being-in-the-world

Mark A. DePristo

Hughes Hall
Cambridge University
Cambridge, England
mdepristo@cam.ac.uk

Robert Zubek

Computer Science Department
Northwestern University
1890 Maple Ave., Suite 300
Evanston, IL 60201, USA
rob@cs.northwestern.edu

Abstract

being-in-the-world is an intelligent agent capable of living autonomously in a Multi-User Dungeon world. In this paper we present a hybrid-architecture approach to building such an agent, discuss the successes and pitfalls of this technique, and potential improvements.

Introduction

The domain of Multi-User Dungeons (MUDs) presents an aspiring AI designer with an interesting set of constraints. The game environments are relatively simple, but also highly dynamic and fast paced, with many independent characters acting simultaneously. While the set of actions is usually small, the ontologies of objects in the world tend to be rather large and rich.

The combination of complex worlds and real-time dynamics makes for an exciting and enjoyable environment, but it complicates development of autonomous agents. Consequently, typical MUD agents fill functional or aesthetic roles only. Restricted to shopkeepers, village gossips, or inert background characters, autonomous agents are neither designed nor expected to live a life similar to that of player characters.

We decided to design an intelligent agent that, unlike traditional MUD agents, would live in a MUD as a player character. This agent would have access to the same sensory information, the same repertoire of actions, and be subject to the same survival requirements as human-controlled characters. Such an agent would need to cope in a world populated with independent, hostile agents, while maintaining its goals of survival and self-advancement.

This paper presents our work towards constructing such an agent. Our architecture, which draws from both symbolic AI and behavior-based robotics, illustrates the difficulties in trying to apply traditional symbolic AI approaches to such highly dynamic domains. We hope that the work presented in this paper can provide insight for artificial agent builders working in domains similar to MUDs, such as massive multi-user systems and role-playing games.

In the following paragraphs we present the architecture for the *being-in-the-world*, as well as details of the MUD environment. A later section will discuss the benefits and problems of our approach and, more generally, of implementing an intelligent agent living in a dynamic environment.

Related work

The problem of implementing artificially intelligent game agents and bots is as old as gaming itself, and the classic solution is to encode the action selection code as a simple state machine (Rabin 2000). For simple game agents this is often sufficient – as seen from the large number of existing Quake and Unreal bots. However, there is also work underway on using full-blown symbolic techniques, such as forward-chaining production systems (Laird 2000), to make game agents substantially smarter.

In MUDs and adventure games in particular, one notable related project is the Angband Borg, an automatic player for the Rogue-like adventure game Angband (Harrison 2000). The Borg serves as an aide that the player can use to perform routine tasks in the game, such as picking out the optimal combination of armor to wear, or automatically approaching the closest monster. The Borg plays the game from the player's point of view using their information about the world.

Architecture overview

Our agent, called *being-in-the-world*, is explicitly designed to be able to survive in a MUD. Survival requires at least the ability to navigate within the world, to maintain health, to avoid hunger and thirst, and to successfully interact with (potentially hostile) human

players and NPCs. A successful autonomous agent should also endeavor to collect gold and resources, purchase better equipment, and earn more experience to become a more powerful character.

These goals require the agent to possess reasonably sophisticated inference abilities. It must have a framework to understand the world in which it lives. It must integrate sensory information from the world into this framework. Further, it must be able to determine, based on its understanding framework and sensory information, the reasonable courses of action to survive and succeed in the game environment. Such abilities require a powerful and efficient inference system.

Survival in a real-time hostile world also implies a need for excellent real-time coping skills. Since we wanted the agent to figure out at runtime the strategies for living in the MUD, given some knowledge of the world and a few immutable survival goals, a pre-compiled inference network was not a sufficiently flexible solution. Without substantial limitations on the inference powers of the engine, we worried that inference speed would be inadequate to handle the real-time aspect of the MUD. Moreover, certain aspects of interacting with the environment, such as perceiving the surroundings or carrying out routine actions, did not conceptually belong in an inference mechanism. We thus decided to employ a separate, independent mechanism for coping with real-time world interactions.

These two constraints – of skillful coping with the world on one hand, and of logical inference on the other – suggested a natural division of the agent architecture into two systems, responsible for coping and thinking.

Hybrid architecture

being-in-the-world employs a two-level hybrid architecture: *Descartes*, the reasoning module, which includes the agent's internal state, world understanding, and goal maintenance, and *Heidegger*, the real-time coping module, which tries to satisfy the agent's most immediate goals while dealing with the world in a simple but timely manner. The modules run asynchronously and largely independently of each other. Communication between modules is accomplished through a queue of goals and a shared world ontology.

Descartes

The agent's planning layer is essentially a logic-based truth maintenance system and reasoning engine (Forbus and de Kleer 1993). Rules about the world resemble the following example:

```
(rule ((:TRUE (IS-A ?subtype
                    ?supertype))
       (:TRUE (INSTANCE-OF ?x
                    ?subtype)))
   (rassert! (INSTANCE-OF ?x
                    ?supertype)))
```

which, for instance, states that any object that is an instance of a subtype is also an instance of all supertypes. Then, if the agent would come across some item KNIFE123 that was an instance of a knife, and it knew that a knife is a subtype of a weapon, it would infer that KNIFE123 is also an instance of a weapon.

Descartes would then use the continuously updated knowledge of the world (provided by Heidegger) to decompose its high-level goals to a graph of simple, immediate goals. For example, Descartes understands weapons and money – that some weapons are better than others, that it needs gold to acquire them, that other agents have gold, and that the way to get that gold is to kill someone and loot their body. When the '(ACQUIRE-ITEM KNIFE123)' goal is activated within Descartes, the system would determine that:

- It needs gold to purchase the knife
- It is poor, so it activates the '(ACQUIRE-GOLD)' goal.
- That the '(ACQUIRE-GOLD)' goal can be satisfied by killing and looting.
- It knows that street sweepers are weak creatures, so it activates the '(KILL-CREATURE sweeper)' goal.

Because Descartes by itself cannot actually *do* anything, it communicates its conclusions to Heidegger, which then tries to meet Descartes' goals. The goals are expressed as concrete actions in the world, such as going somewhere, picking things up, attacking, and so on.

Heidegger

The coping layer is responsible for sensing and affecting the MUD world. It integrates the information from sensory inputs into the knowledge base, and carries out simple actions/goals in the world. It also includes closed-loop reactions triggered when the agent's survival is threatened and immediate action is required.

Heidegger copes with the world in the sense that it knows how to perform concrete actions in the world, and includes built-in reactions to events that cannot wait to be processed by the thinking layer. Thus, the coping layer knows how to get to locations in the MUD, how to pick up objects, or how to attack creatures, and performs appropriate actions per Descartes' request. However, in high-priority situations such as getting attacked, it will deal with the situation directly, ignoring the goals received from the upper layer.

Heidegger sieves the sensory information arriving from the MUD world, pulling out the relevant sensory inputs and communicating them to Descartes by updating the shared world ontology. For example, upon coming across a street sweeper, Heidegger would assert the following statements in the ontology:

```
(CRITTER 103 sweeper)
(EXISTS 103)
(WEAK 103)
```

which means that Descartes now knows that CRITTER 103 is a sweeper, that an instance of a sweeper exists somewhere in the world (Heidegger remembers where), and that this sweeper is weaker than the agent.

Heidegger's other task is to carry out simple commands requested by Descartes. There is a small set of commands that corresponds to what human users can do (such as LOOK, WIELD, HIT, GET, and so on), and a set of macro-commands that let the agent navigate about the world (such as GOTO-ROOM, WANDER, and so on). The commands (and they really are both goals and commands) are carried out opportunistically – that is, if something cannot be done at the moment, such as buying some item, it will remain on the queue in hope that it can be carried out later.

The MUD world

The environment in which *being-in-the-world* lives is an existing MUD server named ScryMUD (Greear 1999). We chose to use an existing MUD to minimize the temptation to over-engineer the environment – the only changes made to the original server were creating a machine-readable display mode and adding object IDs to the list of object properties visible to the agent. The agent communicates with the MUD like any other player character, via a text-based TCP socket connection.

Discussion

Our choice of using a hybrid architecture was motivated mainly by the application of hybrid systems in robotics, where they have been successfully used to combine low-level behavior-based networks with higher-level symbolic reasoning (see Arkin (1998) for an overview).

There are several clear advantages to hybrid architectures. The separation makes it possible to use vastly different architectures for the 'thinking' and 'coping' aspects of the agent, which is especially good given the system constraints – the need for inferential power in Descartes, and need for good reactivity in the Heidegger. It also allows the two layers to maintain a high degree of independence, which is critically important for Heidegger, who commonly must drop everything and react quickly to immediate dangers.

There are, of course, design choices and limitations to this architecture.

The main problem, which is common in hybrid systems in general, is that of the interface between modules. There appears to be no good way of interfacing the deliberative and the reactive systems. Our solution – world ontology updates going up from Heidegger and goal queue updates coming down from Descartes – reflects our conviction that the deliberative system should not direct but only suggest the possible course of action (it is admittedly influenced by the Agre and Chapman model (1990)). But the solution is somewhat ad-hoc – it is not clear what the implementation of a good, clean interface between the two should be.

Another set of difficulties arose due to our choice of architecture for the deliberative system. While the truth maintenance system was excellent at logical inference, a necessary feature for the deliberative layer, it lacked several critical features for modeling the agent's behavior. We encountered two serious limitations in our work. First, the system could not effectively model continuous quantities like the agent's store of gold and health points. Ideally, we should represent the amount of gold owned by the agent as an assertion like '(HAVE-GOLD *x*)', where *x* is some natural number. This representation was possible, but when the quantity of gold changed, the assertion would need to be replaced by another statement '(HAVE-GOLD *y*)'. The fact database quickly became overburdened with hundreds of such statements, one for each quantity of gold the agent ever owned. Worse, the system would maintain an entire truth graph from each such statement, consuming an enormous amount of resources. A future version of the system will have to include better support of continuously varying quantities – perhaps as qualitative numeric relations rather than direct numeric representations.

Second, and more seriously, we realized that we needed a better integration of facts and goals. When Descartes needed to communicate an action goal to Heidegger, it would do so by adding it to the shared goal queue. This communication occurred as a side effect of the triggering of an inference rule. However, this mode of communication became problematic in cases when some precondition of an inference toggled between being true and false – which caused invalidation and revalidation of all subsequent inferences. Goals were enqueued as a side-effect of certain inferences about the world, but our reasoning engine did not have much support for side-effects: due to the caching in the TRE the goals associated with the inference rule would not be enqueued again when the inference became true a second time. The next version of the system will include either a better layer interface or

an extended inference engine that better supports desired side effects.

We have implemented a simple working version of *being-in-the-world*, written in Common Lisp. The system, despite the above problems, is capable of exploring the MUD world, acquiring weapons, attacking and killing creatures, looting their bodies, and surviving – for a time.

Summary

In the end, we are very pleased with having chosen survivability as the primary goal in the design of this agent. It forced us to deal directly with the constraints of the environment, and to find ways of integrating deliberation and reactivity. Such an effective integration was crucial to solving the problems of implementing both high-level cognition and successful coping skills in a dynamic real-time environment. Unfortunately, the unforeseen problems with our truth-maintenance module prevented us from making the agent as robust as planned. Our future attempts will concentrate on improving the deliberative architecture and its interface to the coping mechanism. We will also explore minimizing the background knowledge necessary to bootstrap the agent, and adding rudimentary natural language understanding and modeling of other agents' goals.

Bibliography

Agre, P. and Chapman, D., 1990. "What Are Plans For?" *Robotics and Autonomous Systems*, Vol. 6, pp. 17-34.

Arkin, A., 1998. *Behavior-based Robotics.* MIT Press, Cambridge, MA.

Forbus K. and de Kleer, J., 1993. *Building Problem Solvers.* MIT Press, Cambridge, MA.

Greear, B., 1999. ScryMUD 1.9. http://scry.wanfear.com/

Harrison, B., 2000. The Angband Borg. http://www.phial.com/angborg/

Laird J., et al., 2000. The Soarbot Project. http://ai.eecs.umich.edu/~soarbot/

Rabin, S., 2000. “Designing a General Robust AI Engine”. In DeLoura, M. *Game Programming Gems*. Charles River Media, Rockland, MA.