Cognitive Multi-character Systems for Interactive Entertainment

John Funge
Sony Computer Entertainment America
www.cs.toronto.edu/∼funge

Steven Shapiro
University of Toronto
steven@ai.toronto.edu

Abstract

Researchers in the field of artificial intelligence (AI) are becoming increasingly interested in computer games as a vehicle for their research. From the researcher's point of view this makes sense as many interesting and challenging AI problems arise quite naturally in the context of computer games. Of course, the hope is that the relationship is a symbiotic one so that the incorporation of AI techniques will lead to more interesting and enjoyable computer games. One question that arises, however, is how far this process can continue? In particular, what, if any, are the technical roadblocks to applying new AI research to interactive entertainment, and what would be the expected benefits? In this paper, we will therefore take a critical look at some AI techniques on the horizon of our own current research in developing the software infrastructure required to view interactive entertainment applications as cognitive multi-character systems.

Introduction

In [9] John McCarthy identified the computer game Lemmings (by Psygnosis) as a worthy vehicle for academic research into a variety of tough AI problems. Others have also realized the potential of computer games to provide a convenient platform for performing advanced research [6]. Although care must be taken not to over-simplify the important issues, one of the key advantages of conducting research in a virtual world is that many problems can be studied without worrying about the tiresome details that plague robotics. For example, research in computer vision can be conducted without the usual headaches associated with camera calibration [10], path planning can be studied without worrying about low-level robot locomotion, etc.

Computer games, however, should be viewed as much more than simply providing a convenient virtual world laboratory. They also present an exciting opportunity to transfer technology to a burgeoning multi-billion dollar industry that is thirsty for new ways to improve the overall gaming experience. When we also factor in other emerging interactive entertainment applications, such as smart toys, we can see that the stage is set for a monumental symbiosis between academia and the computer game industry.

Nevertheless, significant challenges face us as we move beyond the application of textbook AI techniques to computer games toward the incorporation of more cutting-edge research. In particular, many potentially useful ideas have been developed with more theoretical applications in mind. For example, in our own work in building intelligent agents, much of the important groundwork has been laid in the context of multi-agent system specification and verification. Adapting this work to the inherently implementation-centric world of interactive entertainment can be extremely challenging. In the remainder of this short paper we will describe those challenges, how we envisage overcoming them, and what prospects exist for the emergence of useful tools.

Background

In [2, 3, 4] some relatively sophisticated knowledge representation (KR) work is introduced to the gaming and animation communities. In particular, *cognitive modeling* is introduced as the new substantive apex of the computer graphics modeling hierarchy. Cognitive models go beyond the behavioral models that have been used previously in computer games in that they govern what a character knows, how that knowledge is acquired, and how it can be used to plan actions. Cognitive models are applicable to directing the new breed of highly autonomous, quasi-intelligent characters that are beginning to find use in interactive entertainment. Moreover, cognitive models can play subsidiary roles in controlling cinematography and lighting.

Nevertheless, cognitive modeling is a potentially vast topic whose riches we have only just begun to be explored. There remains a wealth of additional interesting cognitive phenomena to consider, such as goals and intentions [1, 11], memory [8], etc.

Cognitive Multi-character Systems

The characters in computer games and interactive entertainment are for the most part meant to simulate human or human-like behaviors. They are modeled after humans — perhaps endowed with superhuman abilities and/or supervillanous motives, but at their core they act like people in some limited fashion. How many of us have played Pacman and have thought to ourselves just as our last “woman” was about to bite the dust: “she knew I was going for the power pill and she cut me off” or “they are trying to surround me”.

In other words, it is very natural for us to attribute mental states, e.g., knowledge, goals, intentions, etc. to characters in a computer game and interpret their behavior using mental vocabulary. Now, imagine if the characters in a computer game could attribute mental states to themselves and to other characters and even to the people playing the game. They could then plan strategies together based on their own goals (which may be just to kill the human player, but they might be more complex or less violent) and what they can infer about the goals of the players. Then a typical arcade game would cease to be simply a test of reflexes and hand-eye coordination, but rather it would necessitate more complex strategies on the part of the players. We could imagine teams of people playing a game together, communicating and strategizing with each other as they play against teams of cognitive characters that are doing the same thing: the real versus the virtual.

How could such a game be implemented? It would certainly help if the language used to implement such games had primitives to express the behavior of the characters in terms of their mental states and the mental states of other players. For example, the program for a Pacman could include statements such as: if I think the player is going for the power pill, I will try to cut him off. In order to implement behavior such as trying to surround the player, it would be useful to have communication primitives built in to the language so that characters can express their goals and knowledge to each other to coordinate their actions. In addition to being able to express the goals of the characters, it would be handy to endow them with the ability to construct their own plans to achieve their goals. In order to be able to construct a plan to achieve a goal in a given environment, it is necessary to know what are the actions that can be performed in the environment and what effects the actions have on the environment.

In other words, what we would like is to have a language that:

- Allows us to define the behavior of the characters in terms of their mental states, e.g., their knowledge, goals, intentions, etc.
- Contains primitives for communication which automatically update characters' mental states appropriately when communication occurs. For example, if one character asks another to do something, the other character should, in the absence of conflicting goals, adopt the goal of doing that thing, and the first character should gain the knowledge that the other character adopted the goal of doing it, etc. On the other hand, if a player asks a character to do something, the character should not be so willing to comply. The assumption here is that characters and players are on opposing sides. However, this need not be the case, for we could certainly have games where some players and some characters are on the same side.
- Ensures that the characters can introspect their mental states, e.g., if an agent has a goal to do something then the agent should know it has that goal.
- Gives the characters explicit knowledge of the physical environment, so they can construct and execute plans to achieve their goals in the environment when necessary.

Implementation

In [11], a language was presented for specifying and proving properties of systems like the cognitive multi-character system we have been discussing. It has all the features previously listed as requirements. However, it is a language designed to be used to describe an entire system from an external point of view, rather than a language for implementing such systems. While it is useful for proving properties of systems because it has a well-defined semantics in predicate logic, the operators used to describe characters' mental states are not directly implementable. On the other hand, the specification language is based on a concurrent programming language (ConGolog) that has an interpreter. A system is defined, in part, by writing ConGolog procedures, and these procedures could easily be implemented if the mental state operators were implemented.

An important first step towards implementing mental state operators was taken in [2], where an implementation for knowledge is given that leverages interval arithmetic. Unfortunately, that implementation currently lacks introspection and an explicit representation of goals.

As a concrete example of the kind of system we would like to implement consider the dinosaurs example in [3, 4]. There, the emphasis was placed on endowing a T-Rex with cognitive abilities to allow it to herd some unruly Raptors through a pass. There is no reason why we could not similarly endow the Raptors as well. This would allow the animation of much more complex dinosaur behavior.[1] A lone Raptor is no match for the T-Rex, but imagine the following scenario in which a pack of cunning Raptors conspire to fell their large opponent. Through cognitive modeling, the Raptors hatch a strategic plan—the ambush! Based on their domain knowledge, the Raptors have inferred that the T-Rex's size, his big asset in open terrain, would hamper his maneuverability within the narrow passage under the arch. The leader of the pack plays the decoy, luring their unsuspecting opponent into the narrow passage. Her pack mates, who have assumed their positions near both ends of the passage, rush into it on command. Some Raptors jump on the T-Rex and chomp down on his back while others bite into his legs. Thus the pack overcomes the brute through strategic planning, cooperation, and sheer number.

Real-time Performance

One of the most challenging aspects to building a cognitive multi-character for computer games is to ensure real-time performance. In this regard, the rapid development of modern processors is significant. Nevertheless, many of the algorithms have exponential worst-case complexity or are (in general) not even computable. Therefore, we could not hope to give our system free reign to run til completion of its assigned task.

The solution employed in [4] was three-fold:

1. Characters are constrained to search for plans of less than a certain length. In particular, with a maximum plan length of 6 a cognitive T-Rex was able to operate in real-time on a 400MHz Pentium II and successfully generate

[1]See Robert T. Bakker's captivating novel *Raptor Red* (Bantam, 1996).

paths to herd a pack of unruly Raptors to a desired location. Note that this necessitates the formulation of a goal that expresses partial success toward the ultimate goal, and periodic re-planning when the current plan becomes outdated. By providing problem-specific heuristics, complex actions [7] also help to increase the length of the plan that can be generated in the allotted time.

2. There is an underlying reactive behavior substrate that can act as a backup system if a plan cannot be generated in time. Although the reactive system lacks the sophistication to move the character closer to fulfilling its goals, it does prevent the character from doing anything "stupid". For example, the character will avoid obstacles even if it has to briefly wander around aimlessly.

3. Planning is not necessary at every frame. The reactive system is responsible for executing the plan and monitoring the execution to check whether the plan's assumptions have significantly diverged from what is transpiring. Since planning was done with respect to a simplified model of the world, one of those assumptions was that the plan was only valid for a certain time period.

There is still much room for improvement in the above scheme. One of the problems is that limiting the maximum plan length can be too heavy-handed. The problem is that to guarantee real-time performance, we are constrained by the worst-case scenario in which planning fails after exhaustively searching all possibilities. On average, of course, this will not necessarily be the case. For "easy" problems in which many possible solutions exist the character might usually find a plan, even a long plan, very quickly. Therefore, a better solution is to exert more fine-grained control over the number of inferences performed. The new architecture we are developing allows us to control one inference at a time so that we can simply keeping searching until we find a solution, or run out of time.

The idea of having a separate reactive system and cognitive system can also be taken further. For example, even when a character does run out of time it is not necessarily the case that planning for the existing problem is henceforth useless. For example, if a character is trying to plan a path on a map and there is no pressing danger, it is legitimate for it to decide to do nothing in particular for a while as it "studies" the map. This kind of policy can be implemented by having the character's cognitive system act as a server. The idea is that the cognitive system toils away on its assigned task until it comes up with a plan of action, exhausts all possibilities, or is told that it should give up on the current problem. Until the server is able to come up with a plan, the character's behavior and real-time responses are governed by a reactive system, or a previous plan.

The reactive system also need not be a static entity. For example, it could try to learn to improve itself by observing the solutions to problems generated from the cognitive system. Over time its behavior would come to approximate the cognitive system, thus making it a far better stand-in should circumstances require.

Conclusion

As interactive entertainment attracts breathtaking levels of financial investment and makes an ever deepening cultural impact, it is inevitable that AI will play an important role in enhancing the non-player characters that inhabit virtual worlds. Previous work has already leveraged some cutting-edge AI research, but has still only barely scratched the surface of what's available. In spite of many obstacles, some of which we have elucidated upon, we still believe that existing and emerging work in building intelligent agents represents a vast untapped resource of useful new techniques for computer game developers.

References

[1] D. C. Dennett. *The Intentional Stance*. MIT Press, Cambridge, MA, 1989.

[2] J. Funge. *Making Them Behave: Cognitive Models for Computer Animation*. PhD Thesis, Department of Computer Science, University of Toronto, Toronto, Canada, 1998. Reprinted in SIGGRAPH 98 Course Notes #10, Orlando, Florida.

[3] J. Funge. *AI for Games and Animation: A Cognitive Modeling Approach*. A.K. Peters, 1999.

[4] J. Funge, X. Tu, and D. Terzopoulos. Cognitive Modeling: Knowledge, Reasoning and Planning for Intelligent Characters. In *Proceedings of SIGGRAPH 99*, Los Angeles, CA, August 11-13, 1999.

[5] J. Funge. Representing knowledge within the situation calculus using interval-valued epistemic fluents. *Journal of Reliable Computing*, 5(1), 1999.

[6] M. van Lent and J. Laird. Behavior capture: Motion is only skin deep. In *Lifelike Computer Characters '98*, October 1998.

[7] H. Levesque, R. Reiter, Y. Lespérance, F. Lin, and R. Scherl. Golog: A logic programming language for dynamic domains. *Journal of Logic Programming*, 31, 1997, 59–84.

[8] F. Lin and R. Reiter. Forget it! In Russ Greiner and Devika Subramanian, editors, *Working Notes of AAAI Fall Symposium on Relevance*, November 1994.

[9] J. McCarthy. Partial Formalizations and the Lemmings Game. Technical report, Stanford University, Formal Reasoning Group, 1995.

[10] T. Rabie and D. Terzopoulos, Stereo and color analysis for dynamic obstacle avoidance. *Proceedings of Computer Vision and Pattern Recognition Conference (CVPR'98)*, Santa Barbara, CA, June 1998.

[11] Steven Shapiro, Yves Lespérance, and Hector J. Levesque. Specifying communicative multi-agent systems. In Wayne Wobcke, Maurice Pagnucco, and Chengqi Zhang, editors, *Agents and Multi-Agent Systems — Formalisms, Methodologies, and Applications*, volume 1441 of *LNAI*, pages 1–14. Springer-Verlag, Berlin, 1998.