing at School Working Group, 2012). Teaching first order logic also helps to familiarise students with mathematical notation such as universal and existential quantifiers. This is especially important for school students who go on to study Computer Science at

University, because lack of familiarity with notation has been found to be a key reason why some undergraduates within Computer Science struggle with mathematical units (Fung et al., 1994).

1.1.3 Serious Games

A serious game is a game that is developed to impart an educational benefit (Glover, 2013). Serious games have been proven to be a successful tool in education (de Aguilera and Mendiz, 2003), increasing learning and student motivation (Mildner et al., 2015). The development of a serious game for the teaching of first order logic will provide an additional tool for the teaching of Computer Science in schools. Investigation and user testing of the finished product will help assess the potential for serious games to assist in school students' learning of abstract logical reasoning skills.

1.2 Aims

The primary aim of this project is to develop a serious game that employs visualisation and gamification techniques to teach first order logic. The project also aims to evaluate its effectiveness, particulary for teaching secondary school students.

1.3 Structure

Literature Review The dissertation begins with a review of the literature on serious games and gamification; existing serious games used in mathematics and computer science education; and existing methods for teaching first order logic.

Requirements This research then informs the development of the requirements specification, which describes how the requirements for the game were elicited.

Design The Design chapter details the development of the requirements into designs for the game, covering the design of the levels, the user interface and the internal architecture.

Implementation The Implementation chapter describes decisions that were made connecting to the development of the system, and the results of the earliest tests on the prototype game.

Testing The Testing chapter describes how the implemented game was tested with two different user groups, and evaluates the results of this testing.

Evaluation The test results, as well as reflections on the rest of the project, go on to inform the evaluation of the game and of the project as a whole.

Chapter 2

Literature Survey

2.1 Introduction

This project proposes the development of a serious game to teach first order logic to secondary school students. Gamification is a growing area of research and serious games for use in education are becoming much more widespread. This literature review therefore has a number of areas to draw from in order to inform the design of the proposed game.

Firstly, serious games will be defined and their key features identified. Existing serious games will be analysed to identify features and techniques that have been successful in the past. Relevant serious games fall into two categories. The first are games that teach Computer Science and Mathematics within secondary schools. The second category is games and other resources used to teach logic and closely related subjects at any level. Games in these areas that have been shown to be successful and/or have achieved widespread popularity will be useful for informing the design of the proposed project.

The review will finish by examining existing pedagogy for teaching first order logic and closely related subjects.

2.2 About Serious Games

2.2.1 Definition

There is no universally agreed upon definition of a serious game (Dörner et al., 2016, p.3). Malone (1981) defines a game as having three key characteristics: challenge, fantasy and curiosity. Charsky (2010) adds rules, choices and competition and goals to this list.

Glover (2013) defines a serious game as a game that is developed to impart an educational benefit. Dörner et al. (2016) provide a slightly broader definition, defining a serious game to be a game that attempts to both entertain and achieve another goal, with the additional goal being the characterising goal. Here, the game does not explicitly have to be for learning, but could have another aim, such as improving health. Charsky (2010) refines the definition further, by insisting that a serious game must encourage the development of 'higher order thinking skills' and not just employ drill like activities. This provides a distinction between serious games and 'edutainment', which is where an application employs a simple reward mechanism for the completion of rote learning or simple tasks. Serious games, therefore, must encourage some kind of more complex thought process or inspire deeper learning.

Gamification refers to the addition of game-like features to existing contexts (Deterding et al., 2011) (Dörner et al., 2016). This is normally achieved by adding simple features, such as points and leaderboards; adding a story or theme; tracking achievements or adding challenges or level progression (Hamari et al., 2014).

2.2.2 Serious Games for Education

Games have been proven to be useful tools in education which can aid students in the development of a range of skills (de Aguilera and Mendiz, 2003). Games are a useful tool in education for three main reasons: they are accessible; they are engaging; and they are able to offer instant feedback.

A well designed educational game will be accessible to all of the target user group. Other educational tools, such as textbooks, often require the user to have background knowledge. Within a game, instructions can be provided and a player's options restricted so as to make it easy for a player to make a move. Learners can also repeat and practice a game for as long as they need to, rather than rely on a lesson plan scheduled for a whole class.

Engagement is another important advantage of games for education. A well-designed educational game will be enjoyable, and inspire in the user a desire to keep playing. The popularity of video games means that digital serious games are a good format for generating engagement (de Aguilera and Mendiz, 2003, p.2). Malone and Lepper (1987) state that there is a strong link between motivation and intrinsic learning, so choosing an educational tool that engages and motivates users is an important factor in how effectively people learn. When students are intrinsically motivated, they may spend more time and effort learning, and may be more likely to use the acquired knowledge in the future (Malone, 1981). In one study, Mildner et al. (2015) found that gamification of a quiz application increased user enjoyment; likelihood of playing again and for longer; and perceived learning. This held even though the game elements took time away from answering the questions in the quiz.

Instant feedback is also an important feature of educational games. Users can be told instantly whether or not they have been successful by the program, without needing to get their work checked by a teacher or tutor. Games can be responsive to the needs of a user more easily than traditional teaching methods. Progressions to the 'next level' or the higher levels of difficulty can be decided by user performance. User performance can also be used to inform the behaviour of the game: if the user is repeatedly succeeding at a task, then the difficulty of the game can be increased; if the user is failing tasks or not completing them quickly enough then the game can offer hints, introduce a tutorial or reduce the difficulty of the tasks. New information can be given to the user as and when they require it, reducing the load on memory (Gee, 2003). This was found to be successful by Mildner et al. (2015), who found that adding an algorithm that chose the next question based on past player performance increased both enjoyment and perceived learning for a quiz application.

2.2.3 Assessing a Serious Game

The evaluation of an educational tool or method is usually done based on learning objectives, which are determined before the tool is designed (Hainey et al., 2012). A learning objective is a statement about something a student should know or should be able to do (European and Culture DG., 2008).

Defining Learning Objectives

Defining learning objectives for Computer Science may be particularly difficult, because there is little agreed consensus on what students should be taught at what age. This is highlighted by Kunkle and Allen (2016), who present a research project into an educational assessment tool designed to assess the effectiveness of teaching in several first year undergraduate programming courses. Their study includes a description of the difficulties in designing a tool to assess knowledge in Computer Science, because of the lack of standards for Computer Science curricula.

2.3 Serious Games in Computer Science

A number of serious games and other interactive applications have been developed to teach Computer Science within schools.

2.3.1 Scratch and Lego Mindstorms

One of the most successful computer science education applications is Scratch, a drag-and-drop programming language. School students are reported to find Scratch highly motivating (Resnick et al., 2009, p.66). Maloney et al. (2010) analyse Scratch and list its

key attractions. These include 'tinkerability', its user interface design and the way in which is encourages self-directed learning. Also praised is its lack of error messages - the environment simply only permits sensible combinations of blocks, which Maloney et al. compare to real life play. The Scratch project also includes a website where users can share their projects, which currently has just under 18 million uploads (Lifelong Kindergarten Group, 2016). The average age of participants is just 12 years old (Resnick et al., 2009), showing that there is interest in computer science for younger people.

Another highly successful tool for teaching Computer Science is the LEGO Mindstorm, which is a small robot that can be programmed using a computer and have physical hardware added to it using LEGO blocks (LEGO Group, 2016). The Mindstorm is used to teach programming in schools, and has been shown to enhance student learning and engagement from the perspective of both students and parents (Melchior et al., 2005). The Mindstorm again uses play to enhance learning, with many students seeing it as a toy rather than an educational tool (Mauch, 2001).

Both Scratch and LEGO Mindstorms make use of techniques characteristic of gamification, such as facilitating creativity in a controlled environment. They are highly successful, being engaging and enjoyable for students whilst still having pedagogical value. Their popularity also demonstrates a willingness on the part of both students and teachers to use digital resources for Computer Science education. Their success as educational tools demonstrates that applications designed to be enjoyable can also achieve good educational outcomes.

2.3.2 Hamada's Integrated Environment for automata

Hamada (2008) presents a software package for teaching finite state automata and Turing machines to Computer Engineering undergraduates, which was later expanded to add pushdown automata (Hamada, 2009). Whilst the application does not involve gamification, it employs several features that are common in serious games: it first introduces a topic with a video tutorial that users can navigate themselves, and then expects the student to carry out certain tasks in a directed order; it makes heavy usage of visualisation to explain theoretical content; and its finite state machine and Turing machine simulators offer significant room for exploration and creativity. Hamada found that using the application significantly increased both student motivation and performance.

2.3.3 Isayama et al.'s Finite Automata Serious Game

Isayama et al. (2016) developed an application that uses gamification to teach finite automata to children aged 9-12. They employ puzzle-based learning in place of attempting to teach formal mathematical definitions (Isayama et al., 2016, p. 6).This approach aims to motivate students, who are expected to feel satisfaction upon correctly solving a puzzle.

Gamification is used to motivate students, with the aim of getting them to engage in the game for at least an hour. This claim is backed up by their results; 98% of a sample group of children reported that they found the game enjoyable, with some choosing to play the game even through their allotted break time (Isayama et al., 2016, p. 20).

They also use a 'walkthrough' to teach the concepts and how to play the game; the application provides hints whilst children interact with it, rather than expecting children to read long passages of instructions (Isayama et al., 2016, p. 9). Spreading out the delivery of the content reduces the load on memory, as described by (Gee, 2003).

Isayama et al. put their success partly down to the way they reduced the complexity of the topic from what is normally taught at undergraduate level to an appropriate level for school children. Notably, mathematical notation was completely omitted (Isayama et al., 2016, p. 6), as were more advanced topics such as nondeterminism (Isayama et al., 2016, p. 3). They also attribute some of their success to the design of their metaphor, which is how the abstract concepts were represented in-game (in this case, a robot acting based on a recipe). It is important that the metaphor chosen is intuitive and promotes engagement.

Isayama et al. also use the project as a way of assessing the feasibility of teaching theoretical computer science to younger children, recognising that Computer Science curricula within schools are going through rapid changes. Their findings were that the children played the game successfully, and that some were able to understand advanced concepts (Isayama et al., 2016, p. 21-22). That children were able to understand the concepts of finite automata at this age concurs with Eysink et al. (2001), who, after reviewing the work of educational theorists, state that the ability to think abstractly is likely to be developed at around this age.

2.4 First Order Logic

2.4.1 Introduction

First order logic sentences contain variables and predicates or relations, where a predicate says something about a property of the variable. Sentences are usually written as:

$$P(x)$$

where P is a predicate and x is a variable, or as

$$R(x, y)$$

where R is a relation and x and y are variables.

More complex sentences can be formed using the Boolean operators NOT $\neg$, AND $\wedge$, OR $\vee$, and implication $\Longrightarrow$ and the universal and existential quantifiers $\forall$ and $\exists$.

2.4.2 Language, Proof and Logic and Tarski's World

One of the key works that the proposed project will build on is the serious game Tarski's World. Tarski's World is a computer application that forms part of a courseware package called Language, Proof and Logic (Barwise and Etchemendy, 1993). It places various shapes on to a three-dimensional chess-board style grid and asks users to write sentences about these shapes using a first order logic based language (The Openproof Project, 2016).

Language, Proof and Logic Textbook

The Language, Proof and Logic package contains a textbook of the same name (Barwise and Etchemendy, 1993). Textbook chapters introduce logic topics and are supplemented by exercises within the Tarski's World game. The first topic to be introduced is atomic sentences, where examples are first given in English and then shown using predicate logic, for example:

$$A \textit{ is a } cube$$

becomes:

$$Cube(A)$$

The textbook then gradually increases new concepts, such as functions and boolean operators.

The Language, Proof and Logic package is aimed at either undergraduate or postgraduate study, with the first two chapters forming an introductory course in logic (Barwise et al., 2000, p.11), so my proposed game will involve less mathematical complexity. The textbook goes into great detail about a number of topics in mathematics and logic as appropriate for university level study, and Tarski's World contains enough operations and tools to facilitate exercises for all of these topics. Naturally, many of these are beyond the scope of the proposed project. As in Isayama et al.'s (2016) finite automata serious game, removing some of the mathematical complexity will likely be important for the game to be successful with a younger audience.

Isayama et al. also note the importance of reducing the amount of mathematical notation. For instance, Language, Proof and Logic uses over a page to clarify what effect parentheses would have on a first order logic sentence. Whilst this may be appropriate at University level, including notation that complicates the proposed game will increase the number of rules that users need to memorise in order to play the game, which may detract from student motivation and learning.

Tarski's World

Tarski's World is one of the serious games that is part of Language, Proof and Logic. (Barwise et al., 2000). It consists of a window containing three sections. The first is a three-dimensional chessboard style grid, called a 'World' that has three-dimensional shapes placed on it. The second is a list of sentences, written in a first order logic language. The third is a keyboard with logic symbols, such as quantifiers, to make these easier to input. The sentences are visualised immediately in the world.

Tarski's World is very flexible in its usage (Goldson et al., 1993). Students can create worlds themselves or generate them from a set of sentences. Likewise, sets of sentences can be created by the student, or they can use a set of built-in sentences. As part of an exercise, students may be asked to construct sentences or evaluate existing sentences as true or false based on the world they can see.

One of Tarski's Worlds key features is that it allows students to describe objects in the world directly using first order logic, rather than using translation into a written language (Barwise et al., 2000, p.14). In their review of the game, Goldson and Reeves (1994) particularly admire the way in which sentences can be visualised directly within the world, rather than first being translated into a spoken language.

Tarski's World provides feedback in three different ways: it determines whether a sentence is valid; whether it is syntactically correct and whether is the right answer. It does not attempt to figure out why someone went wrong (Eysink et al., 2001, p.7), but does demonstrate why an answer is incorrect.

In their review of eight games for teaching logical reasoning, Goldson et al. (1993) give particularly high praise to Tarski's World, describing it as flexible and interesting. However, they also note that the game does rely on the accompanying textbook, and as such, does not include any in-line help except for instructions on how to play the game.

2.4.3 Logic in Computer Science

Logic in Computer Science (Huth and Ryan, 2004*b*), is another textbook that teaches first order logic at University level. Like Language, Proof and Logic, it is also accompanied by computer based software, in the form of interactive exercises for each chapter (Huth and Ryan, 2004*a*).

The first chapter of the book introduces propositional logic, including the logical operators for negation, conjunction, disjunction and implication. Chapter 2 focuses on predicate

logic, including variables, predicates, quantifiers, terms and functions (the rest of the book deals with logic topics outside the scope of this project.) As in Language, Proof and Logic, these ideas are first introduced using illustrative sentences in written English, which are then stripped of additional detail and written using first order logic sentences.

The exercises themselves consist of multiple choice questions; when a user selects an answer, they are given immediate feedback. Users are told whether they are correct or incorrect and are given a short explanation of why this is. Most of the questions on propositional and predicate calculus involve translating English sentences to first order logic. Interestingly, this conflicts with the strategy employed by Barwise et al. in Language, Proof and Logic. Barwise et al. (2000, p.14) explicitly criticize this approach to learning first order logic, arguing that it is better to allow students to learn by directly interacting with objects in the language than to go via a spoken language - this was one of the motivations for the Tarski's World software.

The online exercises also differ from Tarski's World in that they are multiple choice questions that do not provide an opportunity for exploration. However, the provision of immediate feedback has been found to be useful. Like Language, Proof and Logic, this package differs from my proposed project in that it consists of an application as an auxiliary to a textbook, rather than a stand-alone. The exercises also do not employ gamification methods.

2.4.4 Organon

Organon is another web-based application to aid the teaching of logic within a university environment (Dostalova and Lang, 2011). It includes a range of logic topics, including first order and propositional logic. Its primary purpose is to familiarise students with formal notation and proof, but it also has features that allow course administration, such as the storing of students' grades on assessments.

Organon is split into three sections: a module of logic questions that can be used for practice or assessment; a grading module that can generate unique homework for individual students and grade these; and a 'practicing module'. The practicing module contains worked sample solutions and questions for students to complete. Questions require the student to show the steps in their working, so that the application can check these and provide more feedback than simply reporting whether the answer is correct or incorrect. Students can use the practice module in three ways: they can ask for a model solution; they can attempt to solve it themselves, with the application checking every step in their solution as they go; or they can attempt to solve independently and have the answer checked at the end.

As students complete tasks, the difficulty is increased. Dostalova and Lang follow a similar strategy to Mildner et al. (2015) in that the software is also able to choose the next problem based on a student's past performance. The application can recognise when a student repeatedly makes similar mistakes and can select an easier question that uses the problem technique.

Interestingly, Dostalova and Lang specifically state that the application facilitates the practicing of skills as drills. The use of gamification merely for the practice of drills is sometimes discouraged by some authors on serious games (Charsky, 2010). However, Dostalova and Lang argue that their application could help students learn the basics of logical manipulation faster, leaving more time for more advanced and interesting topics during the face-to-face time that students have with instructors.

2.4.5 Logic and Proofs

Logic and Proofs is an online introductory logic course developed by the Open Learning Initative at Carnegie Mellon University (Open Learning Initiative, 2015). It is intended either as a stand-alone course, or as a tool to accompany a taught lecture course. It includes a series of multimedia lectures on propositional and first order logic, and is accompanied by two 'virtual labs' which allow interactive learning.

Each chapter contains quizzes that provide automatic and immediate feedback to users. These include simple multiple choice questions which, like the quizzes that accompany Logic in Computer Science, immediately repost whether the user is right or wrong and provide a built-in explanation as to why this is.

One such lab is the 'Proof Lab' which is used for the construction of proofs. Like Organon, it gives immediate feedback as soon as the user appears to be heading in the wrong direction. It also attempts to encourage the user to try certain techniques first.

The online course, like the other tools for teaching logic, begins by explaining ideas like propositions and arguments in English, without use of special notation. Whilst the course is not a serious game, it employs game-like techniques, such as allowing independent exploration but in a controlled way.

The course was found to be successful in some (but not all) contexts (Schunn and Patchan, 2009). In one trial, it was found to possibly be responsible for decreasing the number of students who withdrew from a course and in another there was a significant increase in student performance and content learnt.

2.5 Summary

The review began with looking at definitions of serious games, and at how games can be used in education. The use of serious games in education has expanded dramatically in recent years, and the games themselves have improved in quality. Large-scale digital applications such as Scratch and the LEGO Mindstorms have had a wide uptake within schools, showing willingness on the part of teachers to employ digital applications for teaching Computing. Additionally, research done within both schools and universities on serious games developed for teaching Computer Science theory shows that such applications can benefit from gamification and produce successful results.

It was found that most existing resources for teaching first order logic were aimed at degree level. Most introduced logic in similar ways, by explaining the concepts involved using a written language and then introducing more formal notation, although they differed on what kind of exercises students were asked to complete. Many resources for teaching logic successfully employed game-like characteristics, such as instant feedback and interactivity, but only Tarski's World met the definition of a serious game fully. Despite this, the research on serious games, including games designed for teaching Computer Science within schools, shows that gamification can increase both student motivation and learning. It is reasonable to speculate that a serious game for teaching first order logic to secondary school pupils would be successful.

Chapter 3

Requirements

The requirements section begins with an overview of which sources were used to elicit requirements. It then lists requirements, and justifies their inclusion.

3.1 Sources for elicitation

The main source for requirements elicitation is the literature survey. The literature survey was split into two sections. The first, which looked at serious game and gamification research, is useful for informing the design of the interface and for informing the level progression. The second part of the literature review focused on existing tools, textbooks and courses for teaching first order logic. This section will be used to inform the pedagogical requirements of the game - what the game should be expected to teach. The literature review also looked at examples of interactive applications for teaching maths and computer science topics to Secondary School students - these examples can be used during the requirements gathering and in-depth design phases to inform the level of difficulty of the game and kind of language that the game should use.

3.2 Requirements and their Justification

The Requirements and their Justification section is split into several sections based on what aspect of design they relate to.

The requirements themselves are numbered, and justification for a specific requirement is given in italics immediately below it. Each requirement is also given a priority, with 1 meaning that the requirement 'must' be met, 2 meaning 'should' and 3 meaning 'may'. Requirements with priority 3 are requirements that would add value to the game, but may be omitted should the development need to be scaled down to keep within the scope of the

project.

3.2.1 Gamification Requirements

This section describes requirements elicited from the research into serious games and gamification conducted during the literature review.

1.1 – Must involve graphical visualisation of first order logic sentences.
Visual representations are useful for increasing motivation and demonstrating relevance to Computer Science students (Hamada, 2008).
Priority: 1

1.2 – Must start at an easy level of difficulty and allow the user to progress through increasingly complex levels.
The game should ensure that users are competent answering easier questions before moving on to harder questions. It is important the games gradually increase in difficulty in order to maintain the interest of the user (Dörner et al., 2016, p.11).
Priority: 1

1.3 – Must give immediate feedback if user is incorrect.
Instant feedback is important for allowing users to immediately assess their progress (Dörner et al., 2016).
Priority: 1

1.4 – May give more involved feedback to incorrect answers, attempting to guess where the user went wrong.
More detailed feedback would aid the user in finding the flaw in their reasoning and modifying it. This is one way of adapting a serious game to a user (Dörner et al., 2016, p.10).
Priority: 3

1.5 – Should introduce first order logic ideas and symbols gradually.
Introducing new ideas slowly will likely increase learning by reducing the strain on memory (Gee, 2003).
Priority: 2

1.6 - May require the user to both understand first order logic sentences and to write their own.
Researchers differ on what serious games should be used for. Some believe that games are better used to facilitate rote learning (Dostalova and Lang, 2011), and others believe that games should foster creativity and exploration (Charsky, 2010). Levels where users are

required to understand first order logic would be the former and levels where users have the freedom to design their own sentences would be the latter. Including both means that the game could incorporate both styles of learning.

3.2.2 Pedagogical Requirements

Description

This section describes requirements that specify what should be taught by the game. These requirements are gathered by looking at existing courses and introductory texts to first order logic.

Before an application for education can be developed, the learning outcomes must be established. The learning outcomes describe what a user should be expected to learn from playing the game (Hainey et al., 2012). Serious games are games developed with another purpose besides enjoyment in mind (Dörner et al., 2016); establishing learning outcomes specifies this purpose. The requirements in this section specify learning outcomes by detailing what the game should teach, i.e. what students should learn from it.

Designing pedagogical requirements involves resolving a tension between teaching the subject well and making a game that is engaging. The length of the game is limited both by the scale of this project and by the fact that making the game too long could make the game less engaging. As discussed in the literature review, existing courses on first order logic often take the form of several university level lectures (Barwise and Etchemendy, 1993; Huth and Ryan, 2004*b*; Open Learning Initiative, 2015); teaching the topic with this level of comprehensiveness within the game is not possible.

With a limit placed on the length of the game, a decision is needed on which aspects of first order logic to include. The decision on what to include draws from the problem description given in the introduction of this project. The aim of this serious game is to introduce the notion of First Order Logic and some basic sentences.

Requirements

2.1 – Must require user to understand sentences involving a predicate and an object.
This is the simplest sentence form possible.
Priority: 1

2.2 – Must require user to understand sentences involving relations.
Relations are important for telling a story with the game. Sentences consisting of a relation

between two objects are also amongst the simplest first order logic sentences.
Priority: 1

2.3 – Must require user to understand the logical connectives AND, OR and NOT.
Lack of familiarity with notation has been found to be a key reason why some undergraduates within Computer Science struggle with mathematical courses (Fung et al., 1994). One of the aims of the game is to introduce to secondary school students some of the notation that is used in mathematics courses at university. Introducing these logical connectives allows the introduction of the notation $\vee$, $\wedge$ *and* $\neg$.
Priority: 1

2.4 – May introduce the concepts of objects, predicates and relations formally.
There may be an explanation of first order logic that uses these terms. This is low priority because such an explanation may make the game too complicated or involve large blocks of text. The usability and enjoyability of the game should not be compromised too greatly.
Priority: 3

2.5 – Should require users to understand sentences using variables and quantifiers.
As in 2.3, the quantifiers $\exists$ *and* $\forall$ *are commonly used in university level mathematics courses. Variables are also an important concept in Computer Science.*
Priority: 2

2.6 - May give some background on the development of first order logic.
This would give some context as to why first order logic was important.

3.2.3 Non-Functional Requirements

This section describes non-functional requirements. Many of these are based on the characteristics found in successful serious games.

3.1 – Must not require complex installation or set-up.
A lengthy or tedious set-up process may harm user engagement and reduce enjoyment, which are two of the key advantages of games for education.
Priority: 1

3.2 – Must not have restrictive or expensive hardware and software requirements.
The game should be accessible to the whole target user group. Users must be able to run the game in their own time on their own computers; the learning should be as flexible as possible.
Priority: 1

3.3 – Should not take a long time to learn how to play the game.
As in 3.1, spending a long time learning the rules of the game is likely to damage engagement and enjoyment.
Priority: 2

3.4 – May work on devices without a keyboard
Functioning on tablet devices would further increase the application's accessibility. However, this needs to be balanced against the limitations this places on the rest of the design.
Priority: 3

3.5 – Should be enjoyable
Increasing motivation increases learning (Malone, 1981).
Priority: 2

3.6 - Should be accessible to users with a range of mathematical backgrounds.
One advantage of using a graphical representation of sentences is that users do not require prior knowledge of the mathematics or notation involved.
Priority: 2

Chapter 4

Design

4.1 Introduction

The design section begins by discussing the earliest decisions that needed to be made during the design phase, which includes which aspects of first order logic to teach and the choice of a story for the game to follow. Then, the earliest designs and prototypes are discussed. The design chapter finishes by discussing the design of the individual levels and the final design for the user interface.

4.2 Pedagogical Design - What to Teach

As discussed in the Requirements chapter, the topics to be taught are:

- Predicates
- Objects
- Relations
- The Boolean connectives, $\wedge$, $\vee$ and $\neg$
- Variables
- Quantifiers, $\forall$ and $\exists$

Most of the existing first order logic courses looked at in the literature review introduced new ideas in roughly the same order, beginning with objects, predicates and relations, then covering boolean operators and then variables and quantifiers (Barwise and Etchemendy, 1993; Open Learning Initiative, 2015). It seems reasonable to keep this order for the game.

4.3 Story

A story that users can progress through is one way of increasing motivation (Dörner et al., 2016, p.4). This game requires a story with characters that can interact with the world, so that sentences can be formed from a mixture of objects, predicates and relations.

Requirement 1.6 states that the game may provide some background on the development of first order logic. This requirement can be met by using this background as the story for the game. This choice of story allows the introduction of logicians and their theories as objects, properties of these objects as predicates and relationships between characters and theories as relations.

4.4 Input and Mechanics

Game mechanics refer to the ways in which users physically interact with the game and are an important part of game design (Dörner et al., 2016, p.12).

The input method that was decided upon was a drag-and-drop interface. Drag-and-drop can be made accessible to both mouse and touch-screen devices, which helps to meet requirements 2.2 (not to have restrictive hardware requirements) and 2.4 (to work on devices without a keyboard).

4.5 Prototypes

The first prototype game was developed using html and JavaScript. Each level has its own html front end and JavaScript logic page. The levels share a single First Order Logic JavaScript page, which, when implemented, will evaluate first order logic sentences and return true or false. The structure of this prototype game is given in the following sub-system diagram.

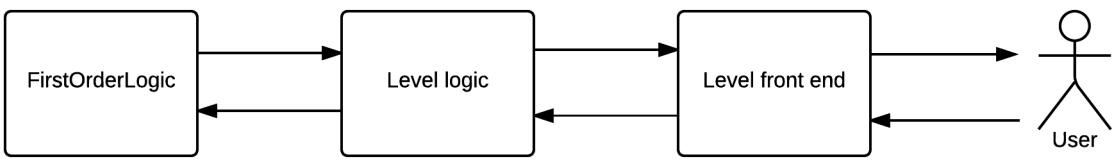


Figure 4.1: Sub-system diagram

4.5.1 Level Prototypes

The first prototype game included three different level designs. The first two required the user to read a sentence written using first order logic and drag and drop symbols into boxes in order to demonstrate understanding, and the third required users to write their own sentences.

The first level had single character and would be used to introduce predicates. The icons in the bottom row could be dragged to the box next to the person to assign a predicate to the person/object.

Figure 4.2: First level prototype

The second level had two characters and would be used to introduce relations. Users would be given a sentence with a relation and would drag arrows between the two characters in order to make the sentence true.

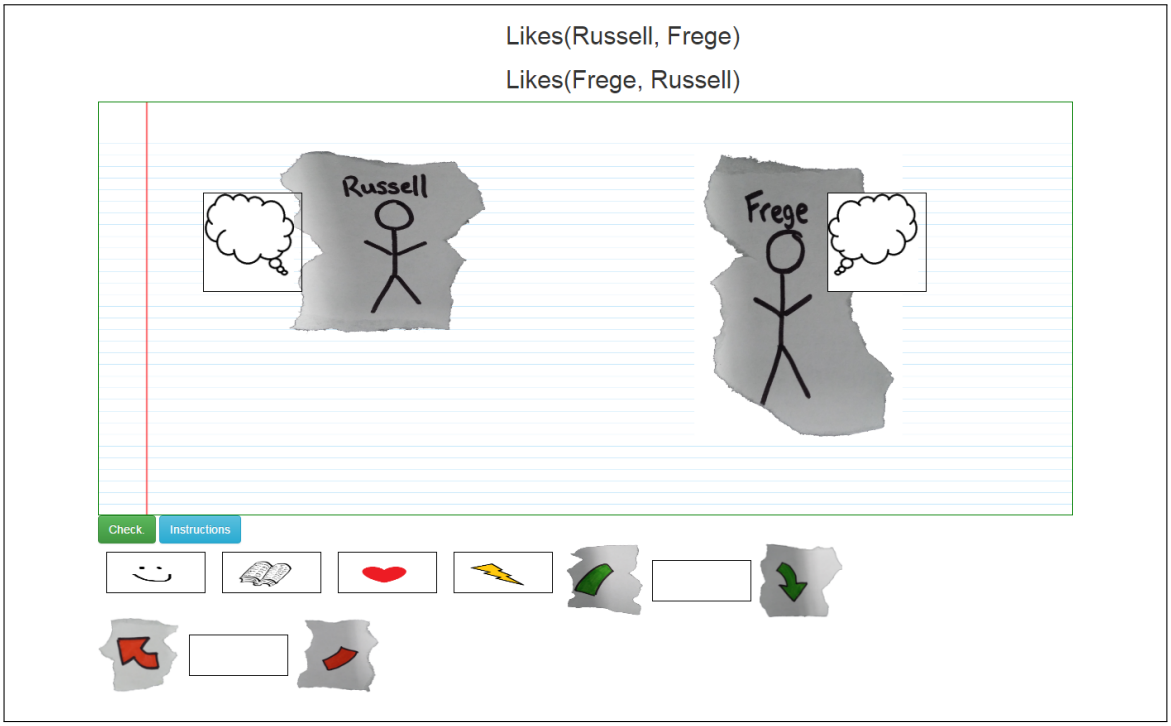


Figure 4.3: Second level prototype

The final level design had a different form of input. Instead of being given sentences and asked to manipulate the scene in order to make them true, users are given a fixed scene and asked to write their own sentences to describe it. An onscreen keyboard is provided for special characters.

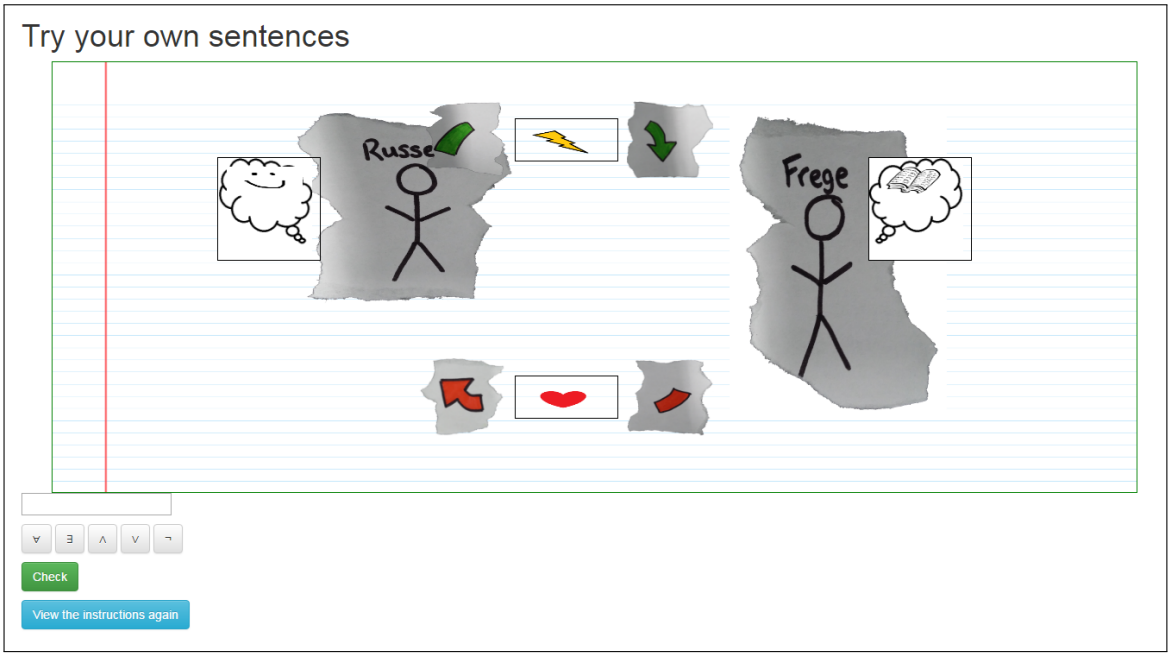


Figure 4.4: Third level prototype

At this point, it became apparent that a correct implementation of this final kind of level would involve work beyond the scope of the project. In particular, allowing users to input first order logic sentences without restriction would require a parser to convert 'natural' first order logic into the representation required by the program. Because of this, these kinds of levels were removed from the game design.

4.6 Game Design - Sequence Diagram

With the prototype levels completed, the next step was to decide which features of the prototype to keep and to design the final game.

It was decided to keep the structure of the prototype game based on the sub-system diagram given in section 4.5. A sequence diagram for a single level is given below.

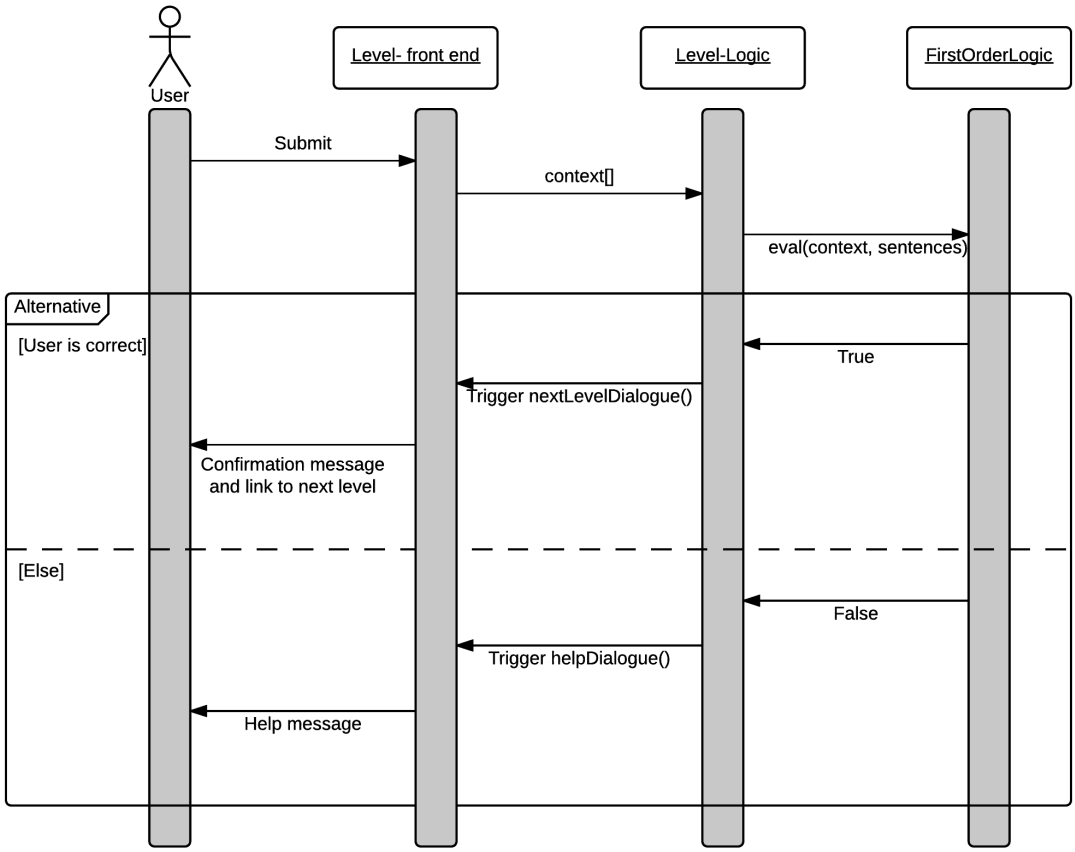


Figure 4.5: Sequence diagram

Each level will have hard-coded sentences, which will be presented to the user. The user must drag and drop icons into boxes such that the sentences would evaluate to true, and then click submit. Once this has been done, the positions of icons on the screen will be used to create a context – an array of true predicates. The hard-coded sentences will then be evaluated against the user-created context using a function eval(), which will be the same for all levels. Eval() will return true if the sentences have been made true and false if not. If the value true is returned, a dialogue confirming that the user is correct and containing a link to the next level will be presented to the user. If the value false is returned, a help message will be displayed instead.

4.7 FirstOrderLogic Design

The FirstOrderLogic code will contain an implementation of first order logic as a language. This will include the function eval(), which will be described in 4.7.3. Eval() must evaluate sentences against a context and return true of false.

The implementation of first order logic therefore needs the following:

1. The program needs its own internal representation of first order logic sentences. Each sentence needs to be stored in a form that can be evaluated against the context.
2. A representation of the context. The context will need to be computed from the scene when the user clicks submit.
3. A function which can evaluate the representation of sentences against a context.

4.7.1 Representation of Sentences

First order logic sentences can be represented as tree structures.

Figure 4.6: Example first order logic trees

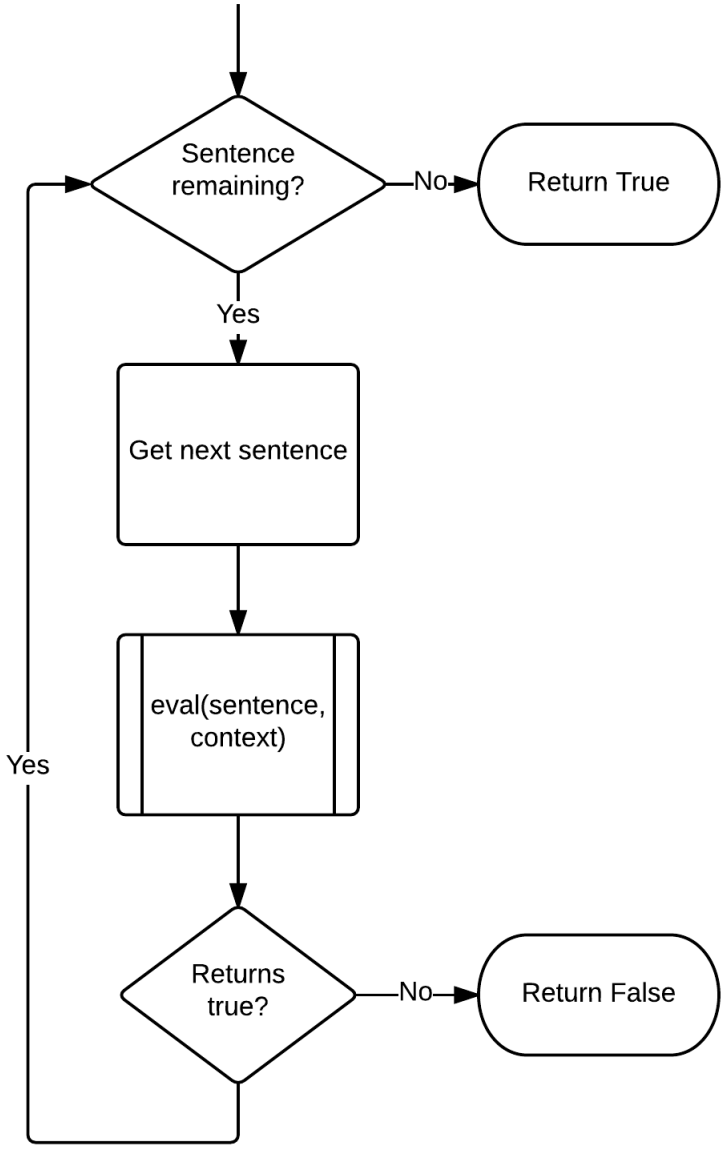
4.7.2 Representation of Context

Sentences are evaluated against a context, which is a list of atomic sentences that are currently true. The context will be an array of these sentences.

4.7.3 Evaluate Sentences Function

There must be a function which takes all of the sentences and evaluates each one against the context. This is shown in the following flow chart:

Figure 4.7: Evaluate all sentences flowchart



The eval() function is recursive and overloaded. Its function is described via the following table and pseudo code:

Table 4.1: Eval() function

Sentence Given	Return
A where A=$P(t_1, ..., t_n)$ (a predicate or a relation)	True is A is in context, false otherwise.
$A \vee B$	eval(context, A) OR eval(context, B)
$A \wedge B$	eval(context, A) AND eval(context, B)
$\forall x.A$	for all variables $v_1, ..., v_n$, eval(context, $A(x = v_1)$ AND ... AND eval(context, $A(x = v_n)$
$\exists x.A$	for all variables $v_1, ..., v_n$, eval(context, $A(x = v_1)$ OR ... OR eval(context, $A(x = v_n)$

Listing 4.1: Eval() pseudo-code

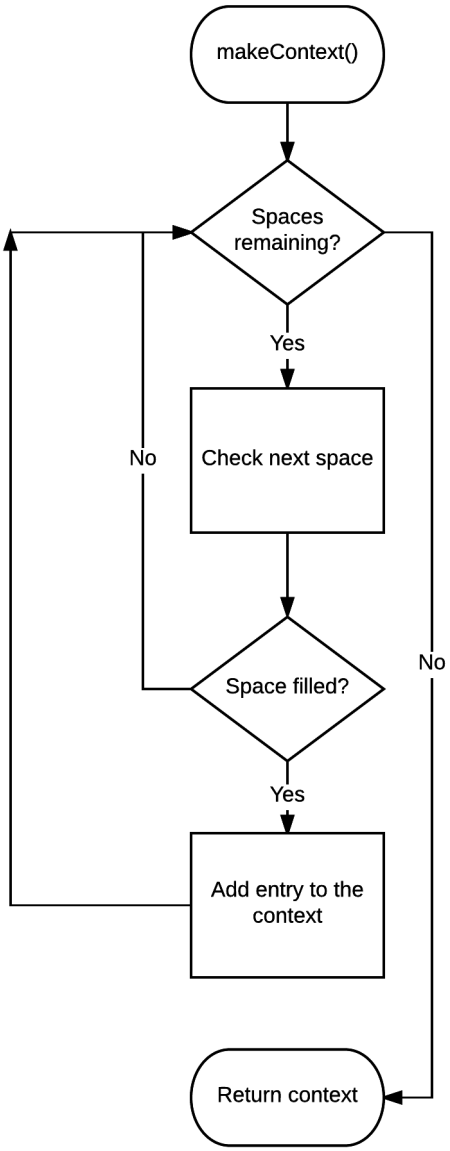
```
eval(context, A ∧ B) = eval(context, A) && eval(context, B)
eval(context, A ∨ B) = eval(context, A) || eval(context, B)
eval(context, ∀x=A = eval(context++[x = v1],A) && ... &&
    eval(context++[x = vn],A)
eval(context, ∃x=A = eval(context++[x = v1],A) || ... ||
    eval(context++[x = vn],A)
eval(context, P(t1,...,tn) = lookup(P(find(context, t1),...,
    find(context, tn))
```

4.8 Level-Logic Design

The back-end code for each level must take the position of elements on the page that the user has manipulated and use these to build a context. This is done via a method, makeContext().

It must check every space where a user is permitted to drag an element, and if there is an element present, add an appropriate entry to the context. This is shown diagramatically in the flowchart given below:

Figure 4.8: MakeContext() flowchart



4.9 Level Design

A key part of designing an educational game is level design. Including a level progression is one way of motivating users to keep playing a serious game (Dörner et al., 2016, p.19).

Increasing the difficulty of tasks at the right speed is an important part of designing a serious game. The difficulty of the game must increase in line with the increase in the user's skill level (Dörner et al., 2016, p.11). Too slow an increase causes the game to become boring and ineffective, too rapid an increase causes the game to become difficult and demotivating.

One of the challenges of story design is to progress the story and progress the difficulty of the levels alongside each other. A simple sentence needed to tell the story may need to be made challenging. Likewise, a more complicated aspect of the story may be difficult to describe early in the game if necessary notation has not yet been introduced. This tension is reflected in the level design for the game.

Level designs were drawn up as a table which detailed the sentences that a user would need to understand in order to complete the level and the information that they would be given before and after. A second table tracks the pedagogical reasoning behind that level. The full level design can be found in the appendix, but a typical example is given below:

Table 4.2: Design for level 7

Level	Before Story	After Story	Sentences
7	We can join more than one sentence together using the AND symbol, which looks like $\wedge$	Logic also has an OR symbol	Likes(Frege, Aristotle) $\wedge$ Logician(Frege)

Table 4.3: Rational design for level 7

Level	Sentences	To teach	Rationale
7	Likes(Frege, Aristotle) $\wedge$ Logician(Frege)	$\wedge$	Introduces $\wedge$ for the first time by joining two familiar sentences.

The level designs went through several modifications in order to refine the balance between telling the story and teaching the material effectively. Early level designs had a story with a much larger scope; they covered more of history and involved more logicians interacting with each other. This caused problems in level design because the story was often too complication to explain well whilst limited to the notation that had already been introduced. Additionally, introducing new notation was sometimes challenging because there was little flexibility in the story. The final level design is the result of compromise between the competing need to tell a story and the need to introduce new notation in a steady

order. It instead focuses mostly on a single logician writing a theory, and includes more mundane actions such as finding a pen. This version is much less constrained by actual history. Hence, some of the background on logic that was initially planned is omitted in order to make the story more flexible.

Chapter 5

Implementation

5.1 Language

Requirement 2.1 states that the game must not require complex installation or set-up and requirement 2.2 states that the game should be accessible and not have restrictive hardware or software requirements. It was therefore decided to develop the game as a web-based application because this ensures that the requirements 2.1 and 2.2 can be met. The alternative option, to create an executable file that could be installed, would only increase installation time.

The game does not require any data storage so can be written as an entirely client side application. With this in mind, JavaScript and HTML were chosen as the primary languages.

The JavaScript used in the project conforms to the ECMAScript 5 standard. There is a more recent version of the standard which contains features that would have been useful, but it is not supported by Internet Explorer and older versions of Microsoft Edge. Reports on browser market share vary, but the percentage of users still using the now outdated Internet Explorer has been reported as being as high as 19.91% between December, 2016 to February, 2017 (NetApplications.com, 2017). Given that accessibility is an important requirement for the game, working well on Internet Explorer is therefore important, so this compromise must be made.

5.2 Environment

The WebStorms IDE was used for most of the development. It provides syntax checking and code completion for HTML, JavaScript and CSS. It also allows quick testing of the

website in different browsers and debugging for JavaScript.

5.3 Implementation Process

The subsystem diagram given in the Design chapter is repeated below:

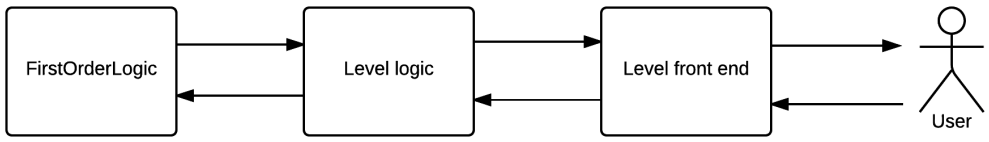


Figure 5.1: Original sub-system diagram.

The first part of the system to be implemented is the First Order Logic JavaScript page.

As specified in the sequence diagram 4.6, the FirstOrderLogic code will be interacted with by the level logic by a single function, eval(). Eval takes a set of sentences and a context (an array of true predicates for a scene) and returns true if the sentences are true and false otherwise.

The modular nature of the design means that FirstOrderLogic can be implemented and tested without any of the other components. A test page is set up testing the eval() function with a range of possible sentences.

Once all of the tests returned the correct answer, the levels were implemented chronologically, allowing the transition between levels to be tested as it will appear to the user.

After the implementation of the levels begun, the overall sub-system diagram was modified to reduce duplication of code, with code required by all levels being moved into its own JavaScript file, of which there is only one copy. The updated diagram is given below.

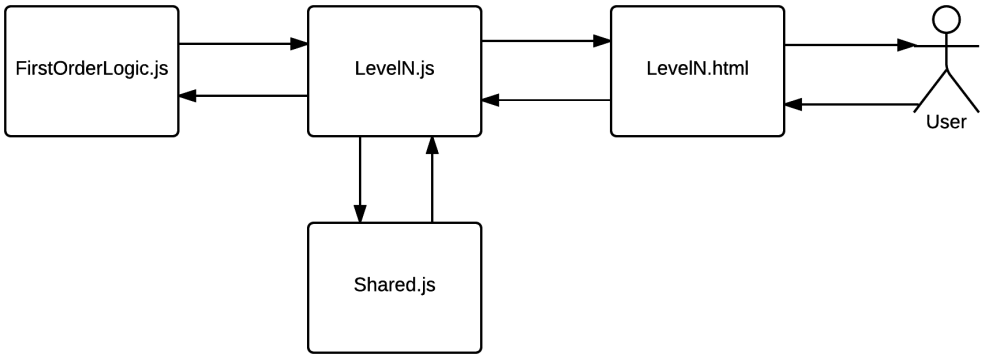


Figure 5.2: Updated sub-system diagram.

Additionally, the project makes use of the following open source libraries:

- jQuery - provides JavaScript utilities (jQuery Foundation, 2017).
- Bootstrap - provides CSS and JavaScript for a range of user interface elements, such as tooltips and pop-up boxes, as well as css for the page layout (Bootstrap Contributors, 2017).

A diagram showing the boundary between the system and the external libraries is given below.

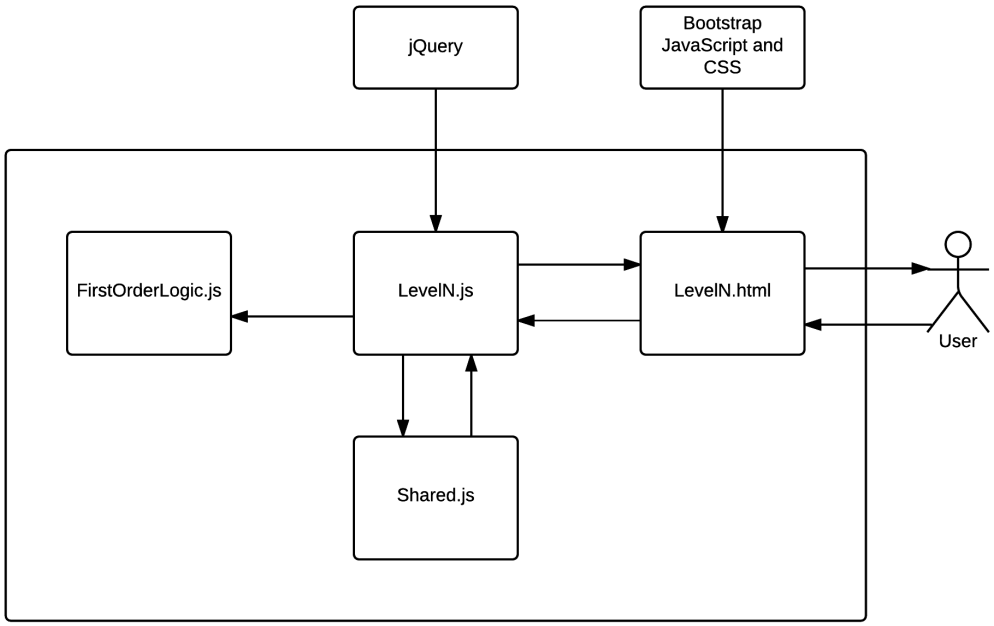


Figure 5.3: System boundary diagram.

The game also makes use of some open source clip art, obtained from https://openclipart.org.

5.4 Final Level Implementation

An example of a typical level implementation is given below:

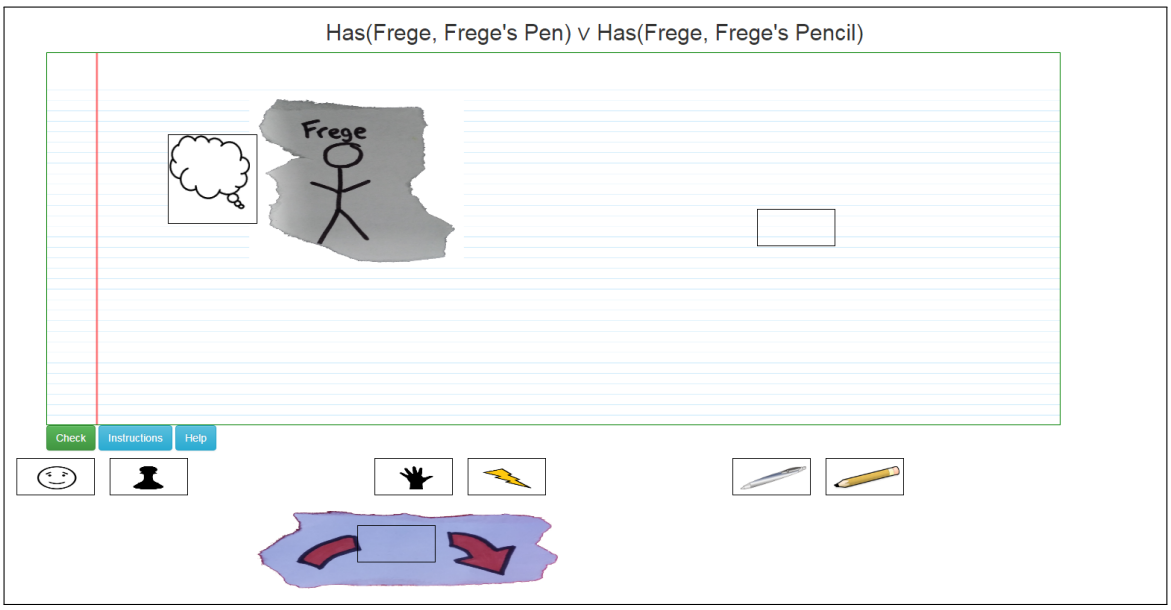


Figure 5.4: A typical level

The final level design includes drag-and-drop arrows; a space for objects to be dragged into on the right-hand-side; and a separation of draggable boxes into predicates, relations and objects.

Chapter 6

Testing

6.1 Introduction

There were three main phases for testing the game:

- Unit Testing
- Initial Game Testing
- Questionnaire

Unit Testing Unit testing was used to test the first order logic JavaScript code, before the actual levels were developed. This was discussed in the Implementation chapter of this report, section 5.3.

Initial Game Testing The Initial game testing took place immediately after all of the levels had been implemented. Its aim was to identify problems with the user interface or with the difficulty of the game. This was the first testing conducted by potential users of the game. This is described in the next section of this chapter.

Questionnaire The questionnaire was the last testing technique employed. The questionnaire aimed to assess whether the game was enjoyable, motivating and successful as a serious game, in short, whether it met some of the most important requirements laid out for the game. The questionnaire was completed by a group of adults and by a group of A-level students.

6.2 Initial Testing

The first test employed a think-aloud protocol (Lewis and Rieman, 1994) with three users. Think-aloud testing involves asking users to use a system whilst verbalizing their thoughts about what they are doing. It is used to check whether users naturally interact with the user interface as the developers expect.

The test involved three users who used the game under supervision. The users were asked to play the game and discuss their thoughts about what they were doing.

Tests using the think-aloud protocol often use video or audio recordings to gather data, rather than simple note-taking (Lewis and Rieman, 1994, p85), as this means that all data is captured. In this case, paper notes made by the observer were used instead to avoid causing unnecessary unease to the users.

The test allowed for observation of points during the game which caused users to stop. Notes were made at levels where there seemed to be confusion, or where users took longer than normal. Users were only given help in situations where they would not otherwise have been able to continue, as described by Lewis and Rieman (1994). Notable observations are discussed below, and the full results are given in Appendix D.

6.2.1 First User

Notable observations from the first test user were:

- First level - they did not read the instructions the first time, and needed to click the button to re-load them.
- Level 3 (the first level where arrows are used to show relations) - they needed help to understand the user interface at this level.
- Level 5 - they read through the instructions three times at this level before coming to an understanding.
- The levels where quantifiers and variables are used took much more time than the other levels.

Following testing with this user, the following changes were made:

- Some instruction modals were made harder to dismiss. This should make it less likely that users will dismiss them without reading them.

- A help button was added to each level, containing instructions on how to use the user interface.
- The instructions given on level 5 were changed, and an image was introduced to illustrate the concept.
- The level design was modified to introduce the idea of variables prior to the introduction of quantifiers, with the intention of reducing the learning curve for levels using both quantifiers and variables.

6.2.2 Second User

The modified game was then tested with a second user. Notable observations were:

- The user used the help button three times, so this was a helpful addition.
- They completed level 3 unaided, but still took longer than other levels.
- There were some problems with the tooltips not vanishing, and obstructing elements of the interface.

Following this testing, the following additional changes were made:

- The amount of drag and drop the user is required to do for level 3 was reduced, simplifying the relation interface for the first time it is used.
- The tooltip bug was investigated and found to occur when the game was run using certain browsers. The bug was then fixed for the browsers in question.

6.2.3 Third User

The third user completed the game with smoothly. Some levels took longer than others, but there were no difficulties with the user interface.

6.3 Adult Questionnaire

The next test involved asking users to play the game for twenty minutes, and complete a short questionnaire before and after the game. The pre-game questionnaire aimed to establish the users' past experience with learning maths and logic and their attitude to this sort of activity. The post-game questionnaire asked the users about how they found the game and how much they learned from it. The questionnaire was completed by ten adult users.

The results from the post-game questionnaire can be used to:

- Establish whether requirement 3.5 (the game should be enjoyable) has been met, by looking at user responses to relevant questions.
- Establish whether requirements 2.1-2.5, which describe what the user should learn from playing the game, have been met. This can be inferred from which level the users reached; for example, if a user completed levels involving relations, then they have sufficient understanding of that topic.
- Judge whether requirement 3.3 (the game should not take a long time to learn how to play) has been met. Evidence to support this can be gathered from the number of users completing all of the levels - if the game took too long to learn, then users would not complete all levels in 20 minutes.

Comparing users' responses to the pre-game questionnaire with their responses to the post-game questionnaire allows us to see which kinds of user were most successful at playing the game. This serves the following purposes:

- To see whether the users combined answers to questions before the game could predict performance in the game.
- To examine whether requirement 3.6 (the game should be accessible to users with a range of mathematical backgrounds) has been met by comparing the performance of people with different levels of mathematics qualification and experience.
- To establish if the game is pitched at the right level. It is likely that people who have heard of first order logic before will find the game easier than those who have not. Ideally people who have not heard of first order logic before should have success playing the game.
- To see which factors predict success in the game the most. Ideally the game should be accessible to a wide range of users, and responses to the pre-game questionnaire should have as minimal impact on the results of the post-game questionnaire as possible.

6.3.1 Post-game Questionnaire Results

Most of the questions on the post-game questionnaire asked users to rate how strongly they agreed or disagreed with some statements on a 1-5 scale. These results are summarised below:

Question	Mean	Median
The game was easy to play	3.8	4
I always knew what I had to do to move on to the next level	3.5	3.5
I always knew what the symbols in the sentences meant	4.4	5
First order logic is easy	3.8	4
The game was enjoyable	4.0	4
The game was interesting	4.4	4.5
Logic puzzles are difficult	3.2	3
Maths is difficult	3.8	4

Table 6.1: Summary of results from the post-game questionnaire

The mean answer for the game being enjoyable was 4 and the mean for the game being interesting was 4.4. This indicates that the requirement for the game to be enjoyable was met.

In addition to these questions, users were asked to state which level they reached within 20 minutes, and if any levels were too difficult. Eight of the ten users completed the game, with seven reporting that no level was too difficult. All users got to at least level 18, indicating that they had understood sentences involving predicates, relations, logical connectives and a simple sentence using the existential quantifier. This means that requirements 2.1-2.5 have been met. Further evidence to support this is given by users' answers to the question 'I always understood what the symbols meant', with the mean response being 4.4 out of 5.

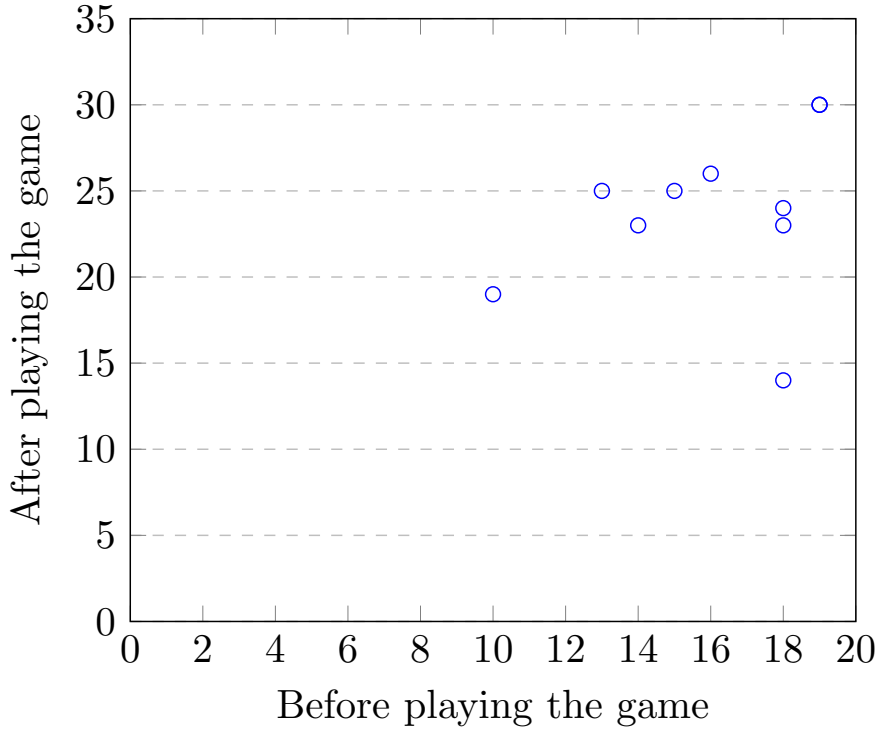
Levels 18 and 19 were the levels that users reported as being too difficult. These are levels that use quantifiers, variables and a logical connective in a single sentence. It is possible that users could benefit from more practice with variables and quantifiers before introducing them into more complicated sentences.

6.3.2 Comparison of Pre-Game and Post-Game Results

One way of measuring whether user responses before and after the game correlate is to sum their responses to each question before and after and compare the two measurements. The pre-game score represents a measure of how likely a user was to enjoy the game and learn from it based purely on their prior experience. The post-game score represents how successful the game was based on whether the user found the game easy to play and whether they enjoyed it. The final two questions on the post-game questionnaire were omitted because they are repetitions of questions in the pre-game questionnaire, and this could artificially increase the correlation between the two measures.

The validity of combining the responses in this way can be measured using Cronbach's Alpha, which measures the internal consistency of a group of variables. The Alpha for the pre-game responses was 0.79, and the Alpha for the post-game responses was 0.86. These measurements are high enough that comparing the two sums is valid.

Figure 6.1: Comparison of summed responses before and after playing the game



The graph shows a slight positive correlation, with a Pearson correlation coefficient of 0.328. This demonstrates that a user's prior attitude towards learning maths and logic had a small impact on the success of the game.

6.3.3 Level of Qualification

The next thing to be examined from the test results is how prior mathematics experience affected responses to the game. The game is aimed at A-Level students, and thus should have been successful with users whose highest level of qualification was A-Level or equivalent.

Of the sample, five were studying mathematics as part of their degrees and four had an A-Level or equivalent in mathematics. Only one user gave GCSE or equivalent as their highest level of mathematics qualification, so this was left out of the comparison.

Question	A-Level in maths	Undergraduate study in maths
The game was easy	3.25	4.40
I always knew what to do to move on to the next level	3.25	3.80
I understood what all of the symbols meant	3.75	4.80
First order logic is easy	2.75	4.60
The game was enjoyable	4.00	4.40
The game was interesting	4.50	4.40

Table 6.2: Mean responses by highest mathematics qualification

Unsurprisingly, people holding or studying for degrees involving mathematics had higher average responses for all measures relating to how easy they found the game. Despite this, there was no evidence to suggest a link between level of mathematics qualification and which level the user reached - 4 of the 5 users with maths as part of their degree and 3 of the 4 users with an A-level completed the game.

The difference between the two groups with regards to how enjoyable and interesting the game was is very small. This is pleasing, as it indicates that the game has a broad appeal across both groups.

6.3.4 Factors that predict game success

The next thing to check was whether any individual score from the pre-game questionnaire predicated any of the results from the post-game questionnaire. This will determine whether the game appealed more to certain types of user. To do this, the Pearson correlation coefficient between each response to the pre-game survey and each response to the post-game survey was calculated. These are given in the table below.

	I am good at maths	I like logic puzzles	I liked maths at school	I am good at learning theoretical concepts	Logic puzzles are difficult	Maths is difficult
The game was easy	0.421	0.267	0.225	-0.088	0.183	0.115
I knew what I had to do to move on to the next level	0.848	0.484	0.626	0.159	-0.372	-0.311
I knew what the symbols meant	0.312	-0.046	0.066	-0.251	0.195	0.327
First order logic is easy to understand	0.395	-0.068	-0.145	-0.267	0.287	0.380
The game was enjoyable	0.685	0.593	0.590	0.429	-0.111	-0.419
The game was interesting	0.431	-0.064	-0.023	0.039	0.120	-0.113

Table 6.3: Pearson correlation coefficients between pre-game and post-game mean answers

Most of the prior responses did not correlate strongly with the responses after the game. This could be because the sample size is not large enough. However, it is possible that this indicates that the game had a broad enough appeal that, for example, whether or not the user likes maths did not matter.

Of the results that did have stronger correlations, most are not surprising. The five strongest correlations were:

Pre-Game Answer	**Post-Game Answer**	**Correlation Coefficient**	**Remarks**
Good at maths	Knew how to get to the next level	0.848	This is unsurprising.
Good at maths	Found the game enjoyable	0.685	It is impossible to say whether this is a direct correlation, or whether both relate to a third variable.
Liked maths at school	Knew how to get to the next level	0.626	It is impossible to say whether this is a direct correlation, or whether both relate to a third variable.
Likes logic puzzles	Found the game enjoyable	0.593	It is not surprising that people who like logic puzzles enjoyed the game more.
Liked maths at school	Found the game enjoyable	0.590	Again, people who enjoy maths are more likely to enjoy the game.

Table 6.4: Discussion of the five strongest correlations

6.3.5 Summary of Results from Adult Questionnaire

- Most users found the game enjoyable and interesting. Requirement 3.5 has been met.
- Most users finished the game, and all users completed levels involving predicates, relations, objects and logical connectives. Most users agreed that they understood what all of the first order logic symbols meant. Requirements 2.1-2.3 have been met.
- Users who did not finish or reported that some levels were too hard found the levels with quantifiers and logical connectives to be the most difficult.
- There was a slight positive correlation between overall response pre-game and overall response post-game.
- Whilst people with higher level mathematics qualifications found the game and first order logic easier, they were not necessarily more likely to finish the game. Level of qualification also had little impact on whether a user found the game enjoyable and interesting.

- Users who considered themselves good at maths, enjoyed maths at school and like logic puzzles were more likely to enjoy the game.

6.4 School Questionnaire

Following the questionnaire with adults, a similar test was completed with A-level students, to see whether the game was as successful with the target audience. An A-Level Computing class consisting of five students were asked to play the game and complete the questionnaire before and after. The students were all studying for an A-level in Computing, four were also studying Mathematics and none had heard of First Order Logic before.

The test had the following purposes:

- To establish whether the A-level students found the game enjoyable and interesting, and how this compared to the adult responses.
- To investigate whether the game was pitched at the right level of difficulty for this user group specifically. This can be deduced by looking which levels the users successfully completed, whether they found any individual levels too difficult and the extent to which they agreed that the game was easy and that first order logic was easy.

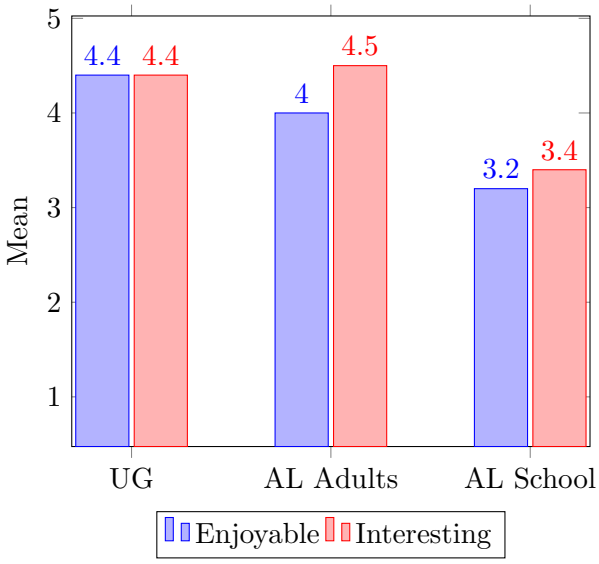
6.4.1 Was the Game Interesting and Enjoyable?

The mean and median responses to the questions 'the game was interesting' and 'the game was enjoyable' are given in the following table.

Question	Mean	Median
The game was enjoyable	3.2	3
The game was interesting	3.4	3

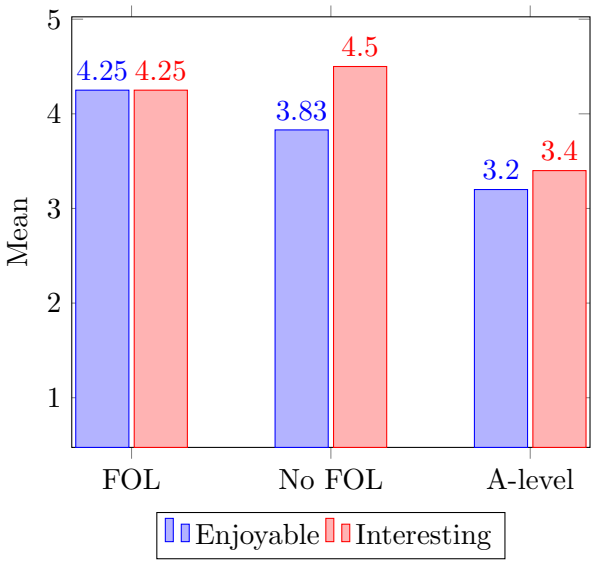
Table 6.5: Mean and median responses to whether the game was enjoyable and interesting

The results for the game being interesting and enjoyable were less encouraging with the A-level students, with averages of 3.2 and 3.4 out of 5 respectively. Interestingly, these results are also lower than the average for this question amongst adults with the same level of education, and lower than the responses from adults who had not heard of first order logic before. This is shown in the following bar charts (note that none of the A-level students had heard of first order logic before):



UG - adults currently studying for or holding a degree that uses maths.
AL Adults - adults whose highest maths qualification is an A-level or equivalent.
AL School - A-level student respondents.

Figure 6.2: Enjoyable and interesting mean responses by highest maths qualification



FOL - Adult respondents who had heard of first order logic before.
No FOL - Adult respondents who had not heard of first order logic.
A-Level - A-level student respondents (none of whom had heard of first order logic before).

Figure 6.3: Enjoyable and interesting mean responses by prior knowledge of first order logic

It is not clear why adults and current A-level students differ. Mildner and Müller (2016, p.62) state that whether or not a game appeals to a group of users is likely to depend on demographic factors, so it is possible that this design was more appealing to older users than to A-level students.

6.4.2 Difficulty of the game

How difficult users found the game can be measured using the following results to the post-game questionnaire:

1. The extent to which they agreed that the game was easy to play.
2. Whether they always knew what to do to move on to the next level.
3. The extent to which they agreed that first order logic was easy.
4. The level that the user reached.
5. Whether any levels were too difficult.

Questions 1-3 in this list were measured by statements scored from 1-5, as in the adult questionnaire. The results are summarised below:

	Mean	Median
The game was easy to play	3.8	4
I always knew what I had to do to move on to the next level	3.2	3
First order logic is easy	3.4	3

Figure 6.4: Mean and median responses to questions relating to difficulty

The mean responses here were less than the means for adults, but not when compared to adults who had not studied/were not studying mathematics or to adults who had not heard of first order logic. This is shown in the bar charts below:

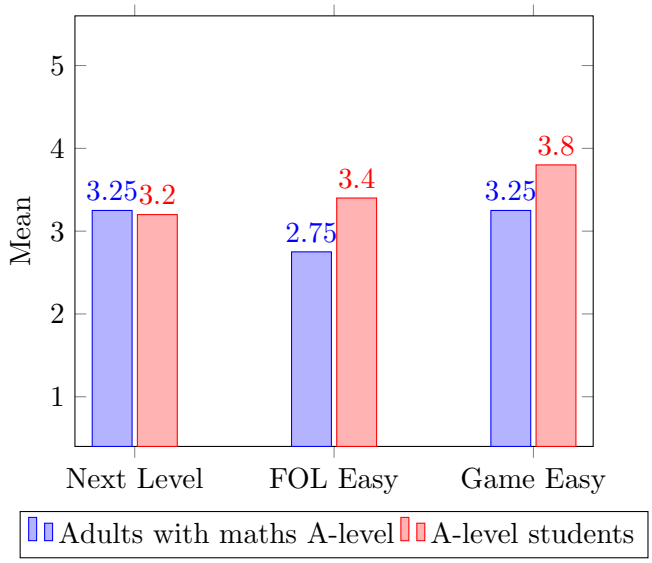


Figure 6.5: Difficulty measures from adults with A-level as the highest maths qualification and from the A-level students

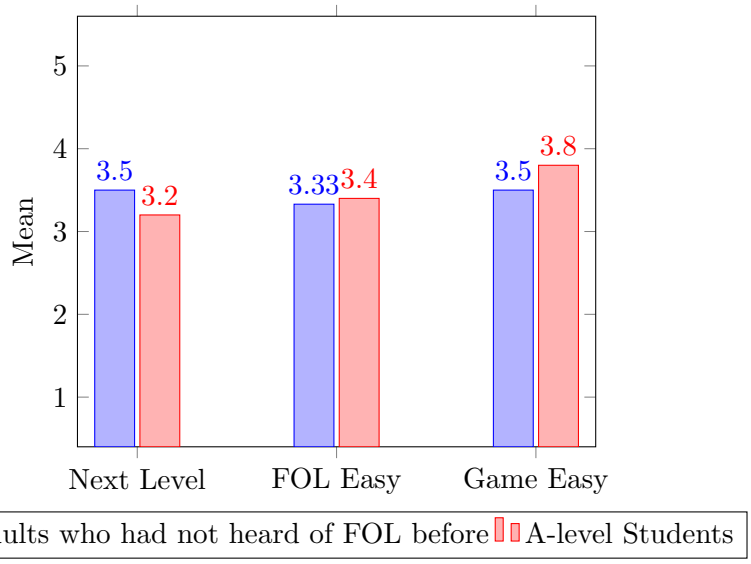


Figure 6.6: Difficulty measures from adults who had no prior knowledge of first order logic and from the A-level students

This indicates that there is little difference between how difficult the game was for the A-level students and for adults with similar levels of mathematical experience.

In addition to these findings, four of the five users completed the game, with one only reaching level 19. Unsurprisingly, the one person who did not complete the game was also the only student not studying A-level Mathematics. None of the students believed any level was too difficult. This supports the finding from the adult questionnaire that requirements 2.1-2.5, which describe what users should learn from the game, have been met. Most of the A-level students believed the game was easy to play, with an average rating of 3.8 and a median rating of 4.

6.4.3 Summary of Findings from the School Questionnaire

A summary of findings from the testing with the class of A-level students is given below:

- A-level students found the game less interesting and enjoyable than adult users. This holds even when the students are compared to adults with the same level of mathematics education and who have never heard of first order logic before. It is unclear why this is.
- A-level students found the game roughly as difficult as adults with similar mathematical backgrounds.
- Four of the five students completed the game, and all reached at least level 19. This means that all students completed levels using predicates, objects, relations, logical connectives and simple sentences using quantification, which means that the game meets its fundamental aim of teaching A-level students the basics of first order logic.
- Most of the students agreed that the game was easy to play, even if they did not always know what to do to move on to the next level.

6.5 Summary of Testing

There were four different phases of testing for the game. Unit testing was carried out during the implementation of the game, to test whether the first order logic functions worked as expected. Initial testing of the game used a think-aloud technique to uncover usability problems in the design. Finally questionnaires were used with two groups of users, adults and a class of A-level Computing students, to establish whether certain requirements were met. This testing found that the game succeeded in teaching comprehension of first order logic sentences to both groups, and thus meets its primary aim.

Chapter 7

Conclusions

7.1 Reflections

7.1.1 Achievements

This project contributes a serious game for teaching first order logic to school students. There are existing serious games that teach this topic, such as Tarski's World (Barwise et al., 2000), but none that are aimed at an audience other than University students studying mathematics or computer science courses. The game also differs from most serious games in logic in that it is entirely web-based, and therefore is accessible without complex installation and is platform independent.

The results from the questionnaire showed that A-level students were able to complete the game. This is an important finding, because it demonstrates that this kind of logic is not prohibitively difficult for school-aged students, and can be taught to people without strong mathematical backgrounds if gamification is used.

It was also found that adults who completed the questionnaires both enjoyed the game and learned from it. This demonstrates that gamification and enjoyment do not detract from learning.

7.1.2 Limitations and Future Work

The following section details the limitations of the project and, based on these, identifies ideas for future work.

Type of user input

The requirements specification for this project initially specified that there should be two kinds of level: one where users were expected to understand sentences, and another where users were expected to write their own. This would have been advantageous, as it would have allowed drill style activities (recognising sentences) and creativity (allowing users to experiment with new sentences to see if they are true or false). Both of these styles of game have been shown to be successful (Charsky, 2010; Dostalova and Lang, 2011). Despite this, the second kind of level was removed from the design in order to keep the system within the scope of the project, as discussed in the Implementation chapter.

The possibility of including levels where users were expected to enter their own sentences could be investigated. Whilst it is possible that this feature would increase the quality of learning from the game, it is possible that the additional constraints this would put on the user interface would actually cause this feature to detract from user experience. Further evaluation would be needed to determine the effect that this change would have.

Aesthetics

The design for the game used a 'scrapbook' theme, designed to be consistent with the drag-and-drop interface. However, the design section paid very little attention to the appearance of the game. Aesthetics are an important way of motivating users to play a game (Mildner and Müller, 2016, p.61). It would have been good to explore the appearance of other similar games during the technology survey, as well as their functionality.

Requirements Elicitation

The requirements elicitation process for this project focussed mainly an existing games, existing methods for teaching first order logic and research into serious games and gamification.

Requirements gathering could have involved prospective users, both students and educators. One problem identified in testing was that A-level students did not find the game as enjoyable or as interesting as adults did. Using participatory design techniques (Preece et al., 2015) to involve A-level students in designing the game might result in a design more appealing to these users.

Personalisation

Dörner et al. (2016, p.10) describes how more in-depth personalization can make serious games more effective. Currently the game adapts to users in two basic ways: users progress

to the next level only having completed the previous one; and a help message is supplied if users submit an incorrect answer.

Future work could look at whether building in real-time assessments of how the user is progressing could be used to adapt the game. For example, by increasing or decreasing difficulty or providing hints that relate specifically to the user's current actions. These strategies were employed successfully by Dostalova and Lang (2011) and Mildner et al. (2015). It is apparent from the questionnaire results that some users found the level 18 and 19 too difficult, but others believed that the game was easy to play - it would be good advantageous if the game was able to adapt to these different users.

Development After Testing

Whilst the game underwent user testing, there was little time after testing to modify the game based on user feedback. Serious games in particular are often modified after testing based on whether users found the game to be too difficult or too easy (Dörner et al., 2016, p.19).

Looking at the questionnaire results, it is likely that users need more practice with quantifiers and variables before introducing more complex sentences.

Evaluation Methods

The user testing conducted as part of this project used questionnaires to assess user's attitudes towards learning mathematics and logic before and after playing the game. However, the game was not compared to other methods for teaching first order logic.

With more time, more effective testing could be carried out. A first order logic course with the same content as the game could be designed without gamification techniques, for example, using a textbook. The test users could then be split into two groups, one given the serious game and the other given the non-gamification course. The results of the questionnaire for each of the group could be compared to establish whether the serious game was as effective as traditional teaching methods.

Schunn and Patchan (2009) researched an online logic course by comparing the exam results of students who used it with students who were taught using traditional methods. Similarly, users of this serious game and users taught first order logic in a different way could also be given a test containing first order logic questions, to test which group learned more.

7.2 Summary

This dissertation has described the development of a serious game to teach first order logic. The game was shown in testing to be successful at teaching first order logic to adults with different mathematical backgrounds and to A-Level Computing students. The findings demonstrate that first order logic, which is normally only taught at degree level, can be taught to secondary school students when gamification is employed. This shows that there is a place for serious games in A-level Computer Science and Mathematics education.

Bibliography

Armoni, M. (2011), 'Looking at secondary teacher preparation through the lens of computer science', *ACM Transactions on Computing Education (TOCE)* **11**(4), 23.

Barwise, J. and Etchemendy, J. (1993), *The language of First-Order logic, including the Macintosh program Tarski's world 4.0: Including the Macintosh Programme, Tarski's world 4.0*, 3 edn, Center for the Study of Language and Information, United States.

Barwise, J., Etchemendy, J., Allwein, G., Barker-Plummer, D. and Liu, A. (2000), *Language, proof, and logic*, Seven Bridges Press,U.S., United States.

Bootstrap Contributors (2017), 'Bootstrap', `http://getbootstrap.com`. Accessed: 10-04-2017.

Brown, N. C. C., Sentance, S., Crick, T. and Humphreys, S. (2014), 'Restart: The resurgence of computer science in uk schools', *ACM Transactions on Computing Education* **14**(2), 1–22.

Charsky, D. (2010), 'From edutainment to serious games: A change in the use of game characteristics', *Games and Culture* **5**(2), 177–198.
URL: *http://gac.sagepub.com/content/5/2/177*

Computing at School Working Group (2012), Computer science: A curriculum for schools, Technical report.
URL: *https://www.computingatschool.org.uk/data/uploads/ComputingCurric.pdf*

de Aguilera, M. and Mendiz, A. (2003), 'Video games and education', *Computers in Entertainment (CIE)* **1**(1), 1.
URL: *http://dl.acm.org/citation.cfm?doid=950566.950583*

Department for Education (2013), National curriculum in england: computing programmes of study, Technical report, London, UK.
URL: *https://www.gov.uk/government/publications/national-curriculum-in-england-computing-programmes-of-study/national-curriculum-in-england-computing-programmes-of-study*

Deterding, S., Dixon, D., Khaled, R. and Nacke, L. (2011), From game design elements to gamefulness: Defining "gamification", *in* 'Proceedings of the 15th International Academic MindTrek Conference: Envisioning Future Media Environments', MindTrek '11, ACM, New York, NY, USA, pp. 9–15.
URL: *http://doi.acm.org/10.1145/2181037.2181040*

Dörner, R., Göbel, S., Effelsberg, W. and Wiemeyer, J. (2016), 'Introduction', *Serious Games* pp. 1–34.

Dostalova, L. and Lang, J. (2011), 'Organon: Learning management system for basic logic courses', *Lecture Notes in Computer Science* pp. 46–53.

European and Culture DG. (2008), The european qualifications framework for lifelong learning, Technical report, Luxembourg.
URL: *https://ec.europa.eu/ploteus/sites/eac-eqf/files/leaflet_en.pdf*

Eysink, T., Dijkstra, S. and Kuper, J. (2001), 'Cognitive processes in solving variants of computer-based problems used in logic teaching', *Computers in Human Behavior* **17**(1), 1–19.
URL: *http://www.sciencedirect.com/science/article/pii/S0747563200000388*

Fung, P., O'Shea, T., Goldson, D., Reeves, S. and Bornat, R. (1994), 'Why computer science students find formal reasoning frightening', *Journal of Computer Assisted Learning* **10**(4), 240–250.

Gee, J. P. (2003), 'What video games have to teach us about learning and literacy', *Computers in Entertainment* **1**(1), 20.

Glover, I. (2013), 'Play as you learn: Gamification as a technique for motivating learners', *World Conference on Educational Multimedia, Hypermedia and Telecommunications* **2013**(1), 1999–2008.
URL: *http://shura.shu.ac.uk/7172/1/Glover_-_Play_As_You_Learn_-_proceeding_-112246.pdf*

Goldson, D. and Reeves, S. (1994), 'Review: The language of first-order logic, including the macintosh program tarski's world', *The Philosophical Quarterly (1950-)* **44**(175), 272–275.
URL: *http://www.jstor.org/stable/2219757?sid=primo&origin=crossref&seq=2#page_-scan_tab_contents*

Goldson, D., Reeves, S. and Bornat, R. (1993), 'A review of several programs for the teaching of logic', *The Computer Journal* **36**(4), 373–386.
URL: *http://comjnl.oxfordjournals.org/content/36/4/373.full.pdf+html*

Hainey, T., Connolly, T., Baxter, G., Boyle, L. and Beeby, R. (2012), 'Assessment integration in games-based learning: A preliminary review of the...: Ebscohost', *Proceedings of the European Conference on Games Based Learning* **1**, 174–183.

Hamada, M. (2008), 'An integrated virtual environment for active and collaborative e-learning in theory of computation', *IEEE Transactions on Learning Technologies* **1**(2), 117–130.

Hamada, M. (2009), 'Pushdown automata simulator', *Learning by Playing. Game-based Education System Design and Development* pp. 328–338.

Hamari, J., Koivisto, J. and Sarsa, H. (2014), 'Does gamification work? – a literature review of empirical studies on gamification', pp. 3025–3034.
URL: *http://dl.acm.org/citation.cfm?id=2585491*

Huth, M. and Ryan, M. D. (2004*a*), 'Lics web tutor'. Accessed: 15-11-2016.
URL: *http://www.cs.bham.ac.uk/research/projects/lics/tutor/index.html*

Huth, M. and Ryan, M. D. (2004*b*), *Logic in computer science: Modelling and reasoning about systems*, 2 edn, Cambridge University Press, Cambridge, United Kingdom.

Isayama, D., Ishiyama, M., Relator, R. and Yamazaki, K. (2016), 'Computer science education for primary and lower secondary school students', *ACM Transactions on Computing Education (TOCE)* **17**(1), 2.

jQuery Foundation (2017), 'jquery', `https://jquery.com`. Accessed: 10-04-2017.

Kunkle, W. M. and Allen, R. B. (2016), 'The impact of different teaching approaches and languages on student learning of introductory programming concepts', *ACM Transactions on Computing Education (TOCE)* **16**(1), 3.

LEGO Group (2016), 'Lego mindstorms ev3', `https://www.lego.com/en-gb/mindstorms/products/mindstorms-ev3-31313`. Accessed: 03-11-2016.

Lewis, C. and Rieman, J. (1994), *Task-Centered User Interface Design: A Practical Introduction.* [online] Available from: `http://hcibib.org/tcuid/tcuid.pdf` (Accessed: 20-03-17).

Lifelong Kindergarten Group (2016), 'Scratch', `https://scratch.mit.edu/`. Accessed: 02-11-2016.

Malone, T. W. (1981), 'Toward a theory of intrinsically motivating instruction', *Cognitive Science* **5**(4), 333–369.
URL: *http://www.sciencedirect.com/science/article/pii/S0364021381800171*

Malone, T. W. and Lepper, M. R. (1987), 'Making learning fun: A taxonomy of intrinsic motivations for learning', *Aptitude, learning, and instruction* **3**(1987), 223–253.

Maloney, J., Resnick, M., Rusk, N., Silverman, B. and Eastmond, E. (2010), 'The scratch programming language and environment', *ACM Transactions on Computing Education (TOCE)* **10**(4), 16.

Mauch, E. (2001), 'Using technological innovation to improve the problem-solving skills of middle school students: Educators' experiences with the lego mindstorms robotic invention system', *The Clearing House: A Journal of Educational Strategies, Issues and Ideas* **74**(4), 211–213.

Melchior, A., Cutter, T. and Cohen, F. (2005), Evaluation of first lego league underserved initiative., Technical report, Executive summary, Brandeis University.

Mildner, P. and Müller, F. F. (2016), *Design of Serious Games*, Springer International Publishing, Cham, pp. 57–82.
URL: *http://dx.doi.org/10.1007/978-3-319-40612-1_3*

Mildner, P., Stamer, N. and Effelsberg, W. (2015), 'From game characteristics to effective learning games', *Lecture Notes in Computer Science* pp. 51–62.

NetApplications.com (2017), 'Browser market share', `https://www.netmarketshare.com/browser-market-share`. Accessed: 10-04-2017.

Open Learning Initiative (2015), 'Logic and proofs - oli', `http://oli.cmu.edu/courses/free-open/logic-proofs-course-details/`. Accessed: 18-11-2017.

Preece, J., Rogers, Y. and Sharp, H. (2015), *Interaction Design: Beyond Human-Computer Interaction*, 4 edn, John Wiley & Sons, Chichester.

Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., Millner, A., Rosenbaum, E., Silver, J., Silverman, B. and Kafai, Y. (2009), 'Scratch: Programming for all', *Communications of the ACM* **52**(11), 60–67.
URL: *http://dl.acm.org/citation.cfm?id=1592779*

Schunn, C. and Patchan, M. (2009), 'An evaluation of accelerated learning in the cmu open learning initiative course logic & proofs', *Learning Research and Development Center, University of Pittsburgh* .

The Openproof Project (2016), 'Openproof courseware: Tarski's world', `https://ggweb.gradegrinder.net/tarskisworld`. Accessed: 01-11-2017.

Appendix A

Level Designs

A.1 Early Level Design

Level	**Story**	**Sentences**	**Rationale**
1	Frege is a logician. We say this using Logician(Frege)	Logician(Frege)	Introduces the user interface and one of the characters. The instructions for this level have the most detailed instructions on how to use the UI. Uses the most basic of possible sentences. Omits some of the options - restricting options for beginner users.
2	Frege wants to talk about everything in the world, just by using logic, but there isn't enough logic to do this.	Sad(Frege)	User should use the UI to do a task of the same kind as in level 1, but without explicit instructions.
3	Frege has an idea.	HasIdea(Frege)	Further practice using the UI and completing a simple drill. A speech-bubble is used to show Frege's idea.

4	Frege owns a plant that is green. A detour from the main story is made to introduce the idea of objects with properties.	Green(Frege's plant)	Introduces a theoretical idea in the instructions (that all things are objects). User is then asked to complete a task of similar difficulty to the one before, to allow them to apply this understanding. The end of the level introduces the idea that people are objects too.
4-1	We can also use the not symbol to talk about things that are not true.	NOT Black(Frege's plant)	Introduces the negation symbol in the instructions. Keeps the same scene and task as previously given allows user to apply the negation symbol to a task.
5	We can use sentences to talk about more than two objects at the same time.	Likes(Frege, Frege's Plant)	Introduces the idea of relations – predicates that can take more than one argument.
6	Will take a detour from the plot to explain quantifiers.		
7	Likewise		
8	Frege wrote his story down	HasTheory(Frege)	For plot .
9	There exists x. Likes(x, FregeTheory)	Bertrand Russell liked Frege's theory a lot.	For further application of the existential quantifier. Also advances plot by introducing Bertrand Russell – having two characters means that more interesting sentences can be typed.
10	Likes(Russell, Frege) AND Likes(Frege, Russell)	Frege and Russell liked each other.	For further practice of predicates with two arguments. Introduces AND.
11	Sad(Russell) AND Sad(Frege)	Russell finds a mistake in Frege's work, which makes them both sad.	Further practice of AND.
14	HasIdea(Russell)	Russell has an idea	Storytelling

15	HasIdea(Russell) AND HasIdea(Whitehead)	Russell and his friend Whitehead come up with their own idea on how to make everything logic.	Practice AND. Introduce Whitehead.
16	HasTheory(Russell) AND HasTheory(Whitehead)	Russell and his friend Whitehead write a new theory.	Practice AND. Storytelling – introduce Principia Mathematica.

A.2 Final Level Design

Level	Before Story	After Story	Sentences
1	Gottlob Frege is a logician. We can say this by writing Logician(Frege). Frege is always thinking about logic - make the sentence Logician(Frege) true by dragging the logic symbol into Frege's thoughts.	Awesome!	Logician(Frege)
2	Frege wants to make everything logic.But it doesn't work! Make the sentence at the top of the screen true.	Like all famous logicians when they're sad, Frege hides in his room and refuses to let go of his teddy bear. This is a bit embarrassing, but at least he hasn't developed a drinking problem.	Sad(Frege)
3	Frege has a teddy bear named after one of the most famous logicians in history - Aristotle. We can talk about this using a relation		
3	A relation talks about how two objects relate to each other. Drag the arrow up to between Frege and Aristotle and add an emotion.	Aristotle is all that stands between Frege and a descent into despair, but this is hard to say in logic, so we'll settle with Likes(Frege, Aristotle).	Likes(Frege, Aristotle)

4	Frege needs to invent some more logic. But he has an idea!	Nice!	Has(Frege, Idea)
5	Frege's idea is to make everything in the world an object with properties. Frege's bed is an object with the property 'comfortable'. We write that as Comfortable(Frege's bed). Aristotle is a bear. We write that as Bear(Aristotle).		
5	This way, all Frege needs to do to make everything logic is to give it a name and say logic(object).	Well that seems easy enough.	Picture with logic(Frege) and logic(Aristotle)
6	Frege's teddy bear is an object with the name Aristotle and the property that it is brown.		
6	Objects are not always physical objects. Frege's idea is still an object, which is why we can write Has(Frege, Idea)	Correct(you)!	Has(Frege, Idea)
7	We can join more than one sentence together using the AND symbol, which looks like $\wedge$	Logic also has an OR symbol	Likes(Frege, Aristotle) AND Logician(Frege)
8	Frege,can either stay in bed OR write his idea down to form a theory.	He decides to do both. In logic, this is allowed, because OR just means that at least one of the sentences is true.	
9	Frege can write his theory with a pen or a pencil.	Great!	Has(Frege, Frege's pen) OR Has(Frege, Frege's pencil)

10	While Frege is working, his housekeeper shouts up: "Would you like tea or hot chocolate?". Frege replies "Yes" because he is a logician.	The housekeeper doesn't think Frege's joke is funny. Not anymore.	Has(Frege, Tea) OR Has(Frege, Hot Chocolate)
11	Frege needs to choose a cup to put his tea or hot chocolate in.		Alarmed(Frege), [picture of a stack of cups]
11	But wait! The cups don't have names yet. They haven't been made logic.		
12	Frege needs to name all the cups		[picture of cups all with names]
12	Frege can't possibly name everything in the world.	Poor Frege...	Despairs(Frege)
13	As is his habit whenever he is in despair, Frege heads upstairs to find Aristotle.		
13	But before he gets there, he is interrupted by the housekeeper. "Every day I make you tea or hot chocolate and every day you refuse to drink it because the cups are not logical enough!"	"I don't understand why you need to name all of the cups just to say that all of the cups are logical!"	IsAngryWith (Housekeeper, Frege) AND Despairs(Frege)
14	Frege realises that he could just use a variable x that can stand in for any of the cups. Use the relation to say that Frege a cup.	Fantastic!	Has(Frege, x)
15	Logician(x). x can be any object in the scene. If x is Frege then Logician(x) is true. If x is Aristotle, then Logician(x) is false because Aristotle is a bear and bears cannot be logicians.		

16	To make talking about variables easier, we use two more symbols. The first symbol is ∃. ∃means 'there exists'. ∃x.Happy(x) means that we can find something to make x that would make Happy(x) true.		
17	Make the sentence true.	Well done!	∃x Happy(x)
18	If we have more than one,object without a name, then we can use more than one variable.	We could give x and y the same value, or they could be different.	∃xy. Happy(x) AND Likes(Frege, Aristotle)
19	If the same variable is used more than once, it must stand in for the same object every time.	Super!	∃xy. Logician(x) AND Likes(x,y)
20	The next symbol we can use is ∀. ∀ means 'for all'. ∀x.Sad(x) means that every object in the scene is sad. If Aristotle is in the scene, then this is not true because bears cannot be sad.		
21	If we take Aristotle out of,the scene then we can make this sentence true.	Fantastic!	∀x. Sad(x)
22	Frege uses all of these symbols to write down his theory in his book – the Begreschifft. Drag the theory book into Frege's thoughts...	Very good!	Has(Frege, Frege's Theory)
23	Frege is happy because he has,fixed logic	Well that's good news!	∃x Happy(x)
23	Hopefully none of the other,logicians will find a mistake in it...		

Table A.3: Rationale for final level design

1	Sentences	To Teach	Rationale
1	Logician(Frege)	Predicate	Simplest level. Most icons omitted from user interface to make ui familiarisation easier.
2	Sad(Frege)	Predicate	Practice predicate, progress story.
3		Relations	Introduce relations theory
3	Likes(Frege, Aristotle)	Relations	Introduce relations user interface, with reduced action required from the user.
4	Has(Frege, Idea)	Relations	Practice relations, introduce standard ui for the relations arrow.
5		Objects	Introduce concept of objects.
5	Picture with logic(Frege) and logic(Aristotle)	Objects	More objects and predicates. Progress story.
6	Has(Frege, Idea)	Objects	More object theory, progress story, practice relations
7	Likes(Frege, Aristotle) $\wedge$ Logician(Frege)	$\wedge$	Introduce $\wedge$ and practice $\wedge$.
8		$\vee$	Introduce OR.
9	Has(Frege, Frege's pen) $\vee$ Has(Frege, Frege's pencil)	$\vee$	Practice $\vee$.
10	Has(Frege, Tea) $\vee$ Has(Frege, Hot Chocolate)	$\vee$	Practice OR.
11	Alarmed(Frege),[picture of a stack of cups]	Variables	Progress story whilst introducing variables.
12	[picture of cups all with names] Despairs(Frege)	Variables	Explain need for variables.
13	IsAngryWith(,Housekeeper, Frege) $\wedge$ Despairs(Frege)	Variables	Explain basic idea of variables.
14	Has(Frege, x)	Variables	Introduce variable x and use it.
15		Variables	Further explanation of variables.
16		$\exists$	Introduce $\exists$
17	$\exists$ x Happy(x)	$\exists$	Practice $\exists$
18	$\exists$ xy. Happy(x) $\wedge$ Likes(Frege,Aristotle)	Variables	Introduce use of multiple variables.
19	$\exists$xy. Logician(x) $\wedge$ Likes(x,y)	Variables	Introduce use of repeated variable.
20		$\forall$	Introduce $\forall$
21	$\forall$x. Sad(x)	$\forall$	Practice $\forall$
22	Has(Frege, Frege's Theory)	Relations	Progress story, revise relations.
23	$\exists$x Happy(x)		Progress story, revise $\exists$
23			Leave cliff-hanger.

Appendix B

Code

The code for the game is too long to include in its entirety, so given in this appendix are shared.js, FOL.js and the html and javascript pages for levels 1 and 3 (level 1 has a single character, level 3 includes a relation).

B.1 File: FOL.js

FOL.js contains JavaScript functions for evaluating a list of first order logic sentences against a context.

```
const variables=["Frege", "Russell"];

function lookup(context, predicate){
    let returnValue = false;

    $.each( context, function( i, l ){
        let temp=i;
        let temp2=l;

        let currentTerm=$(this);
        if (predicate==l){
            returnValue=true;
        }
    });

    return returnValue;
}

function eval(context, formula){
    let tempFormula = formula;
    let returnValue = "error";
    switch(tempFormula.type){

        case "predicate":

            let fullFormula =
                contextElementMaker(formula.
                formula, formula.args);
            returnValue = lookup(context,
                fullFormula);
            break;
        case "binary":
            switch(formula.formula){
                case "AND":
                    returnValue = eval(context,
                        formula.pred1) && eval(
                        context, formula.pred2);
                    break;

                case "OR":
                    returnValue = eval(context,
                        formula.pred1) || eval(
                        context, formula.pred2);
                    break;

                default:
                    returnValue="binary formula
                        not recognised"
            }
            break;

        case "forAll":

            returnValue = subAllVariables(
                context, formula);
            break;

        case "thereExists":

            returnValue = subAllVariables(
                context, formula);
            break;

        case "negation":
```

```
            returnValue = !eval(context,formula.
                pred1);
            break;

        default:

            returnValue = "eval error";
            break;
    }
    return returnValue;



}

function forFolPrinting(){
    document.getElementById("compOutput").
        innerHTML=eval(context, pred11);
}
function contextElementMaker(formula, args){
    let predicate = String(formula)+"(";

        $.each( args, function( i, l ){
            let temp2=i;
            let temp3=l;

            let arg=l;
            predicate = predicate+String(arg)+"
                ";
        });

    predicate = predicate+")";
    return predicate;
}
function subAllVariables(context, sentence){
    let tempPred = sentence.pred1;
const replaceVar = sentence.replaceVar;
let returnValue = "not assigned";
switch (tempPred.type) {
    case "negation":
        if (sentence.formula == "forAll"){
            returnValue = true;

            for(let i=0; i<variables.length;
                i++){
                const variable = variables[i
                    ];
                let freshSentence = $.extend
                    (true,{},tempPred);
                let newPredicate =
                    subVariable(replaceVar,
                    freshSentence, variable);

                if (!eval(context,
                    newPredicate)){
                    returnValue = false;
                }
            }
        }
        else if (sentence.formula == "
            thereExists"){
            returnValue = false;

            for(let i=0; i<variables.length;
                i++){
                const variable = variables[i
                    ];
                let freshSentence = $.extend
                    (true,{},tempPred);
                let newPredicate =
                    subVariable(replaceVar,
                    freshSentence, variable);
```

```
            if (eval(context,
                newPredicate)){
                returnValue = true;
            }
        }
    }
    break;
case "predicate":
    returnValue = "subAllVariables error
        ";
    switch (sentence.formula){
        case "thereExists":
            returnValue = false;

            for(let i=0; i<variables.
                length; i++){
                const variable =
                    variables[i]; //gets
                    a variable
                let freshTempPred = $.
                    extend(true,{},
                    tempPred); //makes a
                    copy of temppred
                let newPredicate =
                    subVariable(
                    replaceVar,
                    freshTempPred,
                    variable); //subs
                    variable in
                if (eval(context,
                    newPredicate)){
                    returnValue = true;
                }
            }
            break;
        case "forAll":
            returnValue = true;
            for(let i=0; i<variables.
                length; i++){
                const variable =
                    variables[i];
                let freshSentence = $.
                    extend(true,{},
                    tempPred);
                let newPredicate =
                    subVariable(
                    replaceVar,
                    freshSentence,
                    variable);
                if (!eval(context,
                    newPredicate)){
                    returnValue = false;
                }
            }
            break;
        default:
            returnValue = "default error
                ";
            break;
    }
    break;

case "binary":

    if (sentence.formula == "forAll"){
        returnValue = true;

        for(let i=0; i<variables.length;
            i++){
            const variable = variables[i
                ];
```

```
            let freshSentence = $.extend
                (true,{},tempPred);
            let newPredicate =
                subVariable(replaceVar,
                freshSentence, variable);

            if (!eval(context,
                newPredicate)){
                returnValue = false;
            }

        }
    }
    else if (sentence.formula == "
        thereExists"){
        returnValue = false;

        for(let i=0; i<variables.length;
            i++){
            const variable = variables[i
                ];
            let freshSentence = $.extend
                (true,{},tempPred);
            let newPredicate =
                subVariable(replaceVar,
                freshSentence, variable);

            if (eval(context,
                newPredicate)){
                returnValue = true;
            }
        }
    }
    else{
        returnValue = "Binary formula
            error";
            }
            break;

        default:
        returnValue = "subAllVariables error";
            break;

    }
    return returnValue;

}


function subVariable(replaceVar, sentence,
    variable){
    const subPredicate = sentence;
    let returnValue = "notChanged";
    let subPredicateType=subPredicate.type;

    switch (subPredicateType){

        case "predicate":
            //this is the base case
            for (let i = 0; i < subPredicate.
                args.length; i++) {
                let currentArg = String(
                    subPredicate.args[i]);
                if (currentArg==replaceVar){
                    subPredicate.args[i] =
                        variable;
                }
            }
            returnValue= subPredicate;
            break;

        case "binary":
```

```
            let firstPred = subPredicate.pred1;
            let secondPred = subPredicate.pred2;
            if (subPredicate.formula == "AND" ||
                subPredicate.formula == "OR"){
                firstPred = subVariable(
                    replaceVar, firstPred,
                    variable);
                secondPred = subVariable(
                    replaceVar, secondPred,
                    variable);
                returnValue = subPredicate;
            }
            else{
                returnValue = "Binary
                    subVariable error";
            }
            break;
        case "negation":
            let singlePred = subPredicate.pred1;
            singlePred = subVariable(replaceVar,
                singlePred, variable);
            returnValue = subPredicate;
            break;
        default:
            returnValue = "subVariable error";
            break;
    }
    return returnValue;
}
```

B.2 File: shared.js

Shared.js contains JavaScript functions shared across levels that do not relate directly to first order logic. This mostly involves methods that facilitate graphical interactions, such as drag and drop.

```
let predicates=["Happy","Logician","Sad","
    Alarmed","Despairs"];
let relations=["Likes","IsAngryWith","Has"];
let objects=["Theory","Idea","Tea","HotChoc","
    Pen","Pencil", "HasTheory"];
$('[rel="tooltip"]').on('click', function () {
    $(this).tooltip('hide')
})
function allowDrop(ev) {
    ev.preventDefault();
}

function drag(ev) {
    ev.dataTransfer.setData("text", ev.target.id
        );
}

function drop(ev) {

    ev.preventDefault();
    let data = ev.dataTransfer.getData("text");
    if (predicates.indexOf(String(data))>-1 ||
        objects.indexOf(String(data))>-1 ||
        String(data).charAt(0)=='x'){
         let image = document.getElementById(data
             ),
              clone = image.cloneNode(true);
```

```
            clone.id = "cloned"+String(counter);
            counter++;
            ev.target.appendChild(clone);
        }
        else{
            $('#unaryAlert').show();
        }

}
function dropB(ev) {
    ev.preventDefault();
    let data = ev.dataTransfer.getData("text");
    if (relations.indexOf(String(data))>-1){
        let image = document.getElementById(data
            ),
             clone = image.cloneNode(true);
        clone.id = "cloned"+String(counter);
        counter++;
        ev.target.appendChild(clone);
    }
    else{
        $('#binaryAlert').show();
    }

}
function dropArrow(event) {

    event.preventDefault();
    let data = event.dataTransfer.getData("text
        ");
    if (String(data)=="topArrow"||String(data)
        =="bottomArrow"||String(data)==""
        startArrowBox"){
         let image = document.getElementById(data
            ),
             clone = image.cloneNode(true);
            clone.id = "cloned"+String(counter);
            counter++;
            event.target.appendChild(clone);
        }
        event.target.style.border= "";

}

function triggerNextLevelDialogue(){
    $('#nextLevelDialogue').modal('show');
}
function triggerTryAgainDialogue(){
    $('#tryAgainDialogue').modal('show');

}
$(document).dblclick(function(event) {
    var element=event.target.id;
    let x=String(event.target.tagName);
    let target = event.target;
    while(x==="TD" || x==="TR" ||x==="TBODY" ||x
        ==="TABLE") {
         let tempElement = target.parentNode;
         x = String(tempElement.tagName);
         target = tempElement;
    }
    let parent = (target.parentNode.id);
    if(parent=="arrowBox"||parent==""
        bottomArrowBox"||parent=="topArrowBox"){
         document.getElementById(parent).style.
             border="";
    }
    let child = target.id;
    removeElement(child, parent);
});
function removeElement(element, parent) {
```

```
    var holder = document.getElementById(parent)
        ;
    var image = document.getElementById(element)
        ;
    let y = image.parentNode;
    holder.removeChild(image);
}
function addBorder(event){
    let eventId = event.target.id;
    if (eventId =="arrowBox"||eventId=="
        bottomArrowBox"||eventId=="topArrowBox"){
        event.target.style.border= "10px dotted
            green";
    }
}
function removeBorder(event){
    let eventId = event.target.id;
    if (eventId =="arrowBox"||eventId=="
        bottomArrowBox"||eventId=="topArrowBox"){
        event.target.style.border= "";
    }
}
```

B.3 File: Level1.html

Level1.html is the simplest level, and as such its short amount of JavaScript was included in the main html file.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <script src="http://ajax.googleapis.com/ajax
        /libs/jquery/2.0.0/jquery.min.js" type="
        text/javascript"></script>

    <script src="../FOL.js" type="text/
        javascript"></script>

    <!-- Latest compiled and minified CSS -->
    <link rel="stylesheet" href="https://maxcdn.
        bootstrapcdn.com/bootstrap/3.3.7/css/
        bootstrap.min.css" integrity="sha384-
        BVYiiSIFeK1dGmJRAkycuHAHRg32OmUcww7on3
        RYdg4Va+PmSTsz/K68vbdEjh4u" crossorigin="
        anonymous">

    <!-- Optional theme-->
    <link rel="stylesheet" href="https://maxcdn.
        bootstrapcdn.com/bootstrap/3.3.7/css/
        bootstrap-theme.min.css" integrity="sha
        384-rHyoN1iRsVXV4nD0JutlnGaslCJuC7uwjduW9
        SVrLvRYooPp2bWYgmgJQIXwl/Sp" crossorigin
        ="anonymous">

    <!-- Latest compiled and minified JavaScript
        -->
    <script src="https://maxcdn.bootstrapcdn.com
        /bootstrap/3.3.7/js/bootstrap.min.js"
        integrity="sha384-Tc5IQib027
        qvyjSMfHjOMaLkfuWVxZxUPnCJA7l2mCWNIpG9
        mGCD8wGNIcPD7Txa" crossorigin="anonymous
        "></script>

    <link rel="stylesheet" type="text/css" href
        ="../Levels.css" media="screen">
    <title>Level 1</title>
    <script src="..\shared.js"></script>

    <script type="text/javascript">
```

```
$(window).load(function(){
    $('#level1Modal').modal('show');
});

let context = [];
let sentence1 = {type:"predicate",
   formula:"Logician", args:["Frege"]};
let sentences = [sentence1];
let counter=0;
let russellEmotion;
function makeContext(){
    context = [];
    let russellEmotionBox = document.
       getElementById("russellEmotion").
       firstElementChild;
    russellEmotion = russellEmotionBox ?
        russellEmotionBox.className :
       false;


    if (russellEmotion){
        context.push(russellEmotion+"(
           Frege )");
    }

    let output = true;

    $( sentences ).each(function() {
        let sentence=$(this)[0];
        if (!eval(context, sentence)){
            output = false;
        }
    });
    if(output){
        triggerNextLevelDialogue();
    }
        else{
            triggerTryAgainDialogue();
        }
    }


    </script>
</head>
<body>
<div class="modal fade" id="level1Modal"
   tabindex="-1" role="dialog" aria-labelledby="
   myModalLabel">
    <div class="modal-dialog" role="document">
        <div class="modal-content">
            <div class="modal-header">
                <button type="button" class="
                   close" data-dismiss="modal"
                   aria-label="Close"><span aria
                   -hidden="true">&times;</span
                   ></button>
                <h4 class="modal-title" id="
                   myModalLabel">Instructions</h
                   4>
            </div>
            <div class="modal-body">
                <p>
                    Gottlob Frege is a logician.
                        We can say this by
                       writing Logician(Frege).
                </p>
                <p>
                    Frege is always thinking
                       about logic - make the
                       sentence Logician(Frege)
                       true by dragging the
```

```
                    logic symbol into Frege's
                        thoughts.
                </p>

            </div>
            <div class="modal-footer">
                <button type="button" class="btn
                    btn-primary" data-dismiss="
                    modal">Got it!</button>
            </div>
        </div>
    </div>
</div>
<div class="modal fade" id="nextLevelDialogue"
    data-backdrop="static" data-keyboard="false"
    tabindex="-1" role="dialog" aria-labelledby="
    myModalLabel">
    <div class="modal-dialog" role="document">
        <div class="modal-content">
            <div class="modal-header">
                <h4 class="modal-title">Awesome
                    !</h4>
            </div>

            <div class="modal-footer">
                <a href="Level2.html" class="btn
                    btn-primary">Go to next
                    level</a>                </
                    div>
        </div>
    </div>
</div>
<div class="modal fade" id="tryAgainDialogue"
    tabindex="-1" role="dialog" aria-labelledby="
    myModalLabel">
    <div class="modal-dialog" role="document">
        <div class="modal-content">
            <div class="modal-header">
                <button type="button" class="
                    close" data-dismiss="modal"
                    aria-label="Close"><span aria
                    -hidden="true">&times;</span
                    ></button>
                <h4 class="modal-title">Bad luck
                    !</h4>
            </div>
            <div class="modal-body">
                <p>You haven't got every
                    sentence true yet.</p>
                <p>The symbol for logician is
                    the chess piece.</p>
            </div>
            <div class="modal-footer">
                <button type="button" class="btn
                    btn-primary" data-dismiss="
                    modal">Try again.</button>
            </div>
        </div>
    </div>
</div>
<div class="modal fade" id="singleHelp"
    tabindex="-1" role="dialog" aria-labelledby="
    singleHelp">
    <div class="modal-dialog" role="document">
        <div class="modal-content">
            <div class="modal-header">
                <button type="button" class="
                    close" data-dismiss="modal"
                    aria-label="Close"><span aria
                    -hidden="true">&times;</span
                    ></button>
```

```
                <h4 class="modal-title">Help</h
                    4>
            </div>
            <div class="modal-body">
                <p>Select one of the symbols
                    from the bottom and drag it
                    to the thought bubble.</p>
                <img src="../images/help1.png">
                <p>If you make a mistake, double
                     click an item to delete it
                    and then try again.</p>
            </div>
            <div class="modal-footer">
            </div>
        </div>
    </div>
</div>
<div class = "container-fluid">
    <div class="row">
        <div class="col-md-12" style="text-align
            :center">
            <h2>Logician(Frege)</h2>
        </div>
    </div>
    <div class="row">
        <div class="col-md-1"></div>
        <div class="col-md-9">
            <div class="holder">
                <img src="../images/frege.png"
                    id="russell" width="250"
                    height="300">
                <div class="box receiverBox" id
                    ="russellEmotion" ondrop="
                    drop(event)" ondragover="
                    allowDrop(event)" data-
                    toggle="tooltip" data-
                    placement="left" title="Frege
                    's thoughts"></div>
            </div>
        </div>
    </div>
</div>
<div class="row">
    <div class="col-md-1"> </div>
    <div class="col-md-9" style="margin-left
        :50px">
        <div class="sentences">
            <button class="btn btn-success"
                id="contextMaker" onclick="
                makeContext()">Check</button>
            <button type="button" class="btn
                 btn-info" data-toggle="modal
                " data-target="#level1Modal">
                Instructions</button>
                        <button type="
                            button" class
                            ="btn btn-
                            info" data-
                            toggle="modal
                            " data-target
                            ="#singleHelp
                            ">Help</
                            button>


        </div>
    </div>
</div>
<div class="row">
    <div class="col-md-1"> </div>
    <div class="col-md-9" style="margin-left
        :50px">
```

```
            <div class="box startBox" id="
                startEmotion" ondrop="drop(event)
                " ondragover="allowDrop(event)">
                 <img class="Logician" src="../
                     images/logic.png" draggable="
                     true" ondragstart="drag(event
                     )" id="Happy" width="88"
                     height="31" data-toggle="
                     tooltip" data-placement="
                     bottom" title="Logician()">
            </div>
        </div>
    </div>
</div>
</body>
</html>
```

B.4 File: Level3.html

Level3.html gives an example of the html for a more complex level, this time involving relations.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <script src="http://ajax.googleapis.com/ajax
        /libs/jquery/2.0.0/jquery.min.js" type="
        text/javascript"></script>
    <script>$.widget.bridge('uitooltip', $.ui.
        tooltip);</script>
    <!-- Latest compiled and minified CSS -->
    <link rel="stylesheet" href="https://maxcdn.
        bootstrapcdn.com/bootstrap/3.3.7/css/
        bootstrap.min.css" integrity="sha384-
        BVYiiSIFeK1dGmJRAkycuHAHRg32OmUcww7on3
        RYdg4Va+PmSTsz/K68vbdEjh4u" crossorigin="
        anonymous">

<!-- Optional theme-->
<link rel="stylesheet" href="https://maxcdn.
    bootstrapcdn.com/bootstrap/3.3.7/css/
    bootstrap-theme.min.css" integrity="sha
    384-rHyoN1iRsVXV4nD0JutlnGaslCJuC7uwjduW9
    SVrLvRYooPp2bWYgmgJQIXwl/Sp" crossorigin
    ="anonymous">

<!-- Latest compiled and minified JavaScript
    -->
<script src="https://maxcdn.bootstrapcdn.com
    /bootstrap/3.3.7/js/bootstrap.min.js"
    integrity="sha384-Tc5IQib027
    qvyjSMfHjOMaLkfuWVxZxUPnCJA7l2mCWNIpG9
    mGCD8wGNIcPD7Txa" crossorigin="anonymous
    "></script>
<script src="https://cdnjs.cloudflare.com/
    ajax/libs/jsPlumb/2.2.9/jsplumb.js"></
    script>
<script src="..\shared.js"></script>

<link rel="stylesheet" type="text/css" href
    ="../Levels.css" media="screen">
<link rel="stylesheet" type="text/css" href
    ="../2Item.css" media="screen">

<title>Level 3</title>
<script src="..\FOL.js" type="text/
    javascript"></script>
```

```
    <script src="..\Level3.js" type="text/
        javascript"></script>
</head>
<div class="alert alert-danger alert-dismissible
     collapse" id="unaryAlert" role="alert">
    <button type="button" class="close" data-
        dismiss="alert" aria-label="Close"><span
        aria-hidden="true">&times;</span></button
        >
    <strong>Oops!</strong> That box is for
        predicates that describe one thing.
    <p>Remember that Likes() and IsAngryWith()
        talk about two people</p>

</div>
<div class="alert alert-danger alert-dismissible
     collapse" id="binaryAlert" role="alert">
    <button type="button" class="close" data-
        dismiss="alert" aria-label="Close"><span
        aria-hidden="true">&times;</span></button
        >
    <strong>Oops!</strong> <p>That box is for
        predicates that describe two things.</p>
    <p>Things like Happy() and Logician() can
        only talk about one thing at a time</p>
</div>
<div class="modal fade" id="doubleHelp" tabindex
    ="-1" role="dialog" aria-labelledby="
    singleHelp">
    <div class="modal-dialog" role="document">
        <div class="modal-content">
            <div class="modal-header">
                <button type="button" class="
                    close" data-dismiss="modal"
                    aria-label="Close"><span aria
                    -hidden="true">&times;</span
                    ></button>
                <h4 class="modal-title">Help</h
                    4>
            </div>
            <div class="modal-body">
                <p>Drag the arrow by clicking
                    the box in the middle</p>
                <img src="../images/help2.png"
                    width="50%" height="50%"
                    style="border: 1pt solid
                    black">
                <p>You will see a green outline
                    when it is in the right place
                    .</p>
                <img src="../images/help3.png"
                    width="50%" height="50%"
                    style="border: 1pt solid
                    black">
                <p>Remember that if you make a
                    mistake, you can delete
                    things by double clicking
                    them.</p>
            </div>
            <div class="modal-footer">
            </div>
        </div>
    </div>
</div>
<div class="modal fade" id="instructionModal"
    data-backdrop="static" data-keyboard="false"
    tabindex="-1" role="dialog" aria-labelledby="
    myModalLabel">
    <div class="modal-dialog" role="document">
        <div class="modal-content">
            <div class="modal-header">
```

```
                <button type="button" class="
                    close" data-dismiss="modal"
                    aria-label="Close"><span aria
                    -hidden="true">&times;</span
                    ></button>
                <h4 class="modal-title" >
                    Instructions</h4>
            </div>
            <div class="modal-body">
                <p>Frege has a teddy bear named
                    after one of the most famous
                    logicians in history -
                    Aristotle.</p>
                <p>We can talk about this using
                    a <strong>relation</strong
                    >.</p>
            </div>
            <div class="modal-footer">
                <button type="button" class="btn
                     btn-info" data-dismiss="
                    modal" data-toggle="modal"
                    data-target="#modal2">Next</
                    button>


            </div>
        </div>
    </div>
</div>
<div class="modal fade" id="modal2" tabindex
    ="-1" role="dialog" aria-labelledby="
    myModalLabel">
    <div class="modal-dialog" role="document">
        <div class="modal-content">
            <div class="modal-header">
                <button type="button" class="
                    close" data-dismiss="modal"
                    aria-label="Close"><span aria
                    -hidden="true">&times;</span
                    ></button>
                <h4 class="modal-title" id="
                    myModalLabel">Instructions</h
                    4>
            </div>
            <div class="modal-body">
                <p>
                    A relation talks about how
                        two objects relate to
                        each other.
                </p>
                <p>
                    Drag the arrow up to between
                         Frege and Aristotle, and
                         add an emotion.
                </p>
                                <p>
                    Remember, if you make a
                        mistake, double click a
                        symbol to delete it.
                </p>
            </div>
            <div class="modal-footer">
                <button type="button" class="btn
                     btn-primary" data-dismiss="
                    modal">Got it!</button>
            </div>
        </div>
    </div>
</div>
<div class="modal fade" id="nextLevelDialogue"
    data-backdrop="static" data-keyboard="false"
     tabindex="-1" role="dialog" aria-labelledby
    ="myModalLabel">
```

```
    <div class="modal-dialog" role="document">
        <div class="modal-content">
            <div class="modal-header">
                <h4 class="modal-title">
                    Congratulations</h4>
            </div>
            <div class="modal-body">
                <p>You made the sentence true</p
                    >
            </div>
            <div class="modal-footer">
                <a href="Level4.html" class="btn
                     btn-primary">Go to next
                    level</a>                </
                    div>
        </div>
    </div>
</div>
<div class="modal fade" id="tryAgainDialogue"
    tabindex="-1" role="dialog" aria-labelledby="
    myModalLabel">
    <div class="modal-dialog" role="document">
        <div class="modal-content">
            <div class="modal-header">
                <button type="button" class="
                    close" data-dismiss="modal"
                    aria-label="Close"><span aria
                    -hidden="true">&times;</span
                    ></button>
                <h4 class="modal-title">Bad luck
                    !</h4>
            </div>
            <div class="modal-body">
                <p>You haven't got every
                    sentence true yet.</p>
                <p>Drag the arrow from Frege to
                    the teddy bear</p>
            </div>
            <div class="modal-footer">
                <button type="button" class="btn
                     btn-primary" data-dismiss="
                    modal">Try again.</button>
            </div>
        </div>
    </div>
</div>
<body>
<div class = "container-fluid">
    <div class="row">

        <div class="col-md-12" style="text-align
            :center">

            <h2>Likes(Frege, Aristotle)</h2>
        </div>
    </div>
    <div class="row">
        <div class="col-md-1">
        </div>
        <div class="col-md-9">
            <div class="holder">

                <div class="box receiverBox" id
                    ="russellEmotion" ondrop="
                    drop(event)" ondragover="
                    allowDrop(event)"></div>
                <img src="../images/frege.png"
                    id="russell">

                <div class="box receiverBox
                    arrowBox topArrowBox" width
```

```
                ="400" height="200" ondrop="
                dropArrow(event)" ondragover
                ="allowDrop(event)"
                ondragenter="addBorder(event)
                " ondragleave="removeBorder(
                event)" id="arrowBox"></div>


            <img src="../images/aristotle.
                png" onclick="this.src='../
                images/aristotle.png'" id="
                frege">
        </div>
    </div>

</div>
<div class="row">
    <div class="col-md-1">

    </div>
    <div class="col-md-9" style="margin-left
        :50px">
        <button class="btn btn-success" id="
            contextMaker" onclick="
            makeContext()">Check.</button>
        <button type="button" class="btn btn
            -info" data-toggle="modal" data-
            target="#instructionModal">
             Instructions
        </button>
        <button type="button" class="btn btn
            -info" data-toggle="modal" data-
            target="#doubleHelp">Help</button
            >

    </div>
</div>
<div class="row">
    <div class="col-xs-1"></div>
    <div class="col-xs-3">

    </div>
    <div class="col-xs-3">
        <div class="box startBox" id="
            heartStart" ondrop="dropB(event)"
             ondragover="allowDrop(event)">
            <img class="Likes" src="../
                images/heart.png" draggable="
                true" ondragstart="drag(event
                )" id="Likes" width="88"
                height="31" data-toggle="
                tooltip" data-placement="
                bottom" title="Likes()">
        </div>
        <div class="box startBox" id="
            lightningStart" ondrop="dropB(
            event)" ondragover="allowDrop(
            event)">
            <img class="IsAngryWith" src
                ="../images/lightning.png"
                draggable="true" ondragstart
                ="drag(event)" id="
                IsAngryWith" width="88"
                height="31" data-toggle="
                tooltip" data-placement="
                bottom" title="IsAngryWith()
                ">
        </div>
    </div>
    <div class="col-xs-3">

    </div>
```

```
</div>
<div class="row">
    <div class="col-xs-3"></div>
    <div class="col-xs-4">
        <div class="startArrowBox" id="
            startArrowBox" draggable="true"
            ondragstart="drag(event)" id="
            topArrow" width="500" height
            ="100" >

            <div class="topArrow">
                <div class="box1 receiverBox
                    " id="russellFrege"
                    ondrop="dropB(event)"
                    ondragover="allowDrop(
                    event)" data-toggle="
                    tooltip" data-placement="
                    top" title="Frege's
                    feelings towards
                    Aristotle">
                    <img class="Likes" src
                        ="../images/heart.png
                        " draggable="true"
                        ondragstart="drag(
                        event)" id="Likes1"
                        width="88" height
                        ="31" data-toggle="
                        tooltip" data-
                        placement="bottom"
                        title="Likes()">

                </div>
            </div>

        </div>
            </div>

        </div>


</div>
</body>
</html>
```

B.5 File: Level3.js

The JavaScript for Level 3 is more complex than for level 1, so it is separated into a different file.

```
$(window).load(function(){
    $('#instructionModal').modal('show');
});
$(document).ready(function(){
    $('[data-toggle="tooltip"]').tooltip();
});
let context = [];
let sentence2 = {type:"predicate", formula:"
   Likes", args:["Frege", "Aristotle"]};
let sentences = [sentence2];
let counter = 0;


function makeContext(){
    context = [];
    let russellEmotion, fregeEmotion,
        russellFrege, fregeRussell;
    if (document.getElementById("russellEmotion
        ")) {
```

```
    let russellEmotionBox = document.
        getElementById("russellEmotion").
        firstElementChild;

    russellEmotion = russellEmotionBox ?
        russellEmotionBox.className : false;
    console.log(russellEmotion);
}

if (document.getElementById("fregeEmotion"))
    {
    let fregeEmotionBox = document.
        getElementById("fregeEmotion").
        firstElementChild;
    fregeEmotion = fregeEmotionBox ?
        fregeEmotionBox.className : false;
}


if (document.getElementById("russellFrege"))
    {
    let russellFregeBox = document.
        getElementById("russellFrege").
        firstElementChild;
    russellFrege = russellFregeBox ?
        russellFregeBox.className : false;
    if (!(document.getElementById("arrowBox
        ").hasChildNodes()))
    {
        russellFrege=false;
    }
}

if(document.getElementById("fregeRussell")){
    let fregeRussellBox = document.
        getElementById("fregeRussell").
        firstElementChild;
    fregeRussell = fregeRussellBox ?
        fregeRussellBox.className : false;
}

if (russellEmotion){
    context.push(russellEmotion+"(Russell )
        ");
}
if(fregeEmotion){
    context.push(fregeEmotion+"(Frege )");
}
if(russellFrege){
    context.push(russellFrege+"(Frege
        Aristotle )");
}

let output = true;

$( sentences ).each(function() {
    let sentence=$(this)[0];
    if (!eval(context, sentence)){
        output = false;
    }
});
if(output){
    triggerNextLevelDialogue();
}
else{
    triggerTryAgainDialogue();
}

}
function addBorder(event){
    console.log(event.target);
    let eventId = event.target.id;
```

```
    if (eventId =="arrowBox"||eventId=="
        bottomArrowBox"||eventId=="topArrowBox"){
         event.target.style.border= "10px dotted
             green";
    }
}
function removeBorder(event){
    console.log(event.target);
    let eventId = event.target.id;
    if (eventId =="arrowBox"||eventId=="
        bottomArrowBox"||eventId=="topArrowBox"){
         event.target.style.border= "";
    }
}
function allowDrop(ev) {
    ev.preventDefault();
}

function drag(ev) {

    ev.dataTransfer.setData("text", ev.target.id
        );
}

function drop(ev) {

    ev.preventDefault();
    let data = ev.dataTransfer.getData("text");
    if (String(data)=="Happy" || String(data)="
        HasTheory" || String(data)=="Logician" ||
         String(data)=="Sad" || String(data)="
        HasIdea"){

         let image = document.getElementById(data
             );
             clone = image.cloneNode(true); //
                 true means clone all childNodes
                 and all event handlers
         clone.id = "cloned"+String(counter);
         counter++;
         ev.target.appendChild(clone);
    }
    else{
        $('#unaryAlert').show();

    }

}
function dropB(ev) {
    ev.preventDefault();
    let data = ev.dataTransfer.getData("text");
    if (relations.indexOf(String(data))>-1){
        let image = document.getElementById(data
            );
             clone = image.cloneNode(true); //
                 true means clone all childNodes
                 and all event handlers
         clone.id = "cloned"+String(counter);
         counter++;
         ev.target.appendChild(clone);
    }
    else{
        $('#binaryAlert').show();
    }

}
function dropArrow(event) {

    event.preventDefault();
    let data = event.dataTransfer.getData("text
        ");
```

```
    if (String(data)=="topArrow"||String(data)
       =="bottomArrow"||String(data)=="
       startArrowBox"){

        //event.target.appendChild(document.
            getElementById(data).cloneNode(true))
            ;
        let image = document.getElementById(data
            );
            clone = image.cloneNode(true); //
                true means clone all childNodes
                and all event handlers
        clone.id = "cloned"+String(counter);
        counter++;
        event.target.appendChild(clone);
        //need to put the counter code in here
    }
    event.target.style.border= "";

}


function triggerNextLevelDialogue(){
    $('#nextLevelDialogue').modal('show');
}
function triggerTryAgainDialogue(){
    $('#tryAgainDialogue').modal('show');

}
$(document).dblclick(function(event) {
    var element=event.target.id;
    let x=String(event.target.tagName);
    let target = event.target;
    while(x==="TD" || x==="TR" ||x==="TBODY" ||x
       ==="TABLE") {
        let tempElement = target.parentNode;
        x = String(tempElement.tagName);
        target = tempElement;
    }
    let parent = (target.parentNode.id);
    console.log(parent);
    if(parent=="arrowBox"||parent=="
       bottomArrowBox"||parent=="topArrowBox"){
        document.getElementById(parent).style.
            border="";
    }
    let child = target.id;
    removeElement(child, parent);
});
function removeElement(element, parent) {
    var holder = document.getElementById(parent)
        ;
    var image = document.getElementById(element)
        ;
    console.log(holder);
    console.log(image);
    let y = image.parentNode;
    holder.removeChild(image);
}
```

Appendix C

Ethics Checklists

C.1 Initial Testing Ethics

This chapter gives the 13-point ethics checklist for the initial user testing - 'think-aloud' testing with three users in person.

C.1.1 Checklist

UNIVERSITY OF BATH

Department of Computer Science

13-POINT ETHICS CHECK LIST

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

1. Have you prepared a briefing script for volunteers? You must explain to people what they will be required to do, the kind of data you will be collecting from them and how it will be used.

A paper briefing script will be given before the test begins.

2. Will the participants be using any non-standard hardware? Participants should not be exposed to any risks associated with the use of non-standard equipment: anything other than pen and paper or typical interaction with PCs on desks is considered non-standard.

There will be no non-standard hardware.

3. Is there any intentional deception of the participants? Withholding information or misleading participants is unacceptable if participants are likely to object or show unease when debriefed.

No - the briefing script details the purpose of the testing.

4. How will participants voluntarily give consent? If the results of the evaluation are likely to be used beyond the term of the project (for example, the software is to be deployed, or the data is to be published), then signed consent is necessary. A separate consent form should be signed by each participant.

Participants will give consent after reading the briefing forms. The test examines the user interface for this project in particular and will not be used outside of this project.

5. Will the participants be exposed to any risks greater than those encountered in their normal work life? Investigators have a responsibility to protect participants from physical and mental harm during the investigation. The risk of harm must be no greater than in ordinary life.
No, playing the game involves no activity that would not be normal for using a web browser.

6. Are you offering any incentive to the participants? The payment of participants must not be used to induce them to risk harm beyond that which they risk without payment in their normal lifestyle.
There are no offered incentive.
7. Are any of your participants under the age of 16? Parental consent is required for participants under the age of 16.
No

8. Do any of your participants have an impairment that will limit their understanding or communication? Additional consent is required for participants with impairments.
No

9. Are you in a position of authority or influence over any of your participants? A position of authority or influence over any participant must not be allowed to pressurise participants to take part in, or remain in, any experiment.
I am not in any position of authority over the participants.

10. Will the participants be informed that they could withdraw at any time? All participants have the right to withdraw at any time during the investigation. They should be told this in the introductory script.
The briefing script informs participants that they can withdraw at any time.

11. Will the participants be informed of your contact details? All participants must be able to contact the investigator after the investigation. They should be given the details of the Unit Lecturer or Supervisor as part of the debriefing.
Participants will be given my email address and my supervisor's email address in the debriefing.

12. Will participants be de-briefed? The student must provide the participants with sufficient information in the debriefing to enable them to understand the nature of the investigation.
Participants will be given a verbal debrief when the test is complete.

13. Will the data collected from the participants be stored in an anonymous form? All participant data (hard copy and soft copy) should be stored securely, and in anonymous form.
Notes will be made based on what the participant says, and these will be anonymous. There will be no video or audio recordings.

C.2 Questionnaire Ethics

This chapter gives the 13-point ethics checklist and briefing script used for the questionnaire. The questionnaire was completed by participants remotely, and was completed by adult participants and A-level students (aged 16-18). Also included is the letter notifying parents/guardians of the A-level students that the testing will take place.

C.2.1 Checklist

UNIVERSITY OF BATH

Department of Computer Science

13-POINT ETHICS CHECK LIST

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

1. Have you prepared a briefing script for volunteers? You must explain to people what they will be required to do, the kind of data you will be collecting from them and how it will be used.

An electronic briefing script will be given at the start of the questionnaire.

2. Will the participants be using any non-standard hardware? Participants should not be

exposed to any risks associated with the use of non-standard equipment: anything other than pen and paper or typical interaction with PCs on desks is considered non-standard.

There will be no non-standard hardware.

3. Is there any intentional deception of the participants? Withholding information or misleading participants is unacceptable if participants are likely to object or show unease when debriefed.

No - the briefing script details the purpose of the testing.

4. How will participants voluntarily give consent? If the results of the evaluation are likely to be used beyond the term of the project (for example, the software is to be deployed, or the data is to be published), then signed consent is necessary. A separate consent form should be signed by each participant.

The briefing script will invite users to click next only if they consent to the form. The test will be completed remotely, so there will be no pressure from researchers to continue.

5. Will the participants be exposed to any risks greater than those encountered in their normal work life? Investigators have a responsibility to protect participants from physical and mental harm during the investigation. The risk of harm must be no greater than in ordinary life.
No, playing the game and completing the questionnaire involves no activity that would not be normal for using a web browser. With regards to the school students, the teacher will be given a copy of the game and questionnaire before hand.

6. Are you offering any incentive to the participants? The payment of participants must not be used to induce them to risk harm beyond that which they risk without payment in their normal lifestyle.
There are no offered incentive.

7. Are any of your participants under the age of 16? Parental consent is required for participants under the age of 16.
No - the school students who will be part of the test are A-level students between the age of 16 and 18. They will be asked to consent to the survey themselves, but their parents/-guardians will be notified that the test will take place.

8. Do any of your participants have an impairment that will limit their understanding or communication? Additional consent is required for participants with impairments.
No

9. Are you in a position of authority or influence over any of your participants? A position of authority or influence over any participant must not be allowed to pressurise participants to take part in, or remain in, any experiment.
I am not in any position of authority over the participants.

10. Will the participants be informed that they could withdraw at any time? All participants have the right to withdraw at any time during the investigation. They should be told this in the introductory script.
The briefing script informs participants that they can withdraw at any time, and their responses will not be stored if they withdraw.
11. Will the participants be informed of your contact details? All participants must be able to contact the investigator after the investigation. They should be given the details of the Unit Lecturer or Supervisor as part of the debriefing.
Participants will be given my email address and my supervisor's email address in the debriefing. These are also included in the letter to the parents/guardians of the A-level students.

12. Will participants be de-briefed? The student must provide the participants with sufficient information in the debriefing to enable them to understand the nature of the investigation.
All of the necessary information on the investigation is given in the briefing before the experiment, the de-briefing reiterates the key points and provides contact details for myself and my supervisor

13. Will the data collected from the participants be stored in an anonymous form? All participant data (hard copy and soft copy) should be stored securely, and in anonymous form.
All questionnaire response are anonymous, and will be stored securely in a spreadsheet. There will be no hard copies of the data.

C.2.2 Briefing

The project being investigated is a serious game for teaching concepts relating to first order logic, a way of expressing facts about the world in terms of true/false statements.

During the study you will be asked to play a game that aims to teach first order logic in an accessible way, for up to twenty minutes. Before and after playing the game, you will be asked to complete a short questionnaire which will ask about your past experience with learning about maths and logic, and what you think of the game.

All data will be collected anonymously and none of the questions are compulsory. You can withdraw from the study at any time - if you choose not to complete the survey then your

responses will not be recorded.

You need to be 16 or above to participate in this survey.

If you would like to participate, click next to begin the first questionnaire.

C.2.3 Parent Letter

Dear parent/guardian

I am a student at the University of Bath and am examining the potential of using a serious game to teach first-order logic as part of my final year project. We are planning on testing the game with students in the A-level Computing class.

The test will involve the completion of a questionnaire before and after playing the game. No sensitive data will be collected and all data will be anonymised. The students do not have to participate if they choose not to.

If you have any questions or concerns please do not hesitate to contact me at alh62@bath.ac.uk, or my supervisor, Dr. Willem Heijltjes at W.B.Heijltjes@bath.ac.uk .

Yours faithfully

Amy Hooper

Appendix D

Initial User Testing

Table D.1: First User Observation

Level	Notes
1	Did not read instructions first time round, had to click to bring them back.
2	
3	Needed help with the user interface - did not understand how the drag and drop arrow worked.
4	
5	Read the instructions through three times before understanding.
6	
7	
8	
9	
10	
11	
12	
13	
14	
15	
16	
17	Took longer than usual
18	Took longer than usual
19	
20	
21	
22	
23	

Table D.2: Second User Observation

Level	Notes
1	Used the help
2	
3	Used the help, still took longer than other levels.
4	
5	
6	
7	
8	
9	
10	
11	
12	
13	
14	
15	
16	
17	Took longer than usual
18	Took longer than usual
19	
20	
21	
22	
23	

Appendix E

Raw Questionnaire Results

E.1 Questionnaire with adults

E.1.1 Results

The table below gives the raw data from the adult questionnaire. Capital letters have been substituted for text in order to fit the table on to a page. A key is given immediately after the table.

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
1	UG	Y	4	4	3	4	3	3	21	[T]	4	3	5	5	4	4	3	3
2	UG	N	5	5	5	4	4	3	23	No	5	5	5	5	5	5	1	3
3	AL	N	5	4	4	3	2	3	23	No	4	5	5	4	4	4	4	3
4	UG	Y	5	5	5	4	2	3	23	No	5	5	5	5	5	5	2	2
5	UG	Y	4	5	5	4	4	4	23	No	4	4	5	4	4	3	4	4
6	UG	N	4	3	4	2	4	4	23	19	4	3	5	4	4	5	3	4
7	AL	N	5	4	4	5	3	3	18	18	3	4	4	2	5	5	4	5
8	GCSE	N	4	2	1	3	4	5	23	no	3	2	4	4	2	4	5	5
9	AL	Y	4	4	2	4	5	4	23	[U]	4	2	4	4	4	5	3	4

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
10	AL	N	4	5	5	4	2	2	23	[V]	2	2	2	1	3	4	3	5

E.1.2 Key for adult results table

Letter	Meaning
A	Participant Number
B	What is the highest level of qualification that you have or are currently studying for in mathematics?
C	Have you ever heard of First Order Logic before?
D	I was good at maths at school
E	I enjoy logic puzzles
F	I enjoyed maths lessons at school
G	I am good at learning theoretical concepts
H	Logic puzzles are difficult
I	Maths is difficult
J	Which level did you reach?
K	Were any levels too difficult?
L	The game was easy to play
M	I always understood what I had to do to move on to the next level
N	I understood what all of the symbols used in the sentences meant
O	First order logic is easy to understand
P	The game was enjoyable
Q	The game was interesting
R	Logic puzzles are difficult
S	Maths is difficult

E.1.3 Comments from the adult questionnaire

Letter	
Comment T	No, though the first use of "exists" was a little unclear
Comment U	No, though occasional tooltip issues made it slightly confusing
Comment V	I just struggle to understand what was going on which made it difficult, by the end i got understood what to do so it wasnt difficult

E.2 Questionnaire with A-level students

E.2.1 Results

The table below gives the raw data from the A-level student questionnaire. Capital letters have been substituted for text in order to fit the table on to a page. A key is given immediately after the table.

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
1	17	Y	Y	N	N	4	5	4	4	3	2	23	No	5	4	4	4	4	3	2	2
2	17	Y	Y	Y	N	5	4	4	4	1	2	23	No	4	4	3	3	3	4	2	2
3	18	N	Y	N	N	2	3	4	2	2	3	19	Not really	4	2	2	3	5	5	3	3
4	17	Y	Y	Y	N	3	4	4	4	3	2	23	no	3	3	3	4	2	2	2	2
5	17	Y	Y	N	N	3	3	3	3	3	3	23	nope	3	3	3	3	2	3	3	3

E.2.2 Key for A-level results table

Letter	Meaning
A	Participant Number
B	What is your age?
C	Are you studying A-level Maths?
D	A-level Computing?
F	Have you ever heard of First Order Logic before?
G	I am good at maths
H	I enjoy logic puzzles
I	I enjoy/enjoyed maths lessons at school
J	I am good at learning theoretical concepts
K	Logic puzzles are difficult
L	Maths is difficult
M	What level did you reach?
N	Were any levels too difficult?
O	The game was easy to play
P	I always understood what I had to do to move on to the next level
Q	I understood what all of the symbols used in the sentences meant
R	First order logic is easy to understand
S	The game was enjoyable
T	The game was interesting
U	Logic puzzles are difficult
V	Maths is difficult