Towards Complex Reasoning Agents for Action Games

Flávio Soares Correa da Silva[1]
Tiago Motta Jorge[1]

[1]Universidade de São Paulo - Departamento de Ciência da Computação
{fcs,tiagoj}@ime.usp.br

Abstract

The applicability of artificial intelligence techniques to build "believable" characters in computer games is no longer questioned. The techniques that have been effectively applied in most cases, however, are quite simplistic if compared to the most sophisticated techniques produced by the AI research community. We advocate that more ellaborate techniques could bring advantages to game programming, not only with respect to the performance of the agents being built, but chiefly due to the possibility of allowing agent designers to focus on the appropriate level of abstraction.

We illustrate our arguments with a concrete example, namely the endowment of reasoning capabilities for agents in an action game, based on a simplified version of the ATMS.

Keywords: *Artificial Intelligence, ATMS, Action Games.*

1 Introduction

It has been long advocated in academia and industry that artificial intelligence can greatly improve the "believability" of characters in computer games, especially those characters not controlled by human players [1].

In many cases, however, the AI techniques employed in the design of agents for computer games is quite simplistic if compared to the most sophisticated techniques produced by the AI research community.

Taking into account that computer games usually attract extremely skillful programmers who know exactly what the goals of their projects are, we would not consider surprising that the inclusion of more convoluted AI techniques had little impact in the final behaviour of programmed agents. We argue, however, that these more sophisticated techniques can greatly improve the *process* of design and implementation of intelligent agents for computer games, chiefly due to the possibility of allowing agent designers and programmers to focus on the appropriate level of abstraction.

In this article we present an ongoing experiment to provide support to our view. We have designed a lightweight version of the assumption- based truth maintenance system [2] and used it to program decisions about actions for syntethic agents in an action game.

Our goals with this experiment are twofold: we want to prove that sophisticated reasoning architectures such as the ATMS can be employed to program agents even for action games with no losses in computational performance, as well as to demonstrate that one such reasoning architecture can release designers and programmers from having to deal with low level agent programming, thus making the design and revision of these agents more flexible and agile.

In section 2 we briefly review the programming action game GUNTACTYX, employed in this experiment. In section 3 we present our simplified version of the ATMS. In section 4 we show how this simplified ATMS can be employed to program the behaviour of agents in GUNTACTYX. Finally, in section 5, we present an additional discussion and some conclusions.

2 GUNTACTYX – A Programming Action Game

The goal of the GUNTACTYX programming action game [3] is to test scripts made by the players that encode the behaviour of agents. While in most games one has to actually play it in real time, in GUNTACTYX you program your team's behaviour – usually employing AI techniques - and then just watch them interacting in the game's environments.

There are three modes of play, each of which with different objectives: Fight, Soccer and Race. In Fight mode, warriors hold a weapon so they can shoot bullets or grenades. Health, energy and the other warrior's properties change according to some rules. The game includes three levels that are tailored for fights. The main purpose of this mode is to kill all the enemy teams. In Soccer mode, the objective of every team of warriors is to send the ball in the goal area of enemy teams. In Race mode, the objective of the warriors is to move around the origin in the counter-clockwise direction avoiding the obstacles of the level.

The scripts must be written in SMALL, which is a language that resembles C with a lot of simplifications. The game provides a function library that the scripts can import. This library includes all the necessary functions to interact with the environment. After compiled, one has only to copy its compiled script into a specified subdirectory in the game's installation directory to be able to chose it once the game is running.

The SMALL language was chosen because of the ease of manipulation by the interpreter. The game engine executes exactly the same number of instructions from each game bot script, thus making the simulation very fair and reliable.

In general, the GUNTACTYX programming game offers a very good environment for prototyping AI-based game scripts.

3 A Simplified Version of the ATMS

The assumption-based truth maintenance system [2] is a sophisticated system for belief revision [4], based on the identification of consistent subsets of possible worlds for a partially inconsistent logical theory. In its original version, it is also computationally complex, and thus hardly feasible for programming agents for action games that have to reason and act in real time about changes in the environment.

In this section we introduce a highly simplified version of the ATMS, which, for the conditions found in e.g. the GUNTACTYX, make it feasible to control the behaviour of agents.

Our system is based on three basic concepts: *assumption*, *observation* and *action*.

An *assumption* is a general statement about the world, e.g. “friendly troops under siege” or “friendly troops are at advantage”. For reasons that will be come clear in the following paragraphs, we assume that each agent must manage a relatively small set of assumptions (typically, less then ten assumptions). We denote the set of assumptions as *A*.

An *observation* is what results from the sensors of the agent. We denote the set of observations as *O*. We assume that there is a partial function $F: 2^O \to A$, that associates a set of observations to an assumption.

Finally, an *action* is what can be sent to the actuators of the agent to determine its behaviour. We denote the set of actions as *C*. We assume that there is a partial function $G: A \to 2^C$, that associates a set of actions to an assumption.

Considering the set of subsets of *A* and the subset relation, we have the natural lattice of subsets of *A*. We now assume that some subsets of *A* are deemed as *inconsistent subsets of assumptions*. By definition, we state that if a set of assumptions **U** contains an inconsistent set of assumptions **V**, then it is also inconsistent.

Each GUNTACTYX agent behaves based on a *sensing-deliberation-action* cycle. Sensing generates a set of observations **O**. The union of all *F(O):* {*O* belongs to 2^O} is a subset of *A*, which we denote as **A**. If all sensors of the agent were fully reliable and consistent with each other, then **A** should necessarily be a consistent subset of assumptions. This is not the case, however, and hence **A** may be inconsistent.

If **A** is inconsistent, we generate the maximal consistent subsets of **A**, denoted as $\mathbf{A}_1, ..., \mathbf{A}_n$. Deliberation in our model amounts to deciding which of these maximal consistent sets of assumptions is the currently valid one. In the initial experiment presented in this article, we simply pick one set at random. In future experiments we shall build more sophisticated decision procedures, based on preference relations between sensors.

Finally, given that the set of assumptions $\mathbf{A}_i$ is assumed as the present one, action amounts to determining and executing the set of actions **C** given by the union of all *G(A):* {*A* belongs to $\mathbf{A}_i$}.

The employment of this system to characterise the behaviour of agents in GUNTACTYX abstracts all low level control away from the agent designer, who now can focus on the design of the cognitive parameters that determine the desired behaviour of agents:

- the set of assumptions *A*;
- the subsets of *A* that are deemed to be inconsistent;
- the set of observations *O* and the function *F*; and
- the set of actions *C* and the function *G*.

4 A Running Experiment

We are just starting to experiment with this theory and the GUNTACTYX action programming game. In the mean time, we are studying knowledge bases in general and belief revision systems in particular. We expect to have more results and details by the time of presentation of this work at the Wjogos conference.

Our experiments are directed towards the Fight playing mode of GUNTACTYX. Sensing is related to positioning in the fighting arena of a bot, its allies and its enemies, to the running actions of the bot (e.g. whether the bot is running, walking or standing still, whether it is shooting or not, etc.) and to the present status of the bot (e.g. how many bullets it still has on stock). The assumptions encode the “interpretation” of the bot about what has been sensed, in terms as referred to in section 3. Finally, the actions are constrained to what GUNTACTYX bots can do.

5 Discussion

The construction of programs to control the behaviour of syntethic agents employing sophisticated AI techniques and abiding by the necessary efficient constraints related to real-timeliness and speedy reactions is a challenging programming task. These constraints are typically found in robotics and gaming.

We have presented an ongoing experiment to embed ATMS – a convoluted technique for belief revision – into an existing programming action game.

Bibliography

[1] J. E. Laird and M. van Lent. *Human-level AI's Killer Application: Interactive Computer Games.* AAAI. 2001.

[2] J. de Kleer. *An Assumption-based TMS.* Artificial Intelligence, v. 28(2). 1986.

[3] http://gameprog.it/hosted/guntactyx/

[4] http://www.beliefrevision.org