From: AAAI Technical Report SS-99-02. Compilation copyright © 1999, AAAI (www.aaai.org). All rights reserved.

Explanations Count

Ian Frank
Complex Games Lab
Electrotechnical Laboratory
Umezono 1-1-4, Tsukuba
Ibaraki, JAPAN 305
ianf@etl.go.jp

Abstract

In research areas such as decision support systems, one of the crucial features is the ability to automatically explain decisions. However, this ability does not typically play a large role in computer games. Here, I suggest that this situation will change: research on the automatic generation of explanations will have real import for many types of computer games. I discuss four possible ways to include explanations in games, and for each of these give examples of current research projects. I make concrete suggestions of how the ideas from these projects can be applied in a wider context: that of commercial computer games.

Introduction

A very good summary of the state of AI in the commercial games industry can be found in (Woodcock 98). However, reflecting on this summary reveals a small surprise: only one mention is made of natural language abilities. Apart from this (a suggestion that speech understanding software will become more widely used in game interfaces), 'game AI' is mostly interpreted to mean 'how to make computer controlled objects and characters move and act sensibly'. In this paper, I suggest that another important — and often overlooked — aspect of AI in computer games is the automatic generation of explanatory statements.

For the purposes of this paper, let me simplify the possible points at which explanations can be incorporated in a computer game as follows:

- **Problem Solving**: if, at any point in a game (or during off-line analysis), the computer can find the optimal way to play a sub-problem in the game, the optimal strategy can be explained to a human user.
- **Commentary**: as a game progresses, the computer can follow and describe it in real time (irrespective of whether the players are humans or computers).
- **Post-mortems**: after a game is over, the computer can analyse the course of the game (irrespective of whether the players were humans or computers).
- **Tutoring**: during play, the computer can use its knowledge of the game to coach a (human) user.

This breakdown is intended as a basic characterisation of the possibilities rather than as a definitive classification. In general, the boundaries between the four categories may be fuzzy. For instance, explaining how to solve sub-problems in a game and tutoring a game both involve communicating a program's knowledge to the player. A genuine tutoring system, however, will additionally monitor and assess the moves of the player, to judge what still needs to be taught. Commentaries and post-mortems also involve interpreting the moves of players, but this time of *all* the players in the game. The obligation to follow the game in real-time is also most severe when producing commentaries.

The following sections look in turn at examples of research projects that fall within each of the four categories. In particular, I review some of my own work on producing explanations in the very different games of Bridge and soccer. In each section, I identify the research issues, and the benefits that come with the ability to explain. I also give concrete examples of how research ideas can be expected to apply in the wider context of commercial games. These applications include the development of commentators for computer games tournaments, the automatic generation of short diaries to enhance the characters of game NPCs (Non-Player Characters), and the improvement of the ability to constructively handle failure.

Problem Solving in Games

Many game-playing programs can solve some aspects of a game optimally. An example of this kind of problem solving ability is the endgame databases used by many board-playing programs. In chess and checkers, for instance, computers can store the optimal moves for many configurations of small numbers of pieces. However, although such databases allow perfect play in many situations, explaining this play can be very hard because it requires *understanding* the database information in human terms. A good example here is the early computer chess research of Komissarchik and Futer, who were described as being "at their wits' end when trying to explicate rules fit for humans to apply" from their endgame databases (Allis *et al* 91).

What does it mean to understand the knowledge of a system in human terms? As an example, I will look briefly at work I have carried out with David Basin and Alan Bundy on the Bridge program FINESSE (Frank *et al* 92; Frank 96). One of the fea-

tures of this system is its ability to solve the Bridge sub-problem of optimal play in a single suit. To do this, FINESSE uses a high-level formalisation of *tactics*, which capture the commonly occurring patterns (such as *finessing* and *cashing*) that human players look for when examining such problems. Just as humans use these patterns to exclude unpromising plays from consideration, FINESSE uses a planner that constrains the possible plays to a set of just seven tactics. This gives FINESSE two advantages. Firstly, searching the space of tactics instead of the space of legal moves reduces the size of typical game trees by two or three orders of magnitude. This reduction enables the use of new search algorithms (Frank *et al* 98; Frank & Basin 98) for identifying optimal (pure) strategies. Secondly, it becomes possible to *communicate* FINESSE's solutions to a user in meaningful terms (see Figure 1).

The benefit of communication is that it can work both ways: from computer to user, and from user to computer. FINESSE, for example, has clear potential for understanding and answering users' "Why...?" questions in plain text during a game. Another exceptionally good illustration here is the theorem-proving research carried out in Edinburgh (Bundy *et al* 91) (of which FINESSE is actually an adaptation). When using a theorem prover, there will be theorems on which the program fails. One solution is then to develop *interactive* theorem provers. (Bundy 99) describes the rationale of such systems as:

> "the burden of finding a proof is divided between the computer and a human user...Usually, the human role is to make the difficult proof guidance decisions while the computer takes care of the bookkeeping and the routine steps."

In order to effectively guide the prover it is vital for a human user to understand the nature of the emerging proof. A high-level representation (such as tactics) provides this ability, and allows the user to direct the prover in a meaningful way. For example, if the prover attempts to apply a tactic but one of the preconditions fails, it can ask the user if there is a way to patch this failure or even attempt to find a patch itself (Lowe *et al* 98). So, a high-level representation increases not only the ability to communicate a program's knowledge, but also the possibilities for the user to communicate problem-solving knowledge to the program.

The carry-over potential of research on these issues can be made clear by simply considering the genre of strategy games. Many of these games involve making the player an 'emperor' or a 'general' whose task is to develop, conquer or defend a territory. The player works within the environment provided by the game to try to achieve the game's goals. Despite being an 'emperor' or a 'general' in name, though, players often have mostly low-level commands at their disposal, such as "build a barracks here," "move this unit here," or "fell these trees to make lumber". Implementing an overall strategy in terms of these low-level commands has the potential to become tedious. Just as in theorem-proving, though, a higher-level representation

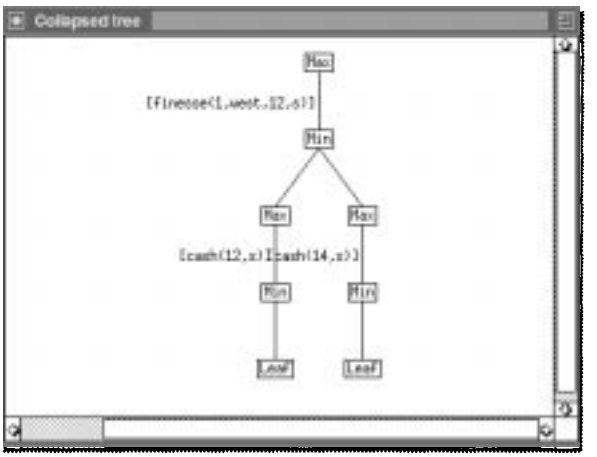

Figure 1: Screen shot from FINESSE, showing the solution for the simple single-suit problem where North holds ♠AQ and South holds ♠2. The complete game tree for this situation (even under simplifying assumptions such as not distinguishing between the missing low cards) has 76 leaf nodes. The tree searched using tactics has just 14 leaf nodes, and the optimal strategy for taking the maximum two tricks collapses into the two branches shown. At the root of the tree, the first action is to finesse the Queen. If West plays the King (the left-hand MIN branch), North wins with the Ace and then cashes the Queen. If neither defender plays the King (the right-hand MIN branch), the finesse succeeds and the Ace can be cashed. FINESSE's tactic-based representation allows it to explain this solution as "Finesse the Queen: this leads to two tricks if West holds the King." Human users can easily understand such explanations, because all FINESSE's tactics derive from Bridge plays familiar to human players.

of the possible commands would allow the 'burden' of playing the game to be divided between the computer and the user. The player would be able to make the difficult overall strategy decisions (such as "establish a beachhead on this island") while the computer takes care of the 'micro-management' and the routine steps.

Interestingly, an existing trend in commercial games directly favours the high-level representation of problem-solving techniques. A number of recent games, such as QUAKE and UNREAL, separate the code controlling the actions of NPCs into *extensible AIs*: files that can be freely modified by game players using scripts and code plug-ins. (Woodcock 98) reports some of the problems facing game developers wanting to take this approach, such as the technical challenges of how to actually provide the hooks in a game for supporting the extensible AIs, and how to prevent hackers using the hooks to do unspeakable things to others' computers. However, the discussion above illustrates that extensible AIs will have a valuable side-effect: the presumably high-level nature of the command language will make it easier to explain to players what is actually happening in the gamef, and allow them to interact more effectively with the game engine itself.

Commentating a Game

In contrast to the discussion of the previous section, the important issue in commentary is the ability to describe events as they happen. To give an example of a research project on commentary, I will look at the game of soccer, and in particular at the MIKE system developed at ETL in Japan (Tanaka-Ishii *et al* 98).

The difficulties posed by the domain of soccer have recently led to it being proposed as a new standard problem for AI research (Kitano *et al* 97b). As part of this challenge, a series of Robot Soccer World Cup (RoboCup) tournaments has been inaugurated, with the ultimate goal of developing "a robot soccer team that can beat the Brazil world-cup team" (Kitano *et al* 97a). RoboCup has three leagues: one for small robots, one for medium-sized robots, and one league run entirely as a simulation. Games in the simulator league are conducted using the Soccer Server (Noda *et al* 98), and it is these simulations for which MIKE provides a commentary. In the most recent RoboCup, thirty-six teams competed in the simulator league. Such a large number of teams makes it very difficult to recruit enough human volunteers to describe all the games, so the most practical remaining way of adding atmosphere for the spectators at RoboCup events is an automatic commentary system.

The MIKE system produces text or spoken commentary directly from the output of the Soccer Server. Every 100ms, detailed information is received from the Soccer Server on player location and orientation (for all players), the ball location, and the game score and play modes (such as throw-ins, goal kicks, *etc*). MIKE then uses this information to make a commentary that consists of any combination of the possible repertoire of remarks shown in Figure 2. Currently, this output is produced with the simple mechanism of template matching, converting the system's internal language into appropriate expressions in either English, Japanese or French. To reduce repetition, this matching process is non-deterministic, and several templates are available for each decision. An example of MIKE's English language commentary might be "Interception by the Yellow-Team,... Yellow10 shoots!... Red4,... Yellow11's shot!... The Yellow-Team's 7th goal!! 7 to 0! Another goal by Yellow11!"

Space constraints make it difficult to give a full description of MIKE's architecture in this paper. However, the difference in nature from the task of explaining sub-problems discussed in the previous section should be apparent from Figure 2: the large majority of MIKE's comments are based on *statistical* evaluations of the game to date. It is less important (indeed probably impossible) to calculate the optimal ways that the players should perform given tasks for the remainder of the game. Rather, assessing the current balance of the game and the changes in strategy becomes the key challenge.

This relaxation of the need for calculating optimal play suggests a novel application: a system that can commentate for the many tournaments where computer programs are played against each other (or against humans). There are many such tournaments (for example the Computer Games Olympiads in London, and the FOST Cup for computer Go) and they invariably feature large numbers of excited participants huddled around large numbers of keyboards and screens. Without a very good knowledge of the game being played (or an interest in one of the competition entrants), watching these spectacles can be hard work. As in RoboCup competitions, human commentators, if present, typically concentrate on games between the leading contenders. So, as in RoboCup, an automatic commentator is the most practical way to add atmosphere. There is no paradox involved in the likelihood that the commentary program will probably not be able to match the competition entrants in terms of playing strength, since there is no necessity for the commentary program to know the best moves in the game. To commentate, it is sufficient to be able to distinguish the good from the bad, and to cope with the real-time constraints, such as injecting atmosphere when the game is nearing its climax.

- **Explanation of complex events**. Formation changes, position change, and advanced plays.
- **Evaluation of team play**. Average formations, formations at a certain moment, player locations, indication of active or problematic players, wasteful movements.
- **Suggestions for improving play**. Loose defence areas, better locations for inactive players, and 'should-have' comments about failed passes.
- **Predictions**. Prediction of passes and shots at goal. Also, prediction of game result by comparing team performance metrics against statistics compiled from a database of played matches.
- **Set pieces**. Goal kicks, throw ins, kick offs, corner kicks, and free kicks.
- **Passwork**. Basic tracking of ball-by-ball play.

Figure 2: MIKE's commentary repertoire

Note that computer game tournaments lie at the interface between academic and commercial domains, since the competing programs are drawn from disparate sources. However, there is no reason why computers could not also be used to commentate in the growing number of *human* computer game-playing tournaments. According to the Times newspaper (Powell 98), there are now enough of these tournaments to support the world's first professional games player, who is expected to earn $100,000 a year in prize money and sponsorship deals. Extending this argument a little further, another application of automatic commentary systems could be in multi-player games (such as networked games), when one player is acting as the observer of the conflict between two others.

An important research issue for any automated commentary system is *discourse planning*. That is, there are typically many options for ordering the descriptions of related facts. In soccer, for example, the players of

one team may exploit their opponents' man-marking tactics to drag defenders out of position. In a commentary, this observation could be pointed out at any time. However, it makes more sense to state the fact when the viewers can see it happening, or better still to use it as an explanation of the cause of some event (such as the scoring of a goal). Interestingly, similar considerations of planning an explanation can occur in Bridge, where the information revealed by the play of the cards means that the best continuation at any point in the game depends not just on the current possible moves, but also on the game history. Neither MIKE nor FINESSE make any attempt to tackle discourse planning, although MIKE does contain a collection of over 50 inference rules that identify relationships between events that it has identified in the game. In genuine natural language systems, more flexibility is produced by the introduction of *intermediate logical forms*. These are stages of representation through which an expression will pass as it is processed from the internal logical language into the *surface structure* from which the textual explanation can be directly produced. A simple example of this in the domain of game-playing is Davey's PROTEUS system (Davey 78), which produces a commentary on a game of noughts and crosses.

Post-mortems

Human games players love to indulge in post-mortems. Indeed, they will sometimes spend longer in a discussion after a game than in playing the game itself. In computer games, however, the automation of post-mortem analysis has been rather neglected. It is just starting in chess, for instance, where there is an annual award for the best game annotation produced by a computer (Björnsson & Marsland 98).

At a simple level, both MIKE and FINESSE could also be adapted to post-game analysis, with MIKE summarising the statistics it collects, and FINESSE indicating where the plays in the game deviated from optimality. However, in this paper, I want to emphasise the importance of describing the course of a game *succinctly*. Many existing commercial games, for instance, will give tables or graphs of statistics at the end of a game, but I argue that this cannot truly be counted as a post-mortem. Rather, a genuine post-mortem should consist of natural language text describing how the game was played and what things the player(s) did well or badly. The issue of discourse planning is also of great relevance here, as the way that events in the game are linked together affects the succinctness of the explanation. Note that the chess annotation programs mentioned above largely sidestep the problem of discourse planning through the simple expedient of giving as output the entire set of game moves, interspersed with automatically generated comments. A more general natural-language post-mortem would call for the game developers to invoke the atmosphere of a game. For example, just a few of the things that could be described are a player's famous victories, ignominious losses, and the turning points in the game.

Note that as well as reporting to a player after a game, a further use of a post-mortem analysis is to enliven a numerical ranking system. Particularly in networked games, ranking systems are widely used, and players come back many times to increase their standings. Yet look at any ranking list and it is a soulless affair, with just names and numbers. Some automatically-generated histories of players' games and achievements could significantly enhance the interest of these rankings.

Indeed, there is actually no reason to limit post-mortem analyses to the end of a game: they can also be applied to *episodes* within a game. For example, consider the strategy game classic, Civilisation. When playing this game, pop-up windows will sometimes appear to inform the player of the demise of an opponent. This pop-up window may contain a message such as "The Babylonian civilisation has been destroyed by the Aztecs". I suggest that a natural language post-mortem on the history of the Aztec nation would be of much more interest to the player, especially if it contained information about the playing styles of possible future opponents. Extending this example, another possible function of the spies in games such as Civilisation could be to send back natural language reports on how opponents have acted in the game to date.

For other types of games, similar possibilities hold. For example, in adventure games (where the player progresses through a planned world), we already have "automatic mapping". Why not also "automatic diaries" that are more than just lists of events that happened to the player? What I envisage here are real stories; stories that may even incorporate bias, so that the player can be cast in the role of a hero, a coward, or a villain. And for games where the player guides a *group* of characters through the story-line, an individual diary could be kept for each. An example of this latter type of game is XCOM, in which the player sends a squad of commandos on repeated missions to repel alien landings on Earth. The most recent sequel of this game (XCOM-III) goes some way towards adding mission post-mortems, by including a count of service days, missions, kills and improvements for every trooper in a player's squad. However, this could be extended with natural language stories, and descriptions of heroic escapades or "deeds of valour".

Tutoring

Explanation during tutoring is the last of the explanation categories I examine, partly because it shares some aspects of each of the other three: any problem solving or post-mortem or commentary system can probably be adapted to give at least a little feedback on a player's actual move decisions during a game. The genuine research issues involved are also varied and complex. A good starting point is the LISP tutor developed by (Anderson *et al* 86), which tackles questions such as the best timing for feedback, the importance of not interrupting, and the construction of user models.

It shouldn't be overlooked that in games such as chess, where the computer may be much stronger than the human player, a tutoring ability may be essential

to maintain the player's interest. Games that include more than a superficial tutoring ability also relieve players from some of the task of reading the huge manuals that seem to accompany most current games.

A tendency towards increased player feedback is already being seen in another large class of game situations: the strategic management games that are widely used in educational institutions. (Keys 97) identifies one of the future trends in these games as "more technological support for participants, such as the inclusion of decision support packages, and extensive notes on developing strategic and tactical plans".

Exploiting Social Responses

Incorporating explanation abilities into a game may seem like a lot of effort, but in fact a little bit of natural language goes a surprisingly long way. For instance, when giving demonstrations of the MIKE system, inserting a simple joke in the template database makes people laugh. They think this is great. It's often the thing that is first on peoples' lips after a presentation.

As another example, Jonathan Schaeffer tells a very revealing story about a chess program he authored. Early versions included an insult generator. This was just a piece of code that would now and then randomly generate an insult and print it out to the user. In an upgrade to his program, Schaeffer removed the insult option, thinking that nobody was using it. He got more requests asking for its re-instatement than about any other feature of his program.

These are two simple examples of what (Reeves & Nass 96) have called the Media Equation. Reeves and Nass point out that human brains respond to 20th century technology (like movies and computers) with old evolutionary aspects of the brain. We are evolved to a world in which anything that interacts, uses language, or fills a social role, deserves human treatment. Since our conscious brains have not caught up with these unconscious and automatic responses, we subconsciously treat media socially (hence the media equation, "media = reality"). Thus, human users will (without consciously realising it) *like* a computer better when it uses humour, and also will think that it is *smarter*. A computer that criticises will also be perceived as more intelligent. One of the advantages of automated explanations, then, is that they can make use of these and other social strategies, such as praise, flattery and imitation of the user's personality. Nass sums up this situation as "You get 10% of the credit for *being* smart, but you get 90% of the credit for *seeming* smart."

Conclusions

In this paper, I have looked at the role of explanations in computer games. I discussed how to increase a program's explanatory abilities in the four categories of problem solving, commentary, post-mortems, and tutoring. I made suggestions of likely carry-overs between academic research and the entertainment industry, and ended by noting how automated explanation systems also offer the potential of exploiting peoples' social reactions to modern media.

References

L.V. Allis, H.J. van den Herik, and I.S. Herschberg. Which games will survive? In D.N.L. Levy and D.F. Beal, editors, *Heuristic Programming in Artificial Intelligence 2 – The Second Computer Olympiad*, pages 232–243. Ellis Horwood, 1991.

J. R. Anderson, R. Farrell, and R. Sauers. The automated tutoring of introductory computer programming. *Comm. of the ACM*, 29(9):842–849, 1986.

Y. Björnsson and T. Marsland. Fritz 5.0 wins the 1997 Herschberg best-annotation award. *ICCA Journal*, 21(1):61–66, March 1998.

A. Bundy. A survey of automated deduction. In M. Fisher, editor, Celebratory edition of Springer LNAI Series, Number 1500. Title to be announced, 1999.

Alan Bundy, F. van Harmelen, J. Hesketh, and A. Smaill. Experiments with proof plans for induction. *Jour of Automated Reasoning*, 7:303–324, 1991.

Anthony Davey. *Discourse production : a computer model of some aspects of a speaker.* Edinburgh Univ. Press, 1978.

I. Frank. *Search and Planning under Incomplete Information: A Study using Bridge Card Play.* PhD thesis, Dept of AI, Edinburgh, 1996. Also published by Springer-Verlag in the Distinguished Dissertations Series, 1998.

I. Frank and D. Basin. Optimal play against best defence: Complexity and heuristics. *Proceedings of The 1st Intl Conf on Computers and Games, CG98*, Tsukuba, 1998. Springer. LNCS/AI Series, to appear.

I. Frank, D. Basin, and A. Bundy. An adaptation of proof-planning to declarer play in bridge. In *Proceedings of ECAI-92*, pages 72–76, Austria, 1992.

I. Frank, D. Basin, and H. Matsubara. Finding optimal strategies for imperfect information games. In *Proceedings of AAAI-98*, pages 500–507, 1998.

J.B. Keys. Strategic management games: A review. *Simulation and Gaming*, 28(4):395–422, Dec 1997.

H. Kitano, M. Asada, Y. Kuniyoshi, I. Noda, E. Osawa, and H. Matsubara. RoboCup: A challenge problem for AI. *AI Magazine*, pages 73–85, Spring 1997.

H. Kitano, M. Tambe, P. Stone, M. Veloso, S. Coradeschi, E. Osawa, H. Matsubara, I. Noda, and M. Asada. The RoboCup synthetic agent challenge 97. In *Proc. IJCAI-97*, pages 24–29, Japan, 1997.

H. Lowe, A. Bundy, and D. McLean. The use of proof planning for co-operative theorem proving. *Journal of Symbolic Computation*, 25(2):239–261, Feb 1998.

I. Noda, H. Matsubara, K. Hiraki, and I. Frank. Soccer Server: a tool for research on multi-agent systems. *Applied Artificial Intelligence*, 12(2–3):233–251, 1998.

N. Powell. School drop-out enjoys life in the fast lane. In *The Times*, London, October 21 1998.

B. Reeves and C. Nass. *The Media Equation.* CSLI Publications, 1996.

K. Tanaka-Ishii, I. Noda, I. Frank, H. Nakashima, K. Hasida, and H. Matsubara. MIKE: An automatic commentary system for soccer. In *Proceedings of ICMAS-98*, pages 285–292, 1998.

S. Woodcock. Game AI: The state of the industry. *Game Developer*, pages 29–35, October 1998.