200 Students Can't Be Wrong! GamesCrafters, a Computational Game Theory Undergraduate Research and Development Group

Yanpei Chen **Patricia C. Fong** **Jerry Hong**
Deepa Mahajan **Cynthia Okita** **David Eitan Poll**
Alan Roytman **Ofer Sadgat** **Daniel D. Garcia**

University of California, Berkeley
387 Soda Hall
Berkeley, CA 94720-1776
{ychen, patriciafong, jerryhong, dmahajan, ciokita, depoll, alanr, ofer, ddgarcia}@berkeley.edu

Abstract

The UC Berkeley *GamesCrafters* research and development group was formed in 2001 as a "watering hole" to gather and engage top undergraduates as they explore the fertile area of computational game theory. 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). The group is accessible to undergraduates at all levels. Those not yet ready to dive into code can create graphics, find bugs, or research the history of games for our website. Programmers can easily prototype a new game with multiple rule variants, design a fun interface, and perform extended analysis. Advanced students are encouraged to tinker with the software core, and optimize the solvers, databases, hash functions, networking, user experience, etc. Over the past six years, over two hundred undergraduates have implemented more than sixty-five games and several advanced software engineering projects. Since this is not a class, but *directed group study*, students can re-register as often as they like; most stay for two or three semesters. This allows for a real community to be formed with veterans providing continuity and mentoring as project leads, allowing for more ambitious multi-term projects. Our alumni have told us how valuable this experience has been for them, providing them with a nurturing environment to mature as researchers, developers and leaders. Their positive feedback, the overwhelming response we get when we demo at our annual campus open house, and the continued student enthusiasm tells us this could be one way to keep students excited and engaged by computer science.

1. Introduction

As computing educators, we face significant challenges. Our enrollment numbers are plummeting. The uninitiated believe our field is just programming. We also may see only a few stages of the graduation pipeline, so our favorite students come and go with no persistence. We ask ourselves: How can we address these problems? How can we find fun and engaging ways to contextualize material for our students? How can we cast the net wide enough to allow access at all levels? How can we build a nurturing environment to allow for student growth? Finally, with most research groups aimed at graduate students, how do we share our joy of computing research and development with *undergraduates*? GamesCrafters, our computational game theory undergraduate research and development group founded in 2001, has tried to hit all of these marks.

The focus for our group is "hunting big game"; i.e., implementing, brute-force solving and analyzing finite, two-person, perfect information games. While others have focused on large games that can take years to solve, like Checkers (van den Herik 2002) (Schaeffer 2007), we have tried to find an approach with more immediate results, i.e., smaller games solvable in less than a semester. Our GAMESMAN software infrastructure began in 1988 as a C program to solve Tic-Tac-Toe, and has evolved into 240K lines of open-source AI source code and 65 implemented games contributed by over *two hundred* alumni!

Figure 1. The spring 2007 UC Berkeley GamesCrafters.

In the following sections, we will explain the history and technical details of the GAMESMAN architecture, the way we run the GamesCrafters group, our outreach efforts, and conclude with results we have measured – through observations of group dynamics and alumni feedback.

How can the GamesCrafters experience be replicated? We welcome other game-theory-fluent faculty who may wish to contribute to our open-source project and serve as a "satellite" arm of our group. Also, interested faculty could apply the lessons from our group management experience, and form similar undergraduate-only projects around their own research and development interests. Finally, there may be a way to appeal to an even younger audience. Below we have rewritten our solver in 6 lines of Scheme and a simple game in 3, to show that we could even tap into the grade school Logo class demographic. Since students can be turned off by computing at an early age (AAUW 2000), this avenue could have the most impact on the landscape.

```
(define (solve P)     ;; GAMESMAN Scheme solver
 (or (primitive P)    ;; P = position, M = move
     (let ((values    ;; This let can be on 1 line
            (map (lambda (M) (solve (do-move P M)))
                 (generate-moves P))))
       (cond ((member 'lose values) 'win)
             ((member 'tie  values) 'tie)
             (else 'lose)))))

;; The game "1,2,...,10". Starting at 0, players
;; add 1 or 2 to running total. First to 10 wins.
(define (primitive P)    (if (>= P 10) 'lose #f))
(define (do-move P M)    (+ P M))
(define (generate-moves P) '(1 2))
```

2. GAMESMAN

The software framework we use is GAMESMAN (Game-independent Automatic Move-tree Exhaustive Search, Manipulation 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 exhaustive search and retrograde analysis. We wish to produce reasonable results for games we *cannot* fully solve, so it also supports static evaluation. Although GAMESMAN is fundamentally grounded in principles from AI, our research touches nearly every other branch of computer science, making it ideal for engaging students and helping them explore our entire discipline.

2.1. Overview

GAMESMAN started as a single C program to solve Tic-Tac-Toe in 1988, and evolved into a Master's project in 1995 with four games implemented: *1,2,...,10*, *Tic-Tac-Toe*, *Dodgem*, and *Tac Tix* (Garcia 1995). The system took in the description of a game (via its rules and starting position), solved it, and played it perfectly through either a command-line or graphical user interface.

The project laid fallow for six years until the formation of the GamesCrafters group in 2001. Since then, successive generations of students have *revolutionized* the original system. Today, there are over 65 games in the repository, most with graphical interfaces. The open-source code base now consists of over 2K lines of C++, 8K lines of Java, 80K lines of Tcl/Tk, and 155K lines of C. We utilize bug-tracking software and a wiki to help with our development coordination. The system compiles and runs on Macs, *nix, and Windows (via Cygwin). GAMESMAN has three main components: the *Core*, the *Graphical User Interface* (see Figure 2), and *Modules* (i.e., games).

The Core is any *game-independent* software not part of the graphical user interface. This includes code for hash functions, solvers, static evaluation, databases, networking, analysis, and a *command-line interface* (CLI). Most of the code in the core is written in C, and we are in the process of rewriting it in C++.

Our Graphical User Interface (GUI) code base is also fully game-independent, and provides the framework that drives our windowed application, as well as generic UI libraries for modules to simplify writing a GUI. All games have a CLI, and most also have a GUI. Our framework has gone through several major iterations since 2001, and there is continued activity to bring games coded with legacy APIs into compliance with our latest API. Most of this code is in Tcl/Tk, and we have a Java effort underway.

Modules encode games by defining the game's *rules*, namely its *starting position*, the *move generation*, and the *endgame conditions* (win, lose, or tie). Each module implements *variants* of the game, so a single module can represent multiple games whose rules are mostly shared. All modules implement the *Misère* variant, in which the winning and losing conditions are reversed. Other common variants involve changing the dimensions of the board, or the number and mobility of the pieces.

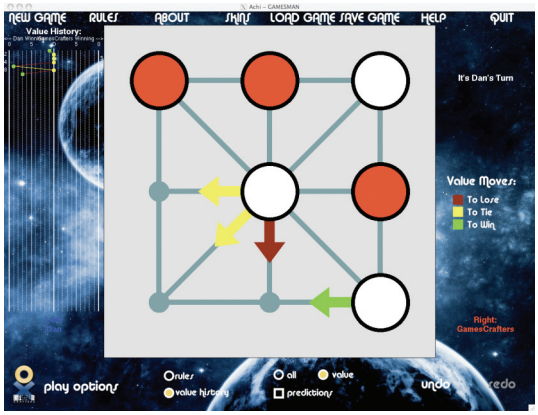

Figure 2. The Achi graphical user interface. The center frame is different for each game module; the outer frame and menus are shared by all games. A visualization of the value history is on the left, and moves are colored by value.

2.2. AI Core

GAMESMAN was initially conceived to solve games, play them perfectly, teach the user the perfect strategy, and allow basic analysis. It has evolved to provide static evalua-

tion for games that are too big to be solved, or for games that can be solved and compared with a particular strategy. AI principles are fundamental in our code for solving, static evaluation, and game analysis.

2.2.1. Solving Games. The process of strongly solving is essentially a minimax algorithm that exhaustively searches all the positions (nodes) in a game tree, with the leaves as *primitives* that evaluate to win, lose, or tie. We store the "answers" in a custom database for retrieval when playing.

The system provides a number of different solvers. The most basic uses a simple, depth-first traversal of the game tree. Another uses *retrograde analysis*, propagating values upward from the bottom of the tree in *tiers* (Gasser 1996). This lends itself nicely to redeployment as a distributed solver for compute clusters, one of our future projects.

2.2.2. Static Evaluation. We recently added the ability to statically evaluate a given game position. For some games, there exist perfect static evaluators (e.g., Nim), which obviate the need for any search. For other games with imperfect heuristics, what results is imperfect AI play. If we compare an AI's guess with that of a perfect opponent (by solving the game) and record where the AI made mistakes, we could potentially encode an extremely compact perfect database as "this imperfect solver plus these corrections". We could also engage simple machine learning to tune our evaluator.

2.2.3. Analysis. In addition to solving games, we gather statistics about the game tree, and introduce heuristics for things such as "interestingness" (for both positions and games). This post-solution analysis often reveals subtle and surprising information about the games, and opens doors to many other applications of AI techniques.

2.3. Beyond AI

GAMESMAN projects often touch many different fields of computer science. The GamesCrafters group acts as a focal point where students with diverse interests can come together and share their passion. Some current projects and their corresponding fields are listed below:

Theory: Combinatorially-perfect game hash functions, calculation and encoding of position symmetries.
Databases: Space-optimization for storing solutions, intelligent partitioning to improve performance.
Graphics and Visualization: Visualizations for game tree traversal, visualizations of post-solution analysis.
Human-Computer Interfaces: GUI, web interface for analysis, pen-and-paper to computer interface.
Operating Systems: Software virtual memory for large databases, Multi-threaded and distributed solvers.
Networking: Online player, databases & pairing system
Software Engineering: Open-source development & docs, OOP system redesign for extensibility.
Programming Languages: User interface description language , simplified game rules description language.

This broad coverage of computer science topics serves the diverse interests of our students well. Regardless of their specialty, there is *always* a GAMESMAN project to work on and students are exposed to various areas of computer science of which they may have been unaware.

2.4. Research Contribution

The group's emphasis has primarily been on software development, with research yielding code rather than papers. Our innovative work on efficient combinatorial hash functions, solvers and solutions of previously unsolved games has certainly been publication-worthy in combinatorics and recreational mathematical journals. From the start, students have been encouraged to write up their findings formally, but for various reasons none have taken advantage of the opportunity. As a result, no technical publications have come out of the group, but this is an avenue we will explore in the near future.

3. Undergraduate Participation

It is the outstanding, diverse, and enthusiastic students that make GamesCrafters unique. They work in teams, learn about game theory and the ups and downs of developing software, have a lot of fun, and form lasting friendships. This section details the human dimension of the group.

3.1. History of GamesCrafters

The group was founded in fall of 2001 with fourteen hand-picked ace undergraduates. Most of their projects focused on adding new games to the system. Over the years, the group has flourished and diversified. Approximately fifteen new students join each semester, enrollment hovers around forty, and sub-groups (with corresponding team leads) spread the management load. We are now one of the largest undergraduate research and development groups on campus.

3.2. Student Recruitment

In the early days of GamesCrafters, recruitment occurred solely on an individual basis – fun and engaging (mostly sophomore) students who stood out in class were hand-selected. Later, an open invitation was extended to the entire class, and students of all abilities and levels (freshman through senior) were encouraged to join. At the close of a semester, students with A+ grades received personalized solicitation via email. In recent years, word-of-mouth from veterans of the group has also been quite effective. Other students have learned about us through our highly visible open house activities (see Section 4.1.1).

The recruitment methods above create a diverse student base for GamesCrafters. Many students are attracted by the supportive team atmosphere, in which everyone gets equal say, regardless of experience or specialty.

3.3. Student Experience

In this section, we present a holistic picture of a student's GamesCrafters experience and how we have shaped it. We convey key lessons from operating a large student group: the need for student leadership within the group and the need for time dedicated to advanced students.

We have weekly meetings in the early evening that last approximately two hours. Some advanced students meet for another two hours to discuss specialized issues, often related to the architecture of the system. Students also meet with their team leads for about an hour per week.

Figure 3. Faculty advisor Dan Garcia (foreground) leads a typical round-table GamesCrafters meeting.

3.3.1. Semester Timeline. The table below illustrates a typical timeline for a 15-week semester.

Table 1. Typical GamesCrafters Semester Timeline

Week 1	Students pair up to play real board games. Ice-breakers, system overview, project brainstorm.
Week 2	Intro tutorials, project selection, teams chosen. Students volunteer as team leads. Photos taken
Week 3 Week 4	Ramp-up and extended tutorials. We code an actual game from start to finish in real time.
Week 5 ... Week 13	Students work on their projects, meet in teams and subgroups, and provide regular progress updates. If needed, further tutorials continue.
Week 14	Project presentations and feedback.
Week 15	End-of-semester party and photos taken.

Early tutorials give students a brief introduction to GAMESMAN and all the ways they can contribute. Students choose their projects based on their interests, are then split into teams, and team-leads provide further tutorials and work with students to set realistic timelines. As teams work on their projects, they meet weekly with their leads to discuss progress and resolve any concerns. They also periodically present their designs or unfinished projects to the entire group, who give feedback in a welcoming, critique-session environment. Toward the end of the semester, they present completed projects and receive in-depth suggestions. Students incorporate some or all of the suggestions before the final check-in. We always celebrate the close of a semester with a party and photos.

3.3.2. Range of Opportunities. Students have the flexibility to take on a variety of projects. Most first-timers join the *new/nice game* team to add game modules to the system or create GUIs for existing CLI-only games. There are also *retrofit*, *graphics* and *architects* teams, who update legacy APIs, and maintain the GUI and Core code bases respectively. Our *maximization* team focuses on updating game modules for advanced solver optimizations authored by our *odeepablue* team. Finally, we have a *docs* team dedicated to system-wide documentation and maintenance of our website and a *bugs* team who track and squash bugs. Many advanced students propose their own projects that often have a tremendous impact. In summary, there are projects for everyone, regardless of experience.

3.3.3. Student Growth. Through GamesCrafters, students develop their confidence and grow in professionalism and maturity. New students shift from feeling overwhelmed by the complexity of GAMESMAN to giving tutorials and tackling multi-semester projects. Over time, some students volunteer to become team-leads and others wrestle with challenging architectural projects. The nurturing and gentle coaxing we do at the early stages pays off when students return to the group semester after semester. What a joy it has been to see so many of our alumni start as shy wall-flowers, become team leads, then graduate with honors and join top graduate schools and industrial firms!

3.3.4. Community Building. For many students, the most memorable aspect of GamesCrafters is how much fun they have in our weekly meetings, in their work with fellow GamesCrafters, and in the honest, but friendly critique of each other's work. Within such an enjoyable environment, students meet fellow students who share their enthusiasm. The resulting partnerships and networking help build a community that accompanies students through their computer science coursework, in the workplace, in graduate school and beyond. In Section 5, we further discuss experiences of GamesCrafters alumni.

3.4. Enriching the Computer Science Curriculum

A network of capable peers is one of the many things that GamesCrafters find useful in their other computer science classes. Alumni have told us that their experience in GamesCrafters helped extend, enrich and ground material from their classes, including AI, databases, graphics, networks, architecture, and others. Several students have used their work in our group for projects in other classes. Conversely, students in these classes with an established interest in an area often sign up to further their knowledge. One of our recent alumni may have said it best: "The GamesCrafters projects are motivated by the students, and in turn, the students are motivated by our projects."

3.5. Springboard to Further Opportunities

For many students, the group is their first taste of computer science research and development. In addition to the actual open-source coding experience, students learn about the software engineering life cycle: design, prototyping, im-

plementation, testing, documentation, debugging, refactoring, and presentation. The teamwork setting, timeline development and progress updates mimic realistic work environments in the industry, and are looked upon favorably by employers. At the same time, students learn to formulate problems, define parameters, and evaluate and present results – skills invaluable in graduate school. Indeed, many students have used their experience in GamesCrafters as a springboard to opportunities in other research groups, industrial jobs and graduate schools.

4. Outreach

4.1. Educating the Public

We are always looking for ways to share our software, results and experiences with the public. We've done this through the face-to-face demos we've given during Campus Open House Day and our external website (`http://gamescrafters.berkeley.edu`). The broad goals of these two initiatives are to spark and cultivate interest in computer science amongst the general public and to provide a means of showcasing our project.

4.1.1. Campus Open House Day. *Cal Day* is the annual campus open house day held every April. It is a great way to present our work and to get visitors excited about computer science. GamesCrafters has consistently been a top attraction in the EECS department and we typically have several hundred people of all ages visit us to learn about our project (see Figure 4). We display twenty laptops around the room, each with a different game loaded.

Figure 4. Images from Campus Open House Day.

Creating an exciting and hands-on environment is the key to Cal Day's success. Students are paired with visitors to introduce them to GAMESMAN, and they can play games on our laptops or with a Wii™ controller. We also offer a prototype pen-and-paper to computer interface. Later, visitors can print a post-game analysis of their game play. Cal Day has served as an excellent opportunity for us to educate the public about our group and to present our recent results.

4.1.2. Website. Our site has evolved over time into quite a resource. Currently, it provides the history, game rules and analysis of the games implemented in our system. There is also background information on GAMESMAN, links to our open source code, and details about our membership. Our vision for the site is to be a one-stop portal for virtually every facet of the games we've solved. This would include history, rules, variants, analysis, strategies, and an intuitive web interface to play the games online.

4.2. Non-Traditional Opportunities

Through our outreach efforts, several people not associated with U. C. Berkeley have requested to join GamesCrafters. They have generally fallen into two categories: local high school talent and remote undergraduate collaborations.

We have had three high school students ask to join the group in the past six years, and in every case it was a resounding success. They each became an integral part of our team, were extremely productive, and at the end of their first semester, were virtually indistinguishable from the undergraduates. They commented that the experience gave them a great taste of development at the university level.

Unfortunately, we have not had the same luck with remote undergraduates working alone, without a remote faculty advisor. Time and time again they began with enthusiasm but dropped out within a few weeks. The lesson here may be that face-to-face faculty interaction is a necessary component for success.

5. Results

In this section, we present evidence to indicate that GamesCrafters has achieved its goal of motivating greater student engagement in computer science. We comment on observed team dynamics, and present alumni feedback.

5.1. Team Dynamics

GamesCrafters has a diverse group of students; women and ethnic minorities are equally represented in leadership roles and serve as role models to new members. Since we have never turned anyone away, we have students with a wide range of GPAs. Those with strong initiative take leadership roles, or join the more advanced students to work on system-wide projects. Overall, we have found that the social aspect of the group is a significant factor to its success, with students who join with friends having more fun and staying with the group for a longer time.

During our meetings, we have very open and lively discussions. The faculty supervisor often provides the theme and sets the stage, while group leaders play a mediating role. Students present prototypes, pitch new project ideas, or bring up technical issues that require the input of the entire group. Veterans are typically more vocal and contribute undocumented institutional knowledge. Most new members are naturally hesitant to participate in a large group setting, especially one in which everyone seems to

know each other well. Thus, we directly solicit their opinions to ensure that no one is excluded from the discussion. Indeed, all members are usually very willing to offer suggestions when approached.

Our group is very much student-driven. They take the initiative to choose their particular project, set realistic goals, and present their results at the close of the semester. The thriving social network combined with the mutual respect between teammates helps students spur each other on and almost always results in work of excellent quality.

5.2. Alumni Feedback

To understand how the experience affected our alumni, a short questionnaire was sent to our alumni mailing list asking how GamesCrafters enriched their undergraduate experiences, affected their perception and interest in computer science, and equipped them for life after graduation. Many of those who responded stated that the unique environment allowed them to develop as leaders, developers and researchers. For some it was an opportunity to apply classroom CS knowledge: "It pulled together all of the theoretical concepts from the various CS classes in providing my first practical application of my degree."

For many, it was their first taste of AI in a research environment. Those already interested in AI found it a fertile ground to further cultivate their interests. One alumni (now an AI graduate student) reported: "It had a significant impact on me. It gave me an introduction to the world of AI."

Former team leads, between juggling school demands and managing and educating new members, revealed that the experience taught them the value of successful time management. One alumnus echoed a common theme: "I wish I had had more time between school and other activities to devote to GamesCrafters!"

Alumni also mentioned that the group provided a nurturing and forgiving environment to learn how to work with a diverse set of people. The differing personalities meant that some students had to adapt, an invaluable skill that helped many in their careers. One alumnus wrote: "The diversity of the group brought together people with different perspectives and different skill sets into a greater whole."

The group was often the first time many were fully responsible for the entire development cycle of a large project, allowing them to grow as programmers: "The experience prepared me for a career in software development in ways that my CS classes never could."

Everyone who responded to the survey said his or her overall experience was a positive one. For some, it was even more meaningful: "GamesCrafters was the defining institution of my undergraduate career at Cal."

5.3. Critical Reflections

Our experience running GamesCrafters has exposed room for improvement in our process. As the group's size grew, it was often difficult to provide individual attention to each student. Also, with so many concurrent student development projects, it was hard to have enough time to support authoring research papers. We will explore limiting the group's size to address these concerns.

6. Conclusion

In this paper, we have presented GamesCrafters, an undergraduate computational game theory research and development group. Our intellectual core centers around GAMESMAN, an open-source AI software framework for encoding, brute-force solving and playing two-person abstract strategy games. After six years, 65 implemented games, and 200 members, we have learned how valuable the experience was for our students. We provided a nurturing group with entry points for undergraduates of all abilities. They could remain active as many semesters as they wanted, choose the projects that interested them, and assume leadership positions when they felt ready. Even though the dominant theme was AI, the projects spanned the spectrum of computer science. The group served as a springboard for our students to move on to other research groups, top graduate schools and software firms.

In terms of scaling our experience up to serve a broader audience, we see three possible models. Given that all of our software is open-source, we welcome other game theory faculty who wish to host their own "satellite" group. We also see the value of a group aimed at grade school children with an extremely simplified code base written in Scheme or Logo. Finally, we wish to stress that while game theory served as our particular appealing application, there are many other projects that can be equally effective to attract, motivate, engage and educate students.

Acknowledgements

We salute our 200+ GamesCrafters alumni. May their contagious enthusiasm and initiative continue unabated.

References

Allis, V. 1994, *Searching for Solutions in Games and Artificial Intelligence*. Ph.D. thesis, Department of Computer Science, University of Limburg.

AAUW Educational Foundation. 2000. *Educating Girls in the New Computer Age*. AAUW.

Garcia, D. D. 1995. *GAMESMAN, A Finite, Two-Person, Perfect-Information Game Generator*. M.S. Project, Dept. of Electrical Engineering and Computer Science, Univ. of California, Berkeley, Berkeley, Calif.

Gasser, R. "Solving Nine Men's Morris", *Computational Intelligence* **12**(1996): 24-41.

Schaeffer, J. et al. "Checkers is Solved", *Science*, 2007.

Van den Herik, H. J.; Uiterwijk, Jos W.H.M.; and van Rijswijck, J. 2002. "Games Solved: Now and in the Future". *Artificial Intelligence* **134**(2002): 277-311.