Please cite as:

Kiefer, P., Matyas, S., Schlieder C. (2005). State space analysis as a tool in the design of a smart opponent for a location-based game, In: Masuch, M. et al. (eds.): Proceedings of the Computer Science and Magic 2005, GC Developer Conference Science Track, Aug.16-18, 2005, Leipzig, Germany, Messe Leipzig

State space analysis as a tool in the design of a smart opponent for a location-based game

Peter Kiefer, Sebastian Matyas and Christoph Schlieder
Laboratory for Semantic Information Processing
Otto-Friedrich-University Bamberg, Germany
Email: {peter.kiefer, sebastian.matyas, christoph.schlieder}@wiai.uni-bamberg.de

Abstract

The paper discusses the design of a virtual smart opponent for a single player version of a location-based game. A novel type of game, CityPoker, serves as use case. We propose an approach for modeling the smartness of a generic and a specific opponent in three areas of competence: spatial search, spatial movement and logical reasoning. A computational analysis of the game's state space reveals that the temporal balance between reasoning and acting constitutes a crucial factor for winning. It is shown how to integrate the ability to deal with the temporal aspect of the game into the opponent model.

1. Introduction

Location-based games are currently on the rise. As noted in [2] they can be built on current wireless games for mobile phones and therefore benefit directly from the predicted huge increase in the mobile gaming market revenues. Moving and acting physically in an outdoor environment instead of staying at home and diving into a virtual world offers a totally different kind of game feeling. One of the most interesting aspects of location-based games is the new role of time and its connection to physical effort. Certainly, computer games played on the screen also use time as a central element for game design, but striving to be faster than an opponent in real-time and in real space - for instance by running or biking - constitutes an additional sportive challenge that is not found in conventional computer games. Nevertheless, location-based games are not reduced to a simple kind of race. They generally include strategic elements which force the players to explicitly plan their next moves.

Finding the right balance between the time spent on reasoning and the time spent on actions is a major key to success in the game. A related type of reasoning-acting balance is of interest to the game designer who compares different sets of rules for the game. Rules could emphasize physical action, e.g. by providing a winning strategy for a player who moves significantly faster than the opponent. Alternatively, the rules could emphasize strategic reasoning, leaving little impact for physical action. Understanding and controlling this balance is essential for creating an entertaining location-based game.

This paper studies the tradeoff between reasoning and acting at design time in the context of a novel location-based game, CityPoker. While most location-based games are developed for multiple players (or multiple teams), our objective is a game that can be played in both, single- and multi-player mode. A straightforward approach for adapting a multi-player game for a single player consists in replacing the other player(s) by virtual smart opponent(s). In location-based games, smartness includes the ability to adequately deal with the reasoning-acting balance.

For the designer as well as for the player, all reasoning about the game is based on the state space, which is the graph (or tree) having the possible states of the game as nodes and the valid transitions between states as edges. However, the traditional analysis of the state space by search (minimax algorithm) does not take the temporal aspect into account. Recently, adaptations of state space analysis have been proposed for real-time settings. An example is the sampling-based method using randomized alpha-beta trees proposed in [5]. Such approaches address the problem of planning an appropriate move at playing time. However, it does not solve the issue at design time where the game designer wants to know how changes in the game's rules affect the reasoning-acting balance.

We explore two different ways to integrate time into the traditional search-based approaches to state space analysis and describe how to use the results of the analysis as one parameter for the design of a smart opponent. The remainder of the paper is structured as follows. Section 2 gives an overview of CityPoker, the location-based game we choose as use case for our study. Section 3 describes the state space analysis with different time models for CityPoker. Section

4 presents our approach for modeling a smart opponent. We conclude with a discussion of related work and an outlook on future research in section 5.

2. CityPoker

CityPoker was conceived as a GPS-game in 2004 at the Laboratory for Semantic Information Technology of Bamberg University. It is played by two players (usually two teams) that start with a given poker hand and move around in an outdoor environment to find cards hidden in caches in order to improve their poker hand.

2.1. Rules of the game

The caches are five geographic locations each hiding two cards. GPS is used to find the caches. The players get a map of the area showing five large rectangular regions which contain the caches. Each of these cache regions is defined by a pair of imprecise geographic coordinates. Precision can be increased if a multiple choice quiz is solved: *"Cache 1 is located at N* 49° *53,XXX - E* 10° *53,YYY. Get more information by correctly answering the following question. Which pope is buried in Bamberg cathedral? (a) Clemens II, XXX=535 YYY=595 (b) ... (c) ..."'* If a team cannot answer a quiz either by the team members' knowledge or by asking someone on the street they will have to search all three possible locations for a cache. Because of the inaccuracy of the GPS localization method, an additional perceptual hint for the location of the cache is provided such as: *"The cache is under a big tree"*. Fig. 1 shows a GPS track for a cache region and the three possible cache locations. Note that the inaccuracy of the GPS signal is illustrated by the grey circles.

Each team may select one out of the two cards at every cache but a team may not visit a cache twice. CityPoker is a perfect information game which means that at any moment in time each team knows which cards the other team possesses as well as which cards are hidden in which cache. This implies communication between teams and is one of the reasons why it is attractive to design a CityPoker assistance system running on smartphones.

When a time limit is exceeded (e.g. 2 hours) the game ends and the team with the best poker hand wins. The game is played with a set of 20 cards: Ace, King, Queen, Jack, and 10 in the four suits spades, hearts, diamonds, and clubs. The winning order of the final poker hands is as follows: Royal Flush, Four of a Kind, Full House, Straight, Three of a Kind, Two Pair, and Pair. CityPoker rules do not rank the hands by suits so that a draw is possible.

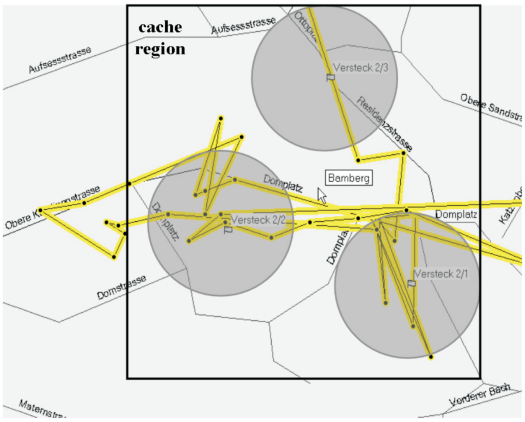

Figure 1. GPS track of a team in a cache region

2.2. Reasoning and acting

We illustrate the game style of CityPoker with an example of a CityPoker game played on April 16, 2004 in Bamberg. Fig. 2 shows how the cards were distributed between team A (= player A), team B (= player B), and the five caches. In CityPoker time matters: with the card distribution shown there is a simple winning strategy ("keep the Ace, get the four Kings") for a team that is capable of reaching four caches before the other team reaches even the first cache. However, the empirical differences between teams relating to spatial search and spatial movement never turned out to be that extreme.

Team A			Team B		
J♣	J♡	A♠	J♠	J♢	A♡
10♠	10♢		10♣	10♡	

C1		C2		C3		C4		C5	
A♢	K♡	K♠	Q♢	K♢	Q♡	A♣	Q♠	K♣	Q♣

Figure 2. Starting card distribution

Both teams were asked to protocol their move planning process which allowed us to reconstruct the following course of the game. Right from the start, team B aimed at obtaining a Royal Flush of hearts although they knew they would need the Jack of hearts from team A to achieve their goal. Team A was trying to get either four Queens or a Full House making the outcome dependent on the strategy of team B. Team B managed to get to cache 3 before team A reached cache 5. When team B picked the Queen of hearts in exchange for the 10 of clubs, team A drew the right conclusion that team B was aiming at a Royal Flush of hearts. Consequently, their first goal of getting four Queens was out of reach, because team B would keep its Queen of

hearts. Team A decided to cross team B's plan by keeping the Jack of hearts until the very end and discarding it at a cache which team B would not be allowed to visit anymore. Being at cache 5, they picked the King of clubs now aiming at a Full House with two Kings and three 10. After four further rounds of riding, searching and exchanging cards, the plan of team A succeeded making it the winner of the game.

3. State Space Analysis

A central requirement for the design of a game is the property of fairness, for players will very soon loose interest in an unfair game. In some games fairness is achieved by an almost symmetrical starting position as it is the case in chess. Other games start from a random position and therefore create fairness by introducing a probabilistic element. Differently from what the name of the game might suggest, fairness in CityPoker is not assured by a probabilistic element as it is in poker. The starting card distribution is not randomly generated but explicitly specified by the designer of the game (see Fig. 2). This raises the non-trivial problem of finding a fair distribution out of approximately $5.3 * 10^{12}$ possible card distributions. In this paper, we do not aim at identifying all fair card distributions, but limit ourselves to give computational criteria by which a card distribution can be categorized as being fair or unfair.

A fair card distribution offers both players the same chance of winning provided that both players act in an equally "smart" way. In other words, if both players move at same speed and do not make mistakes, the result of the game should be a draw.

3.1. CityPoker state space

The simplest assumption about the temporal order of events in CityPoker is that both players show equal abilities in spatial search and spatial movement. This leads to a game in which - very much as in the traditional game-theoretical analysis - the players move alternately. The model assumes that each player changes cards at exactly n caches $(n \in 1, 2, 3, 4, 5)$. Every action of changing cards is called a move. Finally, we assume that each player always chooses the optimal move. We refer to the set of these assumptions as Time Model 1.

This model describes a zero-sum game that can be analyzed by exploring the state space using a minimax approach with alpha-beta pruning ([4], [8]). The results of the minimax leave-nodes are determined by simple comparison of the two hands after n moves with 1 denoting a win for MAX, 0 denoting a draw and -1 denoting a win for MIN. Note that in a real game each of the two teams might be MAX or MIN, depending on who manages to first reach a cache in the game. A final result of 1 therefore would mean that there exists a winning strategy for the player who moves first. Of course, a fair starting card distribution would result in a 0.

No cache may be visited twice, so the number of possible moves decreases by 10 with every visited cache. Obviously, we will have to search $\left(\frac{5!}{(5-n)!} * 10^n\right)^2$ game states without pruning in the worst case.

3.2. Extended Time Models

As a next step, Time Model 1 was enhanced by allowing each of the players to willingly move twice, but only once per game i.e. the double move is comparable to a joker. The modification produces Time Model 2 (double move). Fig. 3 shows the structure of the game tree. A state annotated with (a,b,c) has been created by exchanging the a-th card on the hand with the c-th card in cache number b. Still each player may not move more than n times, so the breadth of the search tree is expanded while the depth remains the same. Moving twice means to rush from one cache to the next and changing cards before the other player is able to reach the next cache. This raises the question: Does the player who first sets his double move joker have any advantages? In other words: Is a team that lays more emphasis on acting than on reasoning more likely to win?

In some situations of CityPoker, the player who moves later might have an advantage for he may drop cards in caches that the other player is not allowed to visit anymore. Furthermore, spending more time on reasoning implies having more time to wait for the opponent's next move. In some cases, reacting on the other's moves instead of moving first turns out to be a good strategy. This was modeled by Time Model 3 (null move) which once more extends Time Model 1. Each player may wait once per game, i.e. make a null move, but only if the other player has not waited right before (for both players waiting directly one after another would not be that interesting). Again this adds new states to the search tree (Fig. 4).

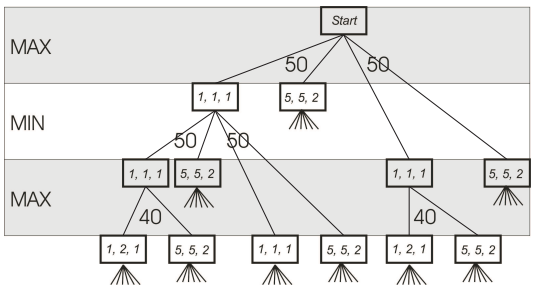

Figure 3. Time Model 2 (double move)

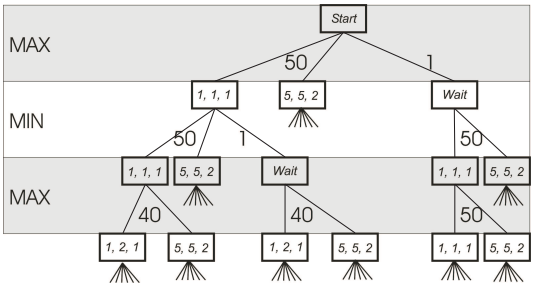

Figure 4. Time Model 3 (null move)

3.3. Results

Table 1 shows the three card distributions that have been analyzed. All of them have been designed intuitively with the intention of being highly symmetrical. Distribution III gives only one pair to each player. On the opposite, distribution II equips both with the maximum hand Royal Flush. Remind that according to our modified rules each player is forced to change cards and therefore cannot stick to the starting hand. Distribution I with its two pairs per player is somewhere in the middle and was used as starting distribution when CityPoker was first played (see Fig. 2).

Running minimax for each time model, each card distribution and different values for the number of moves per player (*n*), we obtained the results listed in Table 2. Cells in the table marked with an X were not computable in acceptable time because of search depth. For the same reason it was not possible to compute any results for *n*=5. But fortunately, a depth of 4 change actions should always be sufficient, because every type of hand (two pairs, flush, street,...) can be reached by changing 4 cards. So the results for *n*=4 come quite close to the real game of CityPoker.

Table 1. Analyzed card distributions

	Card Distributions								
	I (two pairs)			II (royal flush)			III (one pair)		
MAX	J♣	J♡	A♠	A♡	K♡	Q♡	10♡	10♠	J♢
	10♠	10♢		J♡	10♡		Q♡	K♣	
MIN	J♠	J♢	A♡	A♠	K♠	Q♠	10♣	10♢	J♡
	10♣	10♡		J♠	10♠		Q♣	K♠	
C1	A♢		K♡	A♣		K♣	A♡		A♣
C2	K♠		Q♢	A♢		K♢	A♠		K♡
C3	K♢		Q♡	Q♢		10♢	A♢		J♠
C4	A♣		Q♠	J♣		J♢	K♢		Q♠
C5	K♣		Q♣	Q♣		10♣	Q♢		J♣

Time Model 1 is the starting point of our interpretation: Distribution I seems to have an advantage for MIN but for *n*=4 it swaps to draw. Probably this is caused by the fact that for *n*=4 every type of hand may be a goal, while for *n*=3 some hands are never reachable from distribution I. The starting hand Royal Flush (II) is fair for every *n*. Distribution III is advantageous for MAX for search depths *n*=3 and *n*=4. The main changes of Time Model 2 are the wins for MAX in Royal Flush; a distribution that seemed to be absolutely fair suddenly favors the player moving first (MAX) when we give both players a double move joker. On the other hand, Time Model 3 does not affect the results for II, whereas it turns the win for MIN into a draw in distribution I. The results for distribution III are also changed in Time Model 3, but here an interpretation is difficult because of the missing value for *n*=4.

Each of the three time models captures another aspect of CityPoker. But in all cases, the state space analysis indicates that time is an essential factor for winning the game. In other words, there is a difference between heavy-reasoning and heavy-acting style of play on the outcome of the game. In the following, we will show how to model a smart opponent for our example game CityPoker with the aspect of time as main design parameter.

Table 2. Results of minimax analysis

		Card Distributions		
Time Models	n	I (two pairs)	II (rl. flush)	III (one pair)
1 alternately	2	-1	0	0
	3	-1	0	1
	4	0	0	1
2 double move	2	-1	0	1
	3	-1	1	1
	4	0	1	1
3 null move	2	0	0	1
	3	0	0	0
	4	X	X	X

4. Design of a smart opponent in a location-based game

Smartness of an opponent consists of a bundle of competences. In CityPoker, three competences are essential for a smart opponent: The ability to quickly perform spatial search, the ability to move fast between different locations, and the logical ability to plan moves. In the following we will distinguish between the modeling of a generic smart opponent and a specific smart opponent. A specific smart opponent is the virtual representation of a team existing in reality. In that case, data for parameterizing the virtual opponent is collected when a real team plays CityPoker at a particular geographic location, say, the city of Bamberg. Once the real team has played there, any other team may compete at a later date in the city of Bamberg with the virtual opponent. We believe that a game against a virtual but

specific team is more appealing than a game against some generic opponent.

4.1. Spatial search

Three activities are subsumed by the general competence for spatial search in CityPoker: answering the quiz, searching for the cache and communicating with the other team. The search competence is measured by taking the time t_{enter} when entering a cache region and $t_{finished}$ when the team communicates to the other team that it changed a card. With these two times we calculate $t_{n,m} = t_{finished} - t_{enter}$, which denotes the search time needed by team *m* in cache region *n*. Note that the time to answer the quiz is also included in the search time. Table 3 illustrates the search times for team 1 to *m* needed for cache regions 1 to 5. $t_{avg,i}$ is the average search time needed by team *i* in all cache regions. $t_{j,avg}$ is the average search time needed by all teams for cache region *j*. Note that in most cases some of the fields will be missing (e.g. $t_{2,i}$ in Tabel 3) so that the averages can only be calculated from the known search times. To determine the search time needed by a generic smart opponent in some cache region *n* we just take the average of cache region *n*: $t_{n,gen} = t_{n,avg}$.

Table 3. Search times for cache regions

cache region / **team**	1	2	3	4	5	*avg. of team*
1	$t_{1,1}$	$t_{2,1}$	$t_{3,1}$	$t_{4,1}$	$t_{5,1}$	$t_{avg,1}$
…	…	…	…	…	…	…
i	$t_{1,i}$	?	$t_{3,i}$	$t_{4,i}$	$t_{5,i}$	$t_{avg,i}$
…	…	…	…	…	…	…
m	$t_{1,m}$	…	…	…	$t_{5,m}$	$t_{avg,m}$
avg. of cache region	$t_{1,avg}$	$t_{2,avg}$	…	…	$t_{5,avg}$	

On the other hand, a specific smart opponent imitating team *i* should use the exact search time if available ($t_{n,spec(i)} = t_{n,i}$). If $t_{n,i}$ is missing, i.e. if team *i* has never been in cache region *n*, the search time must be estimated from the recorded times. This can be done in several ways. Our suggestion is to classify team *i* as fast or slow by comparing the times needed by team *i* in other cache regions with the average times. The average relative deviation is then taken to increase or decrease the average time other teams needed for the current cache region, refer to equation (1).

$$t_{n,spec(i)} = t_{n,avg} * \frac{\sum_{j \in V} \left(1 - \frac{t_{j,i}}{t_{j,avg}}\right)}{|V|} \quad (1)$$

V is the set of indices of all cache regions team *i* has already visited.

4.2. Spatial movement

The ability to quickly move between caches is important in CityPoker. Modeling is based on a route network of the gaming area where some of the network nodes represent the cache regions. The principle for computing parameter values for the smart opponent is the same as for the spatial search ability. However, we do not look at the search times but at the speed *v* of moving in the route network. For simplicity, the route network is assumed to be undirected, i.e. the speed $v_{x|y,m}$ of team *m* when moving from node x to node y is equals to $v_{y|x,m}$.

The generic smart opponent would again use the average speed between two nodes: $v_{x|y,gen} = v_{x|y,avg}$. The specific smart opponent uses the exact speeds if possible. If not, we estimate the speed according to the search time above using equation (2).

$$v_{x|y,spec(i)} = v_{x|y,avg} * \frac{\sum_{j \in R} \left(1 - \frac{v_{j,i}}{v_{j,avg}}\right)}{|R|} \quad (2)$$

R is the set of edges $a|b$ between nodes a and b where the imitated team *i* has already been traveling. The computation of the optimal path in the route network is performed at play time by standard algorithms (Dijkstra, A*) which are commonly used for that purpose in computer games [7].

4.3. Logical reasoning

The state space analysis showed us the impact of the time model establishing winning strategies for the game. If the smart opponent considers the possibility of a double move (Time Model 2), he might evaluate a certain card distribution as draw instead of a loss (Time Model 1). Our approach is to regulate the smartness of the opponent with respect to logical reasoning by working with different time models. A smart opponent with the logical ability implied by Time Model 1 is easier to beat than a smart opponent with the double move Time Model 2. To build a specific smart opponent representing an existing team, it is necessary to analyze games played by the team and to assume the model most closely matching the behavior shown.

At the beginning of a CityPoker game, the smart opponent performs a shallow search in the game tree to find the best strategy for the first moves. At this stage a good heuristic is necessary because only a search depth of two can be explored in reasonable time. After the first two moves search can be performed on the whole game tree. During a game, the strategy followed by a player needs to be reconsidered after each move of the opponent. Replanning involves a recomputing of the currently best path through the route network taking the spatial abilities of the players into account.

The three competences - spatial search, spatial movement, logical reasoning - interact and can all be used to adjust the smartness of a generic or a specific opponent. The temporal dimension of the game plays a different role in the three cases. Either time is directly measured like search time or time to move, or it is indirectly involved as in the time models on which the reasoning competence is based.

5. Related work and outlook

Strategic games of all kinds consist of phases of reasoning and acting. In order to win, a sufficient balance between these two factors must be found. This balance must be handled by a human player and therefore also by an artificial opponent. CityPoker proves to be an interesting use case for studying the reasoning-acting balance. In current research it is also used as a framework for studying the intentions of users by analyzing spatial motion patterns (see [9]).

Current research in location-based games has not paid much attention to the temporal aspect focusing mainly on design, architecture, context-awareness or interfaces (see [3] and [11]). To our knowledge, no formal analysis has been presented for evaluating the temporal constraints of the reasoning-acting balance of such games. Designers approach the problem by trial-and-error, that is, by playing and testing the game on the streets and then evaluating the feedback from the players (see [6]).

We argue that a formal analysis of the temporal constraints for reasoning and acting facilitates the building of a challenging smart opponent. In this paper, we presented state space analysis as a tool in the design of a smart opponent. We presented non-probabilistic time models that generalize the classical minimax search tree. In this first analysis of CityPoker, we did not follow a probabilistic approach such as the *-minimax algorithm described in [1] or [10] simply because we do not yet possess sufficient data for estimating the probability distribution of possible actions in CityPoker. We are planning to analyze CityPoker with suitable probabilistic methods in our future research.

Concentrating on the specific smart opponent, we used the expectation to model the generic smart opponent. Nevertheless, a more sophisticated generic smart opponent would be preferable. As future work we will try to incorporate other AI techniques like fuzzy logics into our design.

A model of a smart opponent in the domain of real-time strategy games (RTS) is presented in [5]. Here, probabilistic and non-probabilistic algorithms for decision taking were tested against each other in an abstract combat situation. The results show that probabilistic search methods perform better in games with higher simultaneous move dependence. However, the problem of reasoning-acting balance at design time which we focused on is not addressed.

Temporal constraints in computer real-time games like RTS are a lot harder than in location-based games since reasoning and acting must be done not in minutes but in seconds. Nevertheless, our method and models could be used for RTS games like Blizzard's Warcraft-series [1] or Ensemples Studio's Age of Empires-Series [2]. A separate analysis of early, mid and end game could render a search approach to state space analysis feasible. On the other hand, the analysis might be used for detecting game situations where the reasoning-acting balance is of major importance, like deciding whether building an expansion at a gold mine in Warcraft III is a good move or whether waiting for the opponent's move results in a better situation. This method could help game designers to build more of these challenging game situations into their games. Further research is required here.

References

[1] B. W. Ballard. The *-minimax search procedure for trees containing chance nodes. *Artificial Intelligence*, 21(3):327 – 350, 1983.

[2] S. Benford, R. Anastasi, M. Flintham, A. Drozd, A. Crabtree, and C. Greenhalgh. Coping with uncertainty in a location-based game. *IEEE Pervasive Computing*, 2(3):34 – 41, 2003.

[3] M. Flintham, R. Anastasi, S. Benford, T. Hemmings, A. Crabtree, C. Greenhalgh, T. Rodden, N. Tandavanitj, M. Adams, and J. Row-Farr. Where on-line meets on-the-streets: Experience with mobile mixed reality games. In *CHI 2003: Conference on Human Factors in Computing Systems*. ACM Press Florida, 2003.

[4] D. Knuth and R. Moore. An analysis of alpha-beta prunning. *Artificial Intelligence*, 6(4):293 – 326, 1975.

[5] A. Kovarsky and M. Buro. Heuristic search applied to abstract combat games. In *Proceedings of the Eighteenth Canadian Conference on Artificial Intelligence*, 2005.

[6] P. Lankoski, S. Heli, J. Nummela, J. Lahti, F. Myr, and L. Ermi. A case study in pervasive game design: The songs of north. In *NordiCHI '04*, pages 413 – 16. ACM, October 2004.

[7] A. Nareyek. AI in computer games. *ACM*, February:58 – 65, 2004.

[8] S. Russell and P. Norvig. *Artificial Intelligence A Modern Approach*. Prentice Hall, 2003.

[9] C. Schlieder and A. Werner. Behavior in spatial partonomies. In F. et al, editor, *Spatial Cognition III*, pages 401 – 414. Berlin: Springer, 2003.

[10] M. B. T. Hauk and J. Schaeffer. Rediscovering *-minimax search. In *Computer and Games Conference*, 2004.

[11] B. Thomas, B. Close, J. Donoghue, J. Squires, P. D. Bondi, and W. Piekarski. ARQuake: A first person indoor/outdoor augmented reality application. *Personal and Ubiquitous Computing*, 6:75 – 86, 2002.

[1] www.blizzard.com

[2] www.microsoft.com/games/empires/