Gamesman: A Graphical Game Analysis System

Dan Garcia <ddgarcia@cs.berkeley.edu>

Abstract

We present **Gamesman**, a graphical system for implementing, learning, analyzing and playing small finite two-person complete-information games. This is not to be confused with David Wolfe's excellent *Gamesman's Toolkit* [5], whose primary role was to understand the combinatorial game theoretic values of games and manipulate them. In contrast, **Gamesman** allows you to play the game graphically, analyze strategies, under the limiting assumptions that the players alternate turns and that the game under study can be completely solved. It is written in C and Tcl, and is freely available for Macintosh, Windows, and Unix systems.

1 Introduction

Gamesman began as an independent summer project in 1990 [1] to build a system that would strongly solve (by brute force) two-person complete-information small games and allow interactive play against a user. From its inception, it was assumed that the players moved alternately, and that a *position* was a game state *and* the player whose turn it was to move[2]. This assumption simplified game position values to one of three options: winning, losing or tieing (drawing, in the case of loopy games).

The beauty of knowing the entire game tree is that we can create a perfect opponent who never loses given a winning position, and immediately takes advantage of a fatal losing move. This is a move which hands your winning position over to your opponent. **Gamesman** will allow you to play a game against this perfect opponent (giving you the opportunity to move first or second), or let you play against another human player. In the latter situation, it acts as a "referee", constraining the players' moves to be legal.

The interaction throughout the game is completely graphical; you click to make moves. All moves are represented by either an arrow in the case of a piece-sliding move, a dot representing a piece placement, or a line representing piece removal. This technique allows us to color-code these moves to encode their value, as described in the following section.

[1] Release 1.0 was completed and published in [3].

[2] Just as a *position* in chess would be captured by a picture of the board position and the phrase "white to move".

Figure 1: The **Gamesman** splash screen.

The look and feel to the user is consistent from game to game. This is because we make use of common interface libraries to implement each module. In addition, we made attempts to parameterize each game, allowing the user to change the rules to see how that affects strategy. For example, with each game we offer the misère rules, in which what was normally a winning condition is now a losing condition.

Finally, **Gamesman** is extensible. We provide all the source code with the distribution, and encourage other programmers to write game modules under the *GNU General Public License*; only a knowledge of C and Tcl/Tk is required.

2 Value moves

One major contribution of this system is the idea of *value moves*, or the graphical color-coding of moves based on whether they win (i.e., maintain the winning advantage), lose (i.e., are fatal moves, giving advantage to the opponent) or tie (i.e., hand the opponent a tie position) for the player whose turn it is as in Figure 2. We choose the colors to represent a traffic light, so that win / tie / lose maps to go / cau-

Figure 2: The *value moves* legend, color-coded to mnemonically represent a traffic signal. Losing moves are colored in dark red (i.e., "stop!"), tieing moves in yellow (i.e., "caution") and winning moves in green (i.e., "go!").

tion / stop. This feature allows the player deciding on a move to quickly analyze all possible options and deduce a strategy graphically. Figures 3 through 6 illustrate this for the four game modules shipping with **Gamesman**. We also provide nim-values for any impartial games (specifically, *1, 2, ..., 10* and *Tac Tix*).

3 Implemented Game Modules

There are currently four game *modules* implemented in **Gamesman**: *1, 2, ..., 10*, *Tic Tac Toe*, *Tac Tix*, and *Dodgem*.

3.1 1, 2, ..., 10

This is a simple nim variant [2] used for teaching the system to children, and is called "one line nim" in [1]. Play involves moving your counter either one or two spots above the previously placed counter. The first person to move their counter to the spot labeled "10" wins. Although this is an impartial game, we graphically illustrate this as left and right taking turns placing their blue or red coin on the board, since this provides an nice visual move history. The available rule variant is the misère version, wherein reaching 10 first *loses*.

Figure 3: A *1, 2, ..., 10* position with Right to play and value moves turned on. The existence of a winning move (option) indicates this game is a win for the player whose turn it is, Right. Since Left has moved to the the 8th slot, Right can win by choosing to place his coin two spaces up onto 10, or lose by placing it one space up onto 9.

3.2 Tic Tac Toe

In this classic elementary school game, players alternate placing an $\times$ or $\circ$ on a 3 by 3 board, and the first to achieve three in a row of their piece (horizontally, vertically or diagonally) wins. If the board fills and no one has done so, the game is considered a tie [2]. The available rule variant is the misère version, wherein achieving three in a row of your own

Figure 4: A *Tic Tac Toe* position with $\times$ to move and value moves turned on. The existence of a winning move (option) indicates this game is a win for $\times$, and sure enough, if $\times$ moves in the lower-left, she wins. If $\times$ moves in the lower-middle, she ties (since $\circ$ will block in the lower-left). If $\times$ moves in the lower-right, she loses since $\circ$ will move in the lower-center.

Figure 5: A *Tac tix* position with value moves turned on. The existence of a winning move (option) indicates this game is a $\mathcal{N}$-position. The two winning moves are to remove two vertical stones in the middle or bottom of the third column.

pieces *loses*.

3.3 Tac tix

Invented by Piet Hein and described in [4], this is a simple impartial game. Each player takes as many pieces from any row or column as long as the pieces are *contiguous* (i.e., there is no gap between them). The player who removes the last piece is the winner. Simple m by n full rectangles are trivially played with the *tweedledum and tweedledee* strategy (after taking the central row or column if either m or n is odd), so most games are played with random starting configurations. In our implementation, the user may choose any possible starting position within a 4 by 4 grid. The available rule variant is the misère version, wherein removing the last piece *loses*.

3.4 Dodgem

The rules are taken from [2], p. 685:

> Colin Vout invented this excellent little game with two red cars and two blue cars on a 3x3 board, starting as shown in Figure 6. The players alternately move their cars one square in one of the three permitted directions (E, N or S for blue; N, E or W for red) and the first player to get both his cars off the board wins. Blue's cars may only leave the board across its right edge and Red cars only leave across the top edge. Only one car is permitted on a square, and you lose if you prevent your opponent from moving.

Figure 6: The *Dodgem* initial position with Left to move and value moves turned on. The existence of a winning move (option) indicates this game is a win the player whose turn it is, Left. If Left slides her top coin to the right, she maintains her winning edge. If, however, she slides her lower coin in either direction, she loses (against a perfect opponent).

Dodgem is the only game module which is *loopy*, an interesting twist which required special programming care. Open positions are considered the same as tieing positions, and moves from $\mathcal{N}$-positions to open positions are labeled (color-coded) as tie moves. There are two available rule variants. The first is the misère endgame rule, wherein moving both your cars off the board first *loses*, the second is the misère block rule, wherein you win if you prevent your opponent from moving.

4 Results

There are quite a few results we have concluded through analysis of these four games using **Gamesman**. The most obvious is that we have solved these four games and know the values of both normal and misère forms, which we summarize in Table 1.

Game	Normal rules (good move)	Misère rules (good move)
1, 2, ..., 10	Win (1)	Lose
Tic Tac Toe	Tie (all)	Tie (center)
Tac Tix	depends	depends
Dodgem	Win (top ahead)	Lose

Table 1: The results we have ascertained about the value of these games (specifically, their initial positions) after strongly solving them.

4.1 1, 2, ..., 10

This trivial game has $\mathcal{P}$-positions at 1, 4, 7 and 10 for the normal game and at 0, 3, 6 and 9 for the misère game. Thus the entries in Table 1 indicate that the initial position is winning with 1 being a good move for the normal game and the initial position is losing for the misère version.

4.2 Tic Tac Toe

This game has been solved before, but we confirm that it is a tie game for both the normal and misère rules. Any first move preserves the tie for the normal game, but only the counterintuitive [3] move to the center (with tweedledum and tweedledee followup strategies) as the *only* initial move to guarantee the tie.

We also created a graphical representation of the game tree shown in Figure 7. This can be thought of as a fast-access answer-key. These are also available for several recursion levels on the main **Gamesman** web site.

4.3 Tac Tix

As we already discussed, full m by n boards are trivial. If either m or n is odd, it is a $\mathcal{N}$-position, with the winning move to divide the board into two equal sections. Otherwise, the full m by n board is a $\mathcal{P}$-position. Most games are played on a board consisting of a random scattering of pieces, so the initial position varies.

4.4 Dodgem

Dodgem is the most interesting game of the four. It was fully analyzed in [2] and our results correlate with

[3] This move seems illogical since it has the most three-in-a-row crossings running through it, so it would seem we should avoid it at all costs.

Figure 7: A recursive, graphical representation of the *Tic Tac Toe* game tree for the first three moves. The outer rim is the value of the position (in this case, a tie), and each recursive sub-square is the value the move for that person. For example, the × move to the upper-right is a tie, because the upper-right hollow square (imagine the board being broken up into 3 by 3 hollow squares) is yellow. Recursively, within that upper-right square there are 8 other hollow sub-squares (7 red = losing, 1 central yellow = tieing), and one black filled square (representing the illegal move of placing the ∘ directly onto the × already on the board). ∘ now can choose among any of those 8 moves, let's say it chooses to place its piece in the lower left of the board – this is a hollow red square, thus a losing move. This means × has at least one winning response, and we see from the diagram that it has two, one in the upper-left or lower-right.

their table on page 686. Misère dodgem is fascinating. It seems impossible to be forced off the board, but with the normal blocking rule in effect (i.e., it's illegal to trap someone), it is a losing (and not open) position.

5 Conclusion

We have presented **Gamesman**, a system for playing and analyzing small brute-force-solvable two-player games. It ships with four game modules, and provides a perfect opponent to play against as well as a way to modify the rules and teach yourself the strategy with *value moves*. We're sure you will enjoy it. Download away!

6 Availability

Gamesman is free, and available for the Macintosh, PC or Unix via http://www.cs.berkeley.edu/∼ddgarcia/software/gamesman/. It is distributed under the *GNU General Public License*.

References

[1] Gyles Brandreth. *The World's Best Indoor Games*. Pantheon Books, 1981.

[2] J. H. Conway E. R. Berlekamp and R. K. Guy. *Winning Ways for Your Mathematical Plays*. Academic Press, London, 1982.

[3] Daniel D. Garcia. Gamesman: A finite, two-person, perfect-information game generator. Master's thesis, Department of Computer Science, University of California at Berkeley, Berkeley, CA, May 1995.

[4] Martin Gardner. *The Scientific American book of Mathematical Puzzles and Diversions*. Simon and Shuster, 1959.

[5] David Wolfe. The gamesman's toolkit. In Richard Nowakowski, editor, *Games of No Chance*, pages 93 – 98. Cambridge University Press, London, 1982.

Alongside the Game Report, we also have the analysis function that may be familiar to many users! There are two ways to get there. Danny Rensch explains how to use analysis in this video! Or read more below!Â If you do not have automatic analysis enabled, your button will look like this: Or, if you are already looking at the game report, you can click on the â€˜analysisâ€™ tab, on the top next to the â€˜reportâ€™ tab. This will open the engine analysis of each move, and if you check the â€˜show linesâ€™ box it will show the top lines for each move: Click on the settings gear icon in the top right to adjust how the engine analyses the game: Engine time limit: How long should the engine spend evaluating the best moves? Set it from between 5 seconds and five minutes, or unlimited. The software framework we use is GAMESMAN (Game-independent Automatic Move-tree Exhaustive Search, Ma-nipulation And Navigation). At a high level, it is a system for the solving, playing, and analysis of finite, two-person, perfect information games. Since we wish to strongly solve these games, i.e., calculate values for every position (Allis 1994), the system employs traditional techniques of ex-haustive search and retrograde analysis. We wish to pro-duce reasonable results for games we cannot fully solve, so it also supports static evaluation. Although GAMESMAN is fundamentally grounded in p... Comisat-Games.sf.net >>Comisat Games Collection is an all-in-one free collection of games written in gambas for unix-like system, available in italian and in english. FedoraProject.org/wiki/Extras/SIGs/Games. en.OpenSuse.org/Wishlist_Games.Â At the core of the project is GAMESMAN, an open-source AI architecture developed for solving, playing, and analyzing two-person abstract strategy games (e.g., Tic-Tac-Toe or Chess).Â It is a complete system that includes all the components: a portable engine, graphical interfaces for Unix/Linux/X11, Macintosh, and Windows, multiple AIs, networking for multi-player games, and an extensive game library. IBDS.sf.net >>IBDS is a library for dynamic simulation of multi-body systems in C++. 7. Systems analysis tools and techniques: CASE tools Understand the need for Computer Aided Systems Engineering (CASE) tools in the work of a systems analyst. Understand the use of prototyping techniques in the reduction of a systemâ€™s development time.Â An example of a probabilistic system is a game of cards or the pricing system of an organisation. Open and Closed. When a system is isolated from its surrounding environment it is a closed system.Â (e) Graphical Notation Graphical language can be both simple and powerful in conveying information. Because of its pictorial nature, it can show boundaries, activities, data types, relationships and the number of elements within the boundaries.