Sudoku Setter and Solver

Final Report

Submitted for the MEng in
Computer Science with Games Development

May 2016

by

Matthew Morris

Word Count: 4192

Table of Contents

1 Introduction

A Sudoku puzzle is a logic-based brainteaser, where the objective is to fill a 9x9 grid so that each column, row and sub-square (see fig. 1-1) contains each of the numbers from 1 to 9.

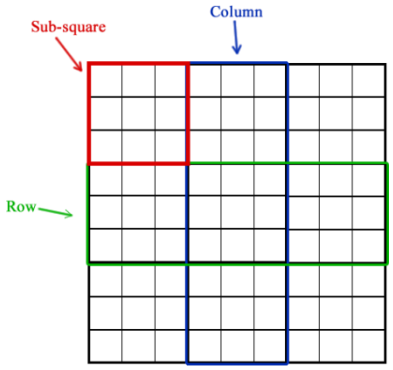


Fig 1-1

The grid will already contain some numbers that cannot be changed, called ‘givens’, ensuring that there is a unique solution. Fig. 1-2 shows a grid at the start of a Sudoku puzzle, displaying some squares filled with givens.

2	7		1	4				5
3		1	5			9		
6	4				9			
1				9	7	2		4
		8	3	2		1		9
9	2			1	5	7	3	
8		6	7	5			9	
4	3			6	1	5	2	
5	9	2			3	6		7

Fig 1-2

The player must then use a variety of strategies to attempt to solve the puzzle, ranging from basic elimination methods to more complicated strategies such as triplets, which will be discussed in detail in section 3.

The aim of this project is to develop a piece of software that can create and solve Sudoku puzzles, focusing on exploring the elements that define the differences between difficulty levels. The user will be able to choose a difficulty level for the puzzle, and will be given a set of tools to aid them in completing it, including the ability to check if entered values are correct, and the ability to note multiple possibilities for one square. If they get stuck, they will also have the option to choose individual squares to be automatically solved, or the option to auto-solve the whole puzzle. Furthermore, there will be a ‘challenge mode’ which allows the user to attempt to solve puzzles of an increasing difficulty that have already been generated and included with the software. The software will be developed in C# using Visual Studio and the XNA framework.

The rest of this report will explore the origins of Sudoku puzzles, outline the specific aims and objectives of the project, investigate solving techniques, discuss the design of the software and critically evaluate the project’s success.

2 Aim and Objectives

To create a piece of software that can generate and solve Sudoku puzzles, with an emphasis on investigating the differences between difficulty levels.

The above overall aim will be met by these following objectives:

Objective 1 – Development of code to generate and solve Sudoku puzzles, including designing a difficulty system based on techniques that a person would use in solving Sudoku puzzles.

- Development of algorithms to solve 9x9 Sudoku puzzles.
- Ability to generate valid 9x9 Sudoku puzzles within one second.
- Generation of a set of various puzzles from each difficulty level that are automatically included with the software.
- Exploration of techniques that can be used to solve Sudoku puzzles.
- Development of algorithms for solving Sudoku puzzles, which use techniques that a player would use.
- Investigation into the possible correlations between difficulty level and amount and distribution of givens.
- Creation of a difficulty level scale, based on the techniques that need to be used to solve the Sudoku puzzle.

Objective 2 – Development of an interactive interface for solving and creating Sudoku puzzles.

- Development and designing of a main menu, including buttons for play, instructions, and exit.
- Development and designing of an interface to display the instructions.
- Development and designing of an interface for playing the game, including:
    I. Displaying the current puzzle, with the initial numbers highlighted.
    II. An option to generate a new puzzle, with the option to choose the difficulty level.
    III. An option to save or load a puzzle.
    IV. The ability to place the numbers from 1 to 9 in the squares.
    V. The ability to erase previously placed numbers.
    VI. Additional tools for solving the puzzle, outlined in objective 3.

Objective 3 – Development of tools, for the 'play' interface, that can assist the user in completing the Sudoku puzzles.

- Development of a tool that allows the user to place values in the corner of squares – this allows the user to note the possible solutions for a square.

- Development of a saving system that can save the current puzzle, granting the user to option revert back to the saved state at any time, allowing the use a trial and error technique.
- Development of a tool to choose squares to be solved automatically, or to solve the whole puzzle automatically.
- Development of an undo button, that can undo as many moves as the user wants.
- Development of an option to show whether or not entered values are correct.

3 Background

Sudoku puzzles first appeared in 1979 in “Dell Pencil Puzzles & Word Games” (although at the time the puzzles were called “Number Place”), but didn’t became popular in the UK until 2004, the year in which a piece of software created by Wayne Gould, simply called “Sudoku”, was published. This piece of software allowed the user to choose from 5 different difficulty levels and provided basic tools for solving the puzzles. Gould eventually succeeded in convincing a newspaper to print his puzzles, and then the popularity exploded.

Despite being published daily in many newspapers across the UK, with some even displaying these daily Sudoku puzzles online, there is no standard way for categorizing the difficulty of Sudoku puzzles. *The Daily Mail* categorizes puzzles with a star rating from 1-5 stars, *The Guardian* categorizes puzzles into three difficulties of easy, medium and hard, and *The Daily Telegraph* categorizes puzzles into four difficulties of gentle, moderate, tough and diabolical. This project will attempt to create a difficulty system using algorithms based on techniques that can be used to solve the puzzles.

There are many programs already on the market for playing Sudoku puzzles, including downloadable software, flash games and apps on the Apple App Store. Therefore by focusing on what separates the difficulty levels, this project aims to create an enhanced piece of software for generating and solving Sudoku puzzles.

It is difficult to unanimously categorize Sudoku puzzles into difficulty levels, as the relative difficulty for the player depends on the techniques that the player is familiar with. Some players may find some puzzles that have been categorized as ‘hard’ easier than ones categorized as ‘medium’, as they have developed the techniques necessary to solve the supposedly harder puzzle. This poses a problem; should the difficulty level of a Sudoku puzzle be solely defined by the difficulty of the techniques required to solve the puzzle, or should it also include the estimated amount of time it would take, and the amount of techniques used?

In total, there are 6.671×10^{21} valid Sudoku grids (Bertram Felgenhauer, 2006), and a Sudoku puzzle must have a minimum number of 17 givens for it to be potentially valid (McGuire, 2012).

Solving Techniques

This section will outline a variety of techniques for solving Sudokus, beginning with the most basic.

Single Possibilities

The most basic technique for solving Sudokus is finding squares where there is only one possible number that can be positioned in that square. For example, in fig. 3-1, the sub-square at the top left of the grid contains eight numbers, and therefore the highlighted empty square must be filled with the number that is not currently present, which is the number 1.

2	9	6						5
5	8	7		9			6	
	3	4			6		7	
6			7	3			4	2
			9			8		
	4		6					
3			4	5				
	6			7	3	1		8
		2	8		1			4

Fig 3-1

Lone Rangers

A slightly more complicated method is finding so called ‘lone rangers’. These are numbers that can only be placed in one square in a sub-square, row or column, determined by elimination of all other possibilities. For example, in fig 3-2, the center sub-square does not yet contain the number 2, so therefore it must be placed in one of the four empty squares. By observing that the number 2 already appears in columns 5 and 6, it can be determined that 2 must be placed in the square that is highlighted green, as it is the only remaining possible square for the number 2 in the center sub-square. Single possibilities and lone rangers are the only techniques required for solving the easier Sudoku puzzles, however, more difficult puzzles will be impossible to solve without the use of more complicated methods.

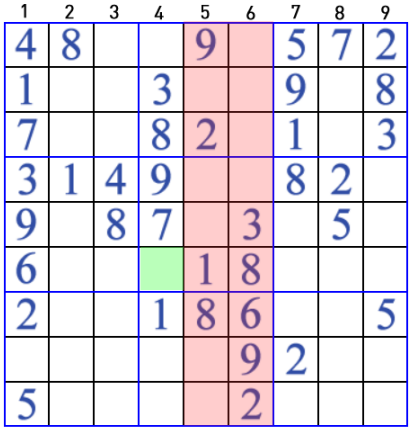


Fig 3-2

Twins

Twins are defined by having two squares that have the exact same two possibilities. Observe fig. 3-3, in which all the possible numbers that can be placed in the empty squares are noted in them. In particular, the two squares (5, 2) and (6, 2) have only the exact same two possibilities, which means that either (5, 2) is 4 and (6, 2) is 5, or (5, 2) is 5 and (6, 2) is 4.

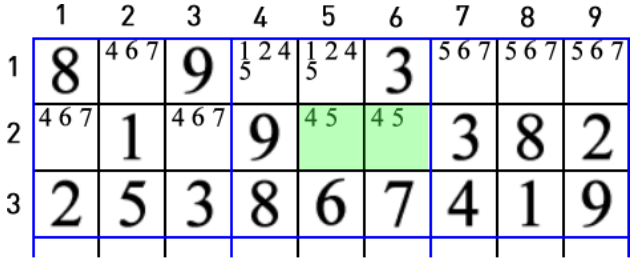


Fig 3-3

Because of this, it can be determined that the other empty squares in the sub-square, row and column cannot contain the numbers 4 or 5. Fig. 3-4 shows an updated grid reflecting these changes.

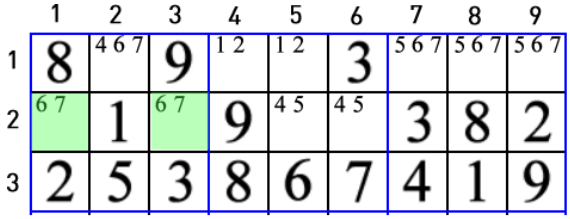


Fig 3-4

Note that squares (2, 1) and (2, 3) can no longer contain the number 4, which means that by using the lone ranger method, the square (1, 2) can now be confirmed to contain the number 4. Fig. 3-5 shows the updated grid reflecting this.

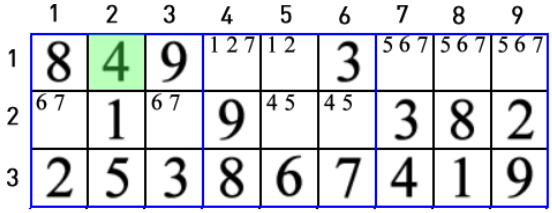


Fig 3-5

Triplets

Triplets are very similar to twins, but with some distinctive differences. Similarly to the method as used in the 'twins' technique, if there are 3 squares that contain the same 3 possibilities, you can eliminate these possibilities from the rest of the empty squares in that sub-square, row and column. For example, the sub-square shown in fig. 3-6 displays 3 squares that have exactly the same 3 possibilities, which means that these numbers can be eliminated from the other squares, as shown in fig. 3-7.

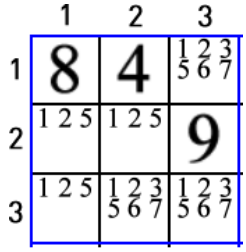


Fig 3-6

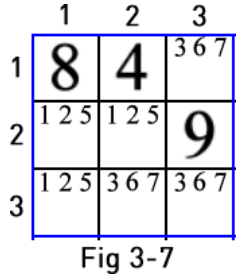


Fig 3-7

However, as previously mentioned, there are some differences between the twins and triplets techniques. For example triplets can be found even when the possibilities for the 3 squares don't exactly match. In fig. 3-8, consider the 3 squares that are highlighted green. Although they don't all have exactly the same possibilities, they can in fact be considered triplets. This is because, out of the possibilities, whichever number the square (3, 1) contains, the squares (1, 1) and (2, 1) become twins. If (3, 1) contains the number 1, then all the other squares in the sub-square have the number 1 eliminated from their possibilities, and the same can be said if (3, 1) contains the number 3. In each instance, the squares (1, 1) and (2, 1) are left with the possibilities 3 and 5 or the possibilities 1 and 5. This means that the numbers 1, 3 and 5 must be positioned somewhere in the squares (1, 1), (2, 1) and (3,1), and can therefore be eliminated as possibilities from the other empty squares, as shown in fig. 3-9.

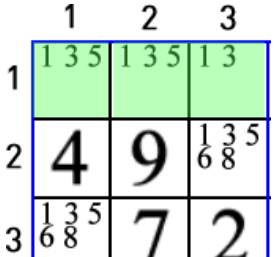


Fig 3-8

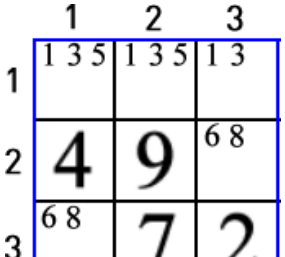


Fig 3-9

X-Wings

X-Wings is a much more complicated technique that can be harder to identify in a Sudoku puzzle. It consists of 4 squares lining up to make two diagonals that are perpendicular to each other, essentially creating an X shape. For example, in fig. 3-10, the green highlighted squares form an X-Wing. These 4 squares must all share one possibility, which in this case is the number 4. This creates so-called ‘locked’ pairs of 4’s, whereby across the diagonals one of the two squares must contain the number 4, as there is no other possibility across the rows. This means that in either (1, 2) or (1, 5) must be the number 4, so 4 can be eliminated as possibility of the other squares in that column. The result of this is shown in fig. 3-11.

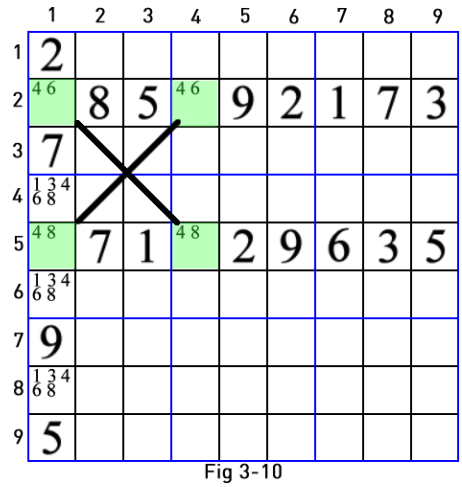


Fig 3-10

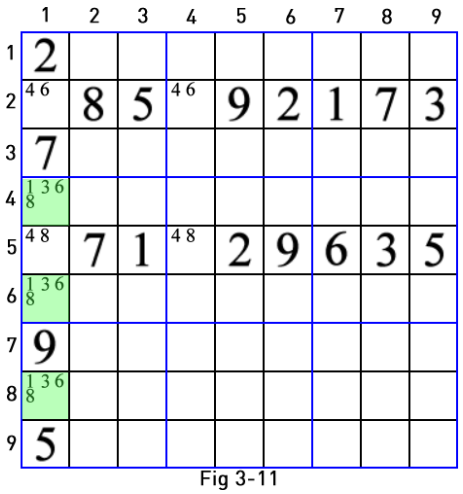


Fig 3-11

4 Technical Development

Solving Algorithms

This section will outline the algorithms used in the software for the implementation of generating and solving Sudoku grids.

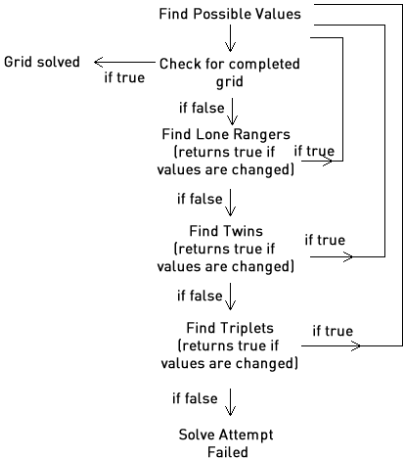
Basic Brute Force

At the very start of the project, an incredibly basic brute force algorithm was used to solve and create valid completed grids. The algorithm works by starting at the top left hand square of the grid travelling across the rows and then down a column until an empty square is found. The square is assigned a number, starting from 1, and then the grid checks if it is valid by checking for any duplicate numbers in sub-squares, rows or columns. If the grid is valid then the method recursively calls itself to find the next empty square. If at any point the grid is invalid then it returns false and therefore backtracks to a previous empty square and tries a different number.

The problem with this algorithm is that it always generates the same completed grid if given an empty grid, and also is inefficient in the sense that it might backtrack and then try a number in a square that it has already tried.

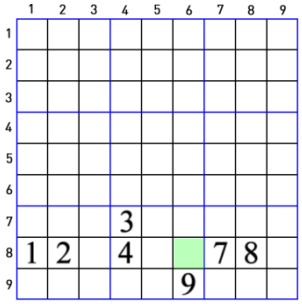
Constraint Propagation

Constraint propagation is an algorithm technique that is highly relevant when considering Sudoku puzzles. The algorithm follows that when a given variable is assigned a value, either directly by the user or by the system, the algorithm recalculates the possible value sets and assigned values of all its dependent variables. (Ambite, 2001). For example, in a Sudoku grid if one value changes then the values for all the other squares are recalculated to reflect this change. This technique is used throughout the algorithms that this piece of software uses to solve Sudoku puzzles, as when a method finds changes in the possibilities for a square, the “Find Possible Values” algorithm is then called. The diagram below shows the process used for attempting to solve puzzles.



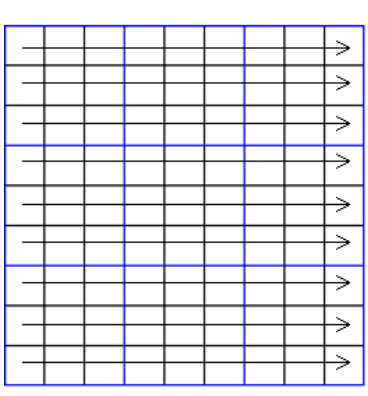
Find Possible Values

Contained within the solving class is a list that contains the possible numbers that can be contained within each square. When this algorithm is passed a coordinate for a square, it goes through each other square in the given square's sub-square, column and row and removes from the possibility list any number which is already confirmed to be in another square in that sub-square, row or column. For example, in the image below, if given the green highlighted square, it's possibilities would be set to "5,6" as the numbers 1 through 4 and 7 through 9 are either in the highlighted squares sub-square, row or column.



Find Lone Rangers

After the above mentioned "Find Possible Values" algorithm has been executed, the "Find Lone Rangers" algorithm can be used. This algorithm searches through each sub-square, column and row searching for numbers that only occur as a possibility in one square. The below diagram demonstrates how the algorithm works when searching through rows.



1. Starts at the top left and scans left to right

2. Checks each row for each number, if the number is found then a counter is increased

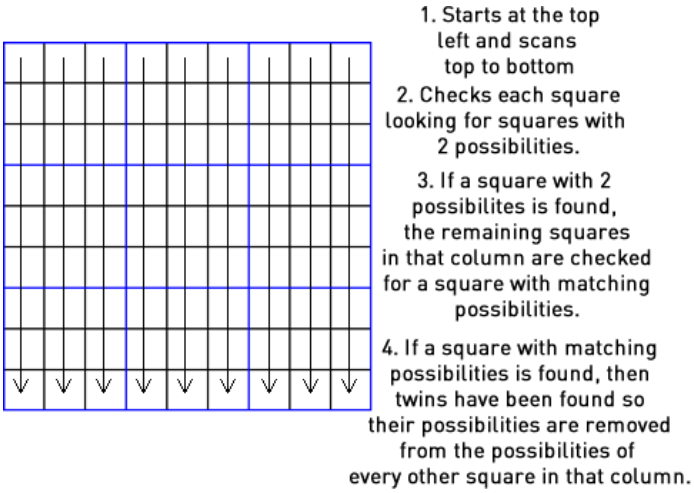
3. If after checking each square in the row the counter is at 1, then there is only one possible square for that number.

Note: Only checks squares that dont have confirmed values

Find Twins & Find Triplets

The "Find Twins" algorithm is used if the "Find Lone Rangers" algorithm fails to change the possibilities of any squares. This algorithm searches through each sub-square, row and column and finds any two squares that contain the same two possible values, and then eliminates these possibilities from the rest of the squares in that sub-square, row or column.

The below diagram demonstrates how this algorithm works when searching through columns.



The find triplets algorithm works in exactly the same way as the twins algorithm, except from that it looks for squares with 3 possibilities, and then looks for 2 other matching squares in the same sub-square, row or column.

Generating New Grids

When the method for generating a new grid is called with a difficulty level, the software randomly generates a completed grid by placing random numbers in squares starting from the top left, and then checking the validity of the puzzle after each number placed. If the puzzle is found to be invalid at any point, it backtracks and changes a number. This algorithm has the potential to generate any possible completed Sudoku puzzle.

Some numbers are then removed in random positions and then it attempts to solve the puzzle using the constraint propagation method described above. Depending on the given difficulty level, it finds a grid that can only be solved by using certain algorithms. For example, if the user requests a hard difficulty level Sudoku, then the returned Sudoku must use the “Find Triplets” method at least once when attempting to solve it, otherwise it restarts the process of generating a new grid. For a basic level puzzle, only the “Find possible values” and “Find lone rangers” algorithms can be used, and it is also produced with more givens. An easy level puzzle is similar to a basic level puzzle, except it will have less givens. For a medium level puzzle, the “Find Twins” method must be used at least once.

Time Taken To Generate New Grids

The table below shows the results of how long it takes to generate the Sudoku puzzles at different difficulty levels. This is calculated by generating 500 Sudokus of each difficulty level, logging the time for each and then calculating the standard deviation and mean.

Difficulty level	Mean time (seconds)	Stand deviation (seconds)
Basic	0.00946	0.00727
Easy	0.0143	0.0164
Medium	0.941	0.823
Hard	5.325	3.869

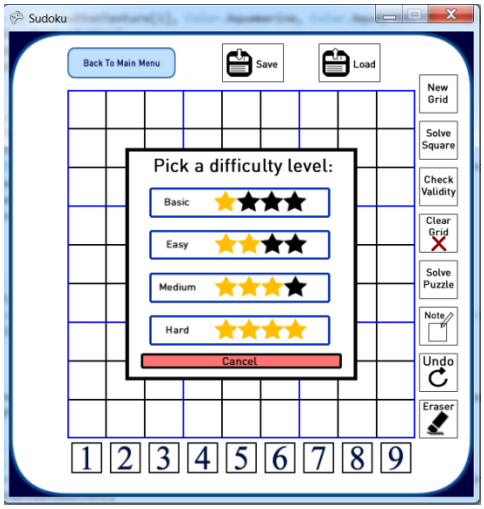
System Design

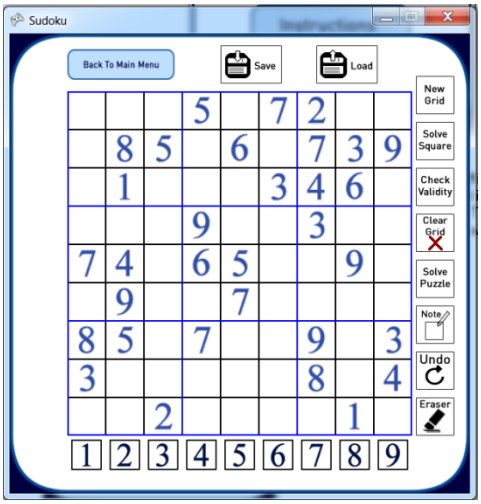
The class design of the software is simple, with classes for buttons, saving & loading, solving and a foremost class called Sudoku. The button class is a simple class for creating button objects which can then be used to get the state of the button, which can be either ‘up’, ‘clicked’ or ‘hover’, which can be used to determine when to perform specific actions. The saving and loading class handles the loading and saving of grids and the conversion of them from the string format into 2d arrays. The solving class has methods for all the different solving techniques that are utilized and is either used to create a new grid with a specified difficulty, or to solve a grid when given a 2d array containing the grid starting numbers (givens). The Sudoku class handles the UI display through the ‘Draw’ method and calls ‘Update’ to handle user actions, such as a user entering a number into the grid or clicking a button.

UI Design

The UI has been designed to be simplistic, with focus on ease of use, recognizing that Sudoku players may not necessarily be technologically literate. The main menu, as shown below, has been designed to be extremely simple, with just four buttons labeled challenge mode, free play, instructions and exit game.

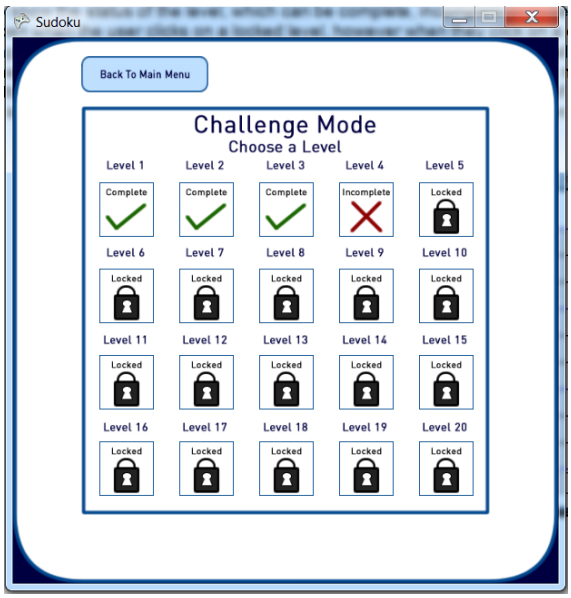
When entering free play mode, the user is prompted to select a difficulty from the four levels of difficulty (basic, easy, medium and hard). When a difficulty level is chosen, a grid of that difficulty level is generated and the starting grid is then displayed. The user also has the option of choosing to not pick a difficulty level, at which point they will be given a blank grid. In free play mode, the number buttons are displayed along the bottom of the grid, the tools are on the right side of the grid and the save and load buttons and at the top. The positioning of the buttons has been designed in such a way that the buttons that the user would utilize the most, during an attempt to solve a Sudoku puzzle, are positioned together – the note, undo and eraser buttons are all positioned in the bottom right, closest to the number buttons.

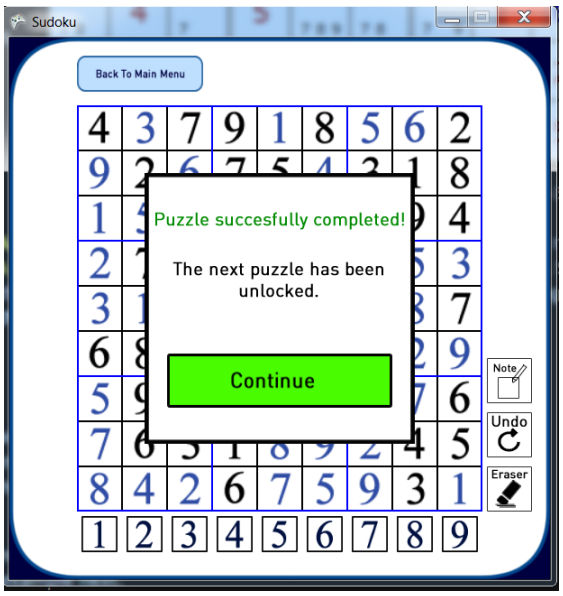




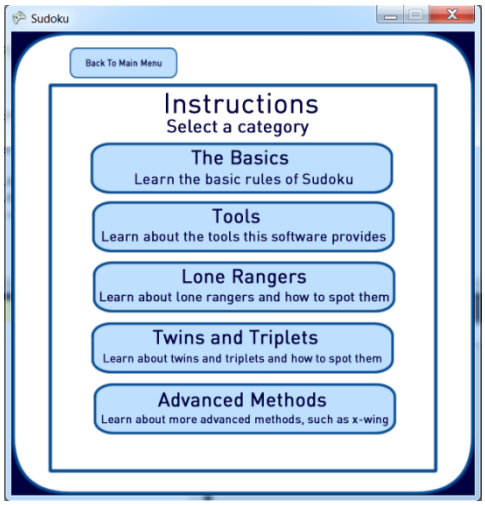
At the top are the buttons that the user would likely use less often – the save, load and new grid buttons. The buttons are designed to be very basic, simply showing the user a small description of what the button does, along with a small picture in some cases.

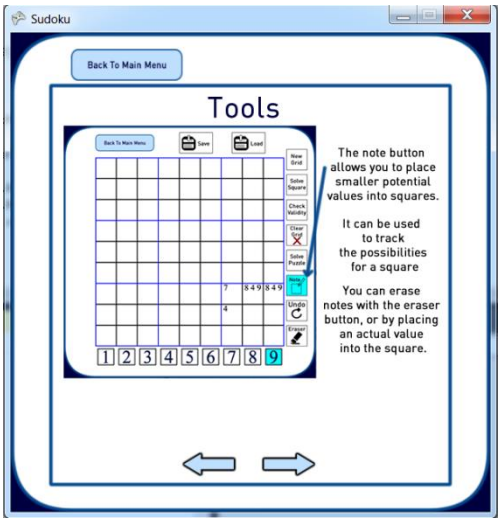
The challenge mode menu displays the numbers of the levels available, along with a small image to show the status of the level, which can be complete, incomplete or locked. Nothing will happen when the user clicks on a locked level, however when they click on a complete or incomplete level, they will be given the starting grid for that level. When in challenge mode, the only buttons available for use are the number buttons and the eraser, note and undo buttons. When the puzzle is completed, a window pops up telling the user if their solution is valid, and if it is valid, that the next puzzle in challenge mode has been unlocked.





The instructions menu displays a list of topics that the user can choose to learn about, as shown below. When the user chooses a category, a walkthrough explanation about the chosen subject is displayed in a format similar to a slideshow. The category shown below is the ‘Tools’ category.





As shown, the UI has a predominantly white, black and blue colour scheme, with shades of red and green incorporated to indicate negative or positive actions. The overall goal this UI design is to not confuse the user, which is accomplished by using a basic colour scheme, simple to use buttons and basic terminology in descriptions.

Test Design

Due to how the software generates puzzles (by first randomly generating a starting grid and then attempting to solve the puzzle using techniques a player would), there is a high degree of certainty that the generated puzzles have a unique solution and are valid. Throughout the tests done to establish how long it takes to generate a puzzle, over 2000 puzzles were generated in different difficulty levels and not once did the generation fail to create a puzzle. Tests have also been conducted to ensure that there is no problems with the functionality of the tools – such as by changing game mode and checking that there is no overlap, and by checking that all the tools interact correctly with each other.

5 Evaluation

The produced software in its current form has met the vast majority of the original aims and objectives of this project. The first objective of:

"Objective 1 – Development of code to generate and solve Sudoku puzzles, including designing a difficulty system based on techniques that a person would use in solving Sudoku puzzles"

has for the most part been successfully met, however the difficulty system could have been more refined. The algorithms did not fully take into account the influence that the amount of givens has on the difficulty level, as this was not fully explored throughout the project. The only conclusion that could be made is that in general less givens means the puzzle is harder, but that it is more correlation than causation. The difficulty system does however take into account the techniques that a person is likely to use when solving a puzzle, as the puzzles generated that have an basic or easy difficulty level only require the most basic techniques to solve. The objective for being able to generate a Sudoku puzzle within 1 second was met when generating difficulties of basic, easy and medium, but was not met when generating hard difficulty puzzles which took a mean time of 5 seconds. Furthermore, this objective included having a set of puzzles that have already been generated included with the software, which was met with the inclusion of a challenge mode, which was an objective that was not originally considered, but is an addition that enhances the software.

The second objective of:

"Objective 2 – Development of an interactive interface for solving and creating Sudoku puzzles"

has been completely and successfully met. All the original goals for the interface have been implemented, including the main menu, the instructions menu, the ability to display the current puzzle with initial numbers highlighted and option to save and load puzzles.

The third objective of:

"Objective 3 – Development of tools, for the 'play' interface, that can assist the user in completing the Sudoku puzzles"

has been met as a result of the implementation of the following tools; the noting tool, the tool to solve squares automatically, the tool to solve the whole puzzle automatically, the undo tool and the tool to check if entered values are correct.

Overall, in terms of software functionality, all the original goals have been met. However, the software could still be improved by refining the differences between difficulty levels, particularly with further investigation into the effect of amount of givens, as well as the implementation of an algorithm for the x-wing technique

As an area of further potential development, the software would function well as an app as the simple button based interfaces would work well with a touch screen input method. The challenge mode would function well as a selling point for the app, with the possible introduction of a point system where you gain points for completing puzzles in challenge mode, which can then be used to automatically solve squares in other challenge mode puzzles.

6 Conclusion

The project is in a state where it can be considered that it accomplished its goals, despite the fact that it could still be built upon. The software is fully working and includes all of the functionality that was intended to be implemented, with no known bugs. For the purposes of this project, this software used to create it was well suited and functioned effectively.

For the user it provides a great tool for generating and completing Sudoku puzzles, with excellent support for users who have never played Sudoku before. The challenge mode provides an incentive for the user to improve upon their skills, and the instruction mode allows them to learn. The free play mode provides the user with all the functionality that they would want when attempting to solve a Sudoku puzzle.

Moving forward, the transition into turning the software into an app is a path that can be considered, with good potential for creating a successful commercial product. In terms of the algorithms used, the ones implemented fulfill the needs of the software, however there is potential for improvement especially in some of the algorithms that outline a generated puzzle's difficulty level.

Appendix A: Initial Task list

#	Task Name	Description	Duration (days)
1	**Initial report**	Write the initial report deliverable.	28
2	Basic play game interface	Design & develop a basic interface for displaying the current Sudoku puzzle.	14
3	Solving techniques research	Research into techniques used for solving Sudoku puzzles.	7
4	Code for generating completed Sudoku puzzles	Implementation of code that can generate random fully completed Sudoku puzzles that follow the rules of Sudokus. This will use a brute force method to solve Sudokus after generating random starting points.	7
5	Code for generating valid initial Sudoku puzzles	Implementation of code to create valid starting points of Sudokus (i.e. with givens), based on completed Sudoku puzzles from the previous task.	14
6	Algorithms for solving Sudokus based on player techniques	Implementation of code that solves Sudokus from the starting point using techniques that a player would use. It will attempt to use the most basic techniques first and then harder techniques.	21
7	Design difficulty level scale	Creation of a difficulty level scale based on techniques that need to be used to solve Sudoku puzzles.	7
8	**Interim report**	Write the interim report deliverable.	21
9	Algorithms to decide puzzle difficulty level.	Based on the solving techniques in task 6, and the difficulty level scale in task 7, design an algorithm to determine a puzzles difficulty level.	21
10	Implementation of basic tools for play menu	Implementation of basic tools includes; placing the numbers 1-9 in squares, eraser tool, checking if the completed puzzle is correct.	7
11	Implementation of noting tool	Implementation of the noting tool which allows the user to place numbers in the corner of squares.	4
12	Implementation of the undo tool	Implement the undo tool – allowing the player to undo previous moves up to a limit of 10.	3
13	Implementation of the ‘solve automatically’ tool	Implementation of tools for solving individual squares or the whole puzzle automatically.	4
14	Implementation of ‘check to see if square is correct’ tool	Implementation of a tool that allows the user to check if a square is correct.	3
15	Implementation of tool allowing players to enter their own Sudokus	Implementation of a tool that allows the user to enter their own Sudoku (i.e. they can enter the givens and then use the other tools as normal). This will also check that there is a valid solution.	7
16	Implementation of loading and saving	Implementation of a saving and loading system that can be used to save the game and return to it at a later time (after the program has been restarted), or to save the state of a Sudoku puzzle and return to it during the same session (for trial and error techniques).	21
17	Implementation of main menu	Implementation of a main menu which has buttons for “play game”, “instructions”, “options” and “quit”.	5
18	Implementation of instructions and options menus	Implementation of instructions menu which includes basic rules and solving techniques. Implementation of options menu which includes changing the colour scheme and rebinding keys.	5
19	Testing	Complete extensive testing of the software.	10
20	**Final report**	Write the final report deliverable.	28

Appendix B: Time Plan

		University Calendar Week – Semester 1																	
#	**Task Name**	**4** 21st Sep	**5** 28th Sep	**6** 5th Oct	**7** 12th Oct	**8** 19th Oct	**9** 26th Oct	**10** 2nd Nov	**11** 9th Nov	**12** 16th Nov	**13** 23rd Nov	**14** 30th Nov	**15** 7th Dec	**16** 14th Dec	**17** 21st Dec	**18** 28th Dec	**19** 4th Jan	**20** 11th Jan	**21** 18th Jan
1	Initial Report				D1														
2	Basic play game interface																		
3	Solving techniques Research																		
4	Code for generating completed Sudoku puzzles																		
5	Code for producing valid initial Sudoku puzzles										M1								
6	Algorithms for solving Sudokus based on player techniques																		
-	Christmas Week																		
7	Design difficulty level scale																		
8	Interim Report																		D2

Key

D – Deadline.
M – Milestone.

D1 - Initial report, 15th October.

M1 – This milestone marks the point where the program can produce valid Sudoku puzzles.

D2 - Interim report, 21st January.

	University Calendar Week – Semester 2														
#	**Task Name**	**22** 25th Jan	**23** 1st Feb	**24** 8th Feb	**25** 15th Feb	**26** 22nd Feb	**27** 29th Feb	**28** 7th Mar	**29** 14th Mar	**30** 21st Mar	**31** 28th Mar	**32** 4th Apr	**33** 11th Apr	**34** 18th Apr	**35** 25th Apr	**36** 2nd May
9	Algorithms for solving Sudokus based on player techniques															
10	Algorithms to decide puzzle difficulty level															
11	Code for producing valid initial Sudoku puzzles								**M2**							
12	Implementation of 'solve automatically' tool															
13	Implementation of 'check to see if square is correct' tool															
14	Implementation of tool allowing players to enter their own Sudokus										**M3**					
15	Implementation of instructions and options menus															
16	Testing															
17	Final report deliverable															**D3**

Key

D – Deadline.
M – Milestone.

M2 – This milestone marks the point where Sudokus can be generated and categorized by difficulty level.

M3 – This milestone marks the point where all the tools have been added and the 'play' screen is fully functional.

D3 – Final Report, 5th May.

Appendix C: Risk Analysis

Risk	Severity (L/M/H)	Likelihood (L/M/H)	Significance (Sev. x Like.)	How to Avoid	How to Recover
Data loss	H	M	HM	Keep Backups, use SVN	Reinstate from backups or SVN repository
Loss of backups	H	L	HL	Multiple Backups	Use alternate backup
Other deadlines	H	H	HH	Good time management & planning.	Create new time plan, change priorities
Bad time management	H	M	HM	Consistent time plan evaluations	Create new time plan
Illness	M	L	ML	Generally unavoidable	Make sure to stick to time plan to minimize time loss.
Insignificant knowledge	H	L	HL	Research into task before starting it	Further research into area and adjust time plan to adapt to this
Software errors	H	L	HL	Intensive testing	Find bug/fault & fix
Hardware failure	L	L	LL	Use within recommended limits	Use alternate hardware

References

Ambite, J.-L., 2001. *Constraint Propagation.* [Online]
Available at: http://www.isi.edu/info-agents/papers/www10/node9.html
[Accessed 10 April 2016].

Bertram Felgenhauer, F. J., 2006. [Online]
Available at: http://www.afjarvis.staff.shef.ac.uk/sudoku/felgenhauer_jarvis_spec1.pdf
[Accessed 13th October 2015].

Daily Mail, The Mail on Sunday & Metro Media Group, 2015. *Sudoku | Puzzles | Daily Mail Online.* [Online]
Available at: http://www.dailymail.co.uk/coffeebreak/puzzles/sudoku.html
[Accessed 14th October 2015].

Gould, W., 2007. *Sudoku - Download.* [Online]
Available at: http://www.waynegouldpuzzles.com/sudoku/download/
[Accessed 5th October 2015].

Guardian News And Media Limited, 2015. *Sudoku | Life and Style | The Guardian.* [Online]
Available at: http://www.theguardian.com/lifeandstyle/series/sudoku
[Accessed 14th October 2015].

Hayes, B., 2006. *Unwed Numbers >> American Scientist.* [Online]
Available at: http://www.americanscientist.org/issues/pub/2006/1/unwed-numbers/2
[Accessed 4th October 2015].

Lee, W.-M., 2006. *Programming Sudoku.* New York: Apress.

McGuire, G., 2012. [Online]
Available at: http://www.math.ie/McGuire_V1.pdf
[Accessed 14th October 2015].

Norvig, P., n.d. *Solving Every Sudoku Puzzle.* [Online]
Available at: http://norvig.com/sudoku.html
[Accessed 12 March 2016].

Rosenhouse, J. & Taalman, L., 2011. *Taking Sudoku Seriously.* New York: Oxford University Press.

Stone, K., 2016. *BrainBashers: Sudoku: X-Wing.* [Online]
Available at: https://www.brainbashers.com/sudokuxwing.asp
[Accessed 10 April 2016].

Telegraph Media Group Limited, 2015. *Number Puzzles - Telegraph Puzzles.* [Online]
Available at: http://puzzles.telegraph.co.uk/site/number_puzzles
[Accessed 14th October 2015].