Hierarchical Agent Interface for Animation

Armin Bruderlin and Sidney Fels and Silvio Esser and Kenji Mase
ATR Media Integration & Communication
Seika-cho Soraku-gun
Kyoto 619-02, Japan
email:armin,fels,esser,mase@mic.atr.co.jp

Abstract

Asynchronous, Hierarchical Agents (AHAs) provide a vertically structured multilevel abstraction hierarchy. In this paper, we argue that this multilevel hierarchy is a convenient way to create a human-agent interface at multiple levels of abstraction. In this way, the agent has several layers of *specification* (input) and *visualization* (output) which facilitates users with problem solving, because such an interface parallels the hierarchical and iterative nature of human creative thought processes. The AHA interface presents an intuitive, intimate interface which supports interactions on a scale from direct manipulation to delegation, depending on the user's choice. Another feature of this interface is its two modes of interaction: direct device interaction (mouse clicking) and interpretive, command line or scripting mode. This way, agents can be "forced" to perform certain activities via mouse clicks (direct control), or they can be programmed via scripts on the fly. We present examples of the use of the AHA interface from the domain of animating human-like agents in a virtual world.

Keywords: intelligent agents, creative process, motion control.

1 Introduction

In this paper, "agent" refers to an autonomous computer program which performs some task or tasks on behalf of an entity, communicates asynchronously with other entities in the world and responds asynchronously to events in the world. Entities may be people, other agents, other traditional programs or objects in the world. Our agents are structured hierarchically and respond asynchronously to events and are called Asynchronous, Hierarchical Agents (AHAs).

The term "autonomous computer program" implies that the program can run on any machine without intervention from a person. The "asynchronous" nature of the agent implies that it responds to events which occur as-they-happen rather than requiring a global synchronization signal. Of course, this does not mean that synchronous behaviour cannot be implemented by asynchronous agents, rather, synchronization must be made explicit where it is needed.

The term "hierarchical" means that the internal structure and knowledge of the agent is organized in several conceptual layers. These layers also correspond to the levels of interaction which the user has to communicate with such an agent. Our AHAs use a vertically layered approach [Müller *et al.*, 1994]. The main focus of this paper is to discuss the hierarchy with respect to the different levels and modes of interaction. The combination of multiple levels, which provide access to the activities of agents from low-level, direct manipulation to high-level, complete delegation, together with mouse-click and scripted interaction modes results in a flexible interface which represents a useful and convenient approach to real-time interaction with virtual characters in virtual worlds.

In the next section, related work is described. Section 3 explains the asynchronous agent hierarchy in more detail, and introduces the different modes of interaction within this hierarchy. Examples from animation of virtual worlds are given in section 4. Finally, section 5 gives conclusions and outlines future work.

2 Related Work

Our work on hierarchical agents continues the tradition of layered computation [Brooks, 1986] [Ferguson, 1992] [Bonasso *et al.*, 1995]. The combination of a layered approach and a belief, desire, and intention (BDI) architecture found in the InteRRaP system [Fischer *et al.*, 1995] [Müller *et al.*, 1994] most closely resembles our approach. Like [Fischer *et al.*, 1995], we take a pragmatic approach to the problem of designing an agent and do not propose a new theory for knowledge representation. In their model, the knowledge base is structured hierarchically with increasing knowledge abstraction. Likewise, agent control is layered functionally with the behaviour-based layer (BBL), the local planning layer (LPL) and the cooperative planning layer (CPL). Access from the control layers to the knowledge base maintains the hierarchy abstraction. Further, in the InteRRaP system, there is a world interface unit (WIF) which allows information flow to and from the world. In our system, we have a structure called the knowledge monitor which contains a knowledge base that is structured so that all control layers have access to the information contained; providing

access to knowledge that applies to all levels of control abstraction. Our hierarchical decomposition of knowledge is also similar in spirit to [Blumberg and Galyean, 1995], who propose a three-layer system to control the behaviours of Silas, a virtual dog.

One of the motivations for our design is to allow the agentee (agent user) to have control over the agent at different levels of abstraction. The philosophy is related to Norman's seven steps of human action [Norman, 1988]. The 7 steps are generally divided into goal, intention, action sequence, execution, perception, interpretation, evaluation. While we divide our agent's actions into different layers (see section 3.1 below), we are consistent in considering that the agent acts upon the world (and itself) and evaluates changes in the world (and itself) as a result.

3 AHA System

A block diagram of an Asynchronous Hierarchical Agent (AHA) is shown in Figure 1. The three fundamental aspects in the system are:

1. hierarchical agent structure;

2. knowledge monitor;

3. user interaction.

The remainder of this sections describes each of these parts in detail.
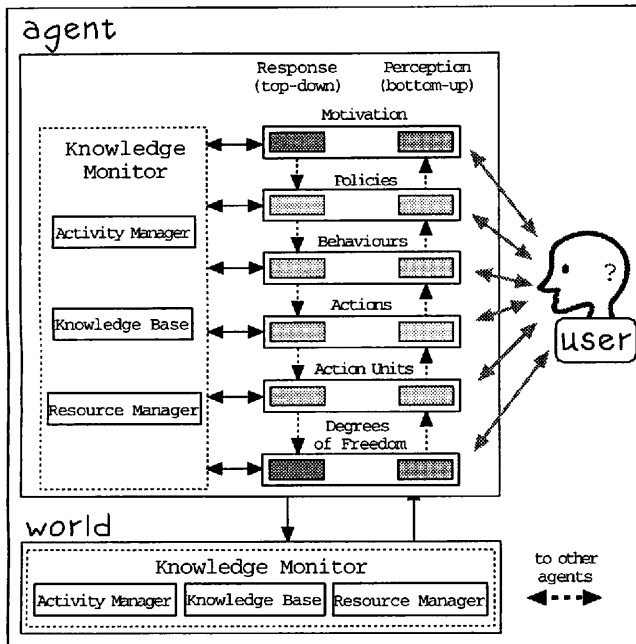

Figure 1: AHA structure.

3.1 Hierarchical Agent Structure

The multi-level hierarchical structure of the system is comprised of the following layers:

1. Motivation (needs);

2. Policy (plan);

3. Behaviour (goal-directed [sequences of] actions);

4. Action (automated sequences of atomic action units);

5. Action Units (sets of degrees of freedom);

6. Degrees of Freedom (effectors and sensors which interact with the world).

Each layer corresponds to a separate level of functional and conceptual activity inside the agent and communicates (through the knowledge monitor) to its neighbours. Typically, for responding to events which have occurred in the world, information flows from the top of the hierarchy to the bottom; eventually reaching the bottom layer which can act on the world directly. When the agent is trying to understand and perceive the world, information flows from the bottom of the hierarchy to the top. The dotted connections indicate logical data-flow. All real data flow occurs through the knowledge monitor.

3.2 Knowledge Monitor

The second component is called the *knowledge monitor* and is made up of three parts: the *knowledge base* which contains global variables and knowledge shared between layers; the *resource manager* which administers the co-ordination of resources; the *activity manager*.

The activity manager is implemented using a client/server model. The server keeps track of three main types of information specified by clients (layers in hierarchy); first, an activity list containing all the activities which are concurrently happening; second, events which occur (i.e. starting or stopping of activities); third, a list of all the perceptual filters watching for events and activities to occur. A filter is specified by a client as a four-tuple of the following form:

```
{ start       {running activity      { stop
  activity}     expression}            activity}

              --> {command to execute}
```

When *all* the conditions specified in the filter are true then the filter triggers the command associated with it. Here are two examples:

```
{raining} {outside & have_umbrella}    { }
               --> "start use_umbrella"

   { }        {use_umbrella}         {raining}
               --> "stop use_umbrella"
```

As seen in Figure 1, the knowledge monitor structure is also used to create a knowledge monitor for the world. This knowledge monitor is used *stand-alone* to maintain the world state and keep track of events which occur in it.
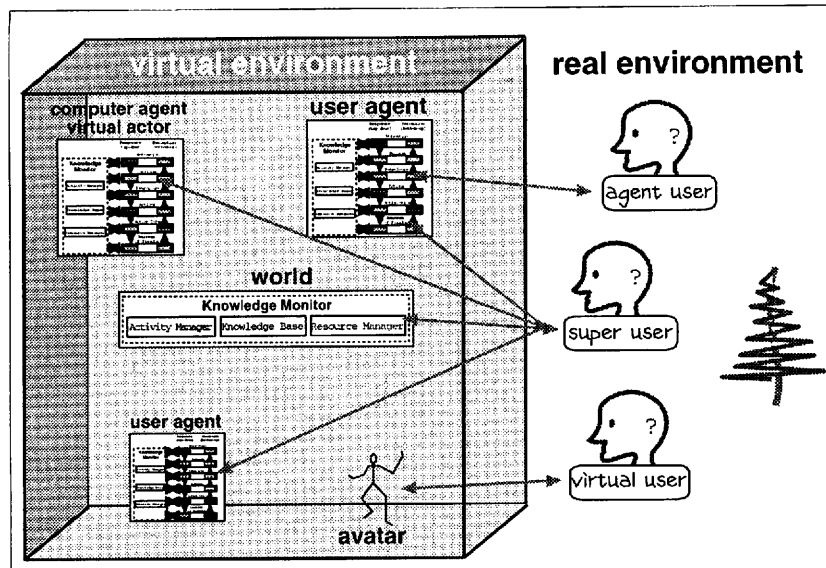
Figure 2: Interactions between users and virtual environment.

3.3 User Interaction

When agentees interact with an agent or virtual character (see Figure 1), the different layers of abstraction provide a convenient decomposition related to their own way of thinking. Abstraction is an important concept for designing an agent interaction process: it allows for the convenient creation and modification of the agent by the agentee at different levels of abstractions or granularities, enabling the viewing of a "tree" without knowing about "forests", and the examination of a "forest" without distraction of the details of the "trees". We can consider the control of virtual characters[1] a complex synthesis task similar to the composition of a musical piece or the design of a new product. Such creative processes are inherently hierarchical, iterative, and make use of alternate views of the problem to find a solution [Simon, 1969]. Koestler notes that the act of discovery or creation often occurs when distinct representations are recognized as depicting the same object, idea, or entity [Koestler, 1990]. Thus, by having both, levels of knowledge and levels of interaction represented in this hierarchical framework, users can choose the level of detail with which to specify and visualize at any moment, depending on their current state of mind.

A more complete illustration of interaction with agents in a virtual environment is shown in Figure 2. Three types of users can be distinguished: the *agent user*, who is represented in the virtual world by his/her *user agent*, i.e. an Asynchronous, Hierarchical Agent as discussed in the last section, which can autonomously carry out tasks or be controlled at multiple layers of abstraction; the *super user*, who can access anything in the virtual world, e.g., he can initialize the world, user agents, as

[1]This holds, in general, also for any complex computer program, or a program in a complex environment.

well as *virtual actors*, which are agents without agent users. Finally, there is the *virtual user*, who has no agent support, but is represented by an *avatar* whose actions are completely controlled by its virtual user at any time. We are currently working on implementing such a system based on the AHA architecture. Examples are given in the next section.

4 Examples

Figure 3 shows an example of an agent user interacting with his user agent A via the multi-layered interface. Agent A has chosen the indicated motivations, policies, behaviours, etc. and its agent user has decided to make A "kick B" this time by interactively setting A's behaviour. Likewise, an agent user might choose low-level, direct manipulation control by bending the knee of his agent a bit more.

Currently, our AHA hierarchical agent/interface project is still at a developing stage, and we do not yet have a general system to control human-like agents as illustrated in Figure 3. However, we have implemented the basic AHA framework, and are now testing it in the simplified scenario of a board game, called the *Malefitz* game [Esser et al., 1997]. Figure 4 shows a snapshot of the game in progression. There are four players (with four pieces each), each implemented as as an Asynchronous Hierarchical Agent. The players know the rules of the game and act autonomously or with respective agent user interactions with the layers. For instance, player A might have the motivation "winning the game" which triggers the policy "aggressive" which in turn activates the behaviour "kick out player B" (after the proper die roll) which will result in the two actions of player B being sent home and player A being positioned where player B was. However, the user agent for A might decide instead not to kick out B but advance through a dif-
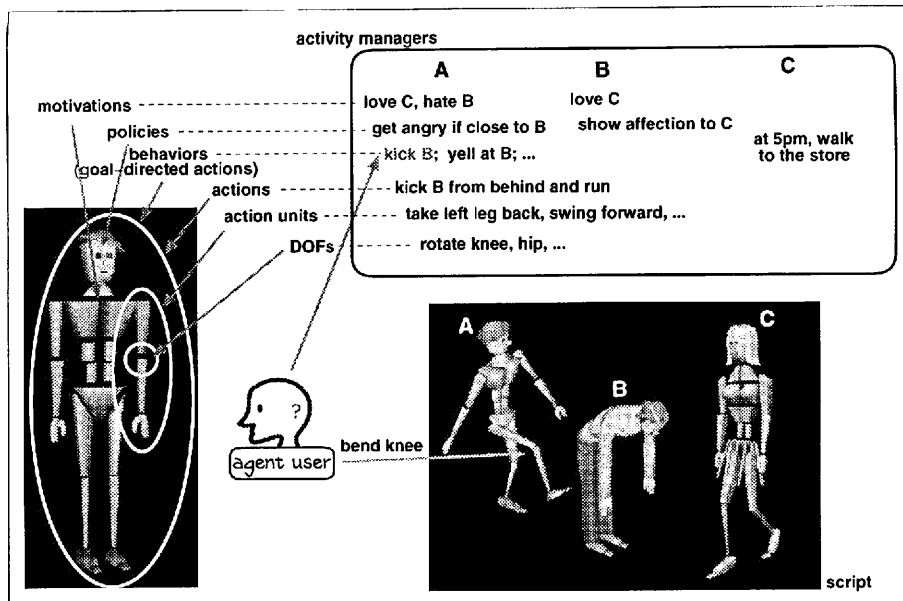
Figure 3: Multi-level interaction via scripts and direct manipulation.

ferent path, by overwriting the policy "aggressive" with "advancing". There might also be a virtual user participating in the game, i.e. a user without agent support who has to move his/her pieces "by hand" with a set of proper move commands.
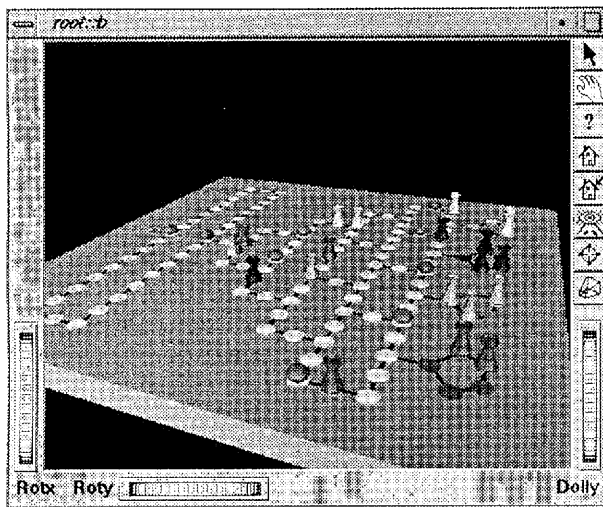

Figure 4: Snapshot of AHA Malefitz game.

Another current research effort towards a general AHA system to control human-like agents in virtual environments is focusing on implementing parameterized low-level, believable behaviours, elementary motions (like kicking, walking, running, knocking), as well as tools for customizing and personalizing, blending and overlapping these elementary motions. Figure 5 shows an example of our interactive human locomotion system [Bruderlin and Calvert, 1996], where the agent

user can interactively specify certain parameters like speed, step length, amount of arm swing, torso tilt while the walking or running human-like agent reflects these changes. This is an example of direct manipulation by dragging sliders to affect changes in motion. We are currently looking into "hooking up" the walking and running algorithms into the AHA structure, so that these movements can be called via scripts and the personality or style of the motion (i.e. the value of the sliders) is set by the motivation and policy layers.

5 Conclusions and Future Work

We have introduced a framework for asynchronous, hierarchical agents and discussed its relevance from the perspective of multi-layered human-agent interfaces. We believe that such an interface supporting multiple levels of abstraction facilitates interactions with agents in a virtual environment, as users can choose their level of involvement at any time to suit their thinking.

After successful demonstration of the basic principles of the AHA architecture within the Malefitz game, we are looking into providing a more complex example, such as animating a crowd scene, and creating personal information and navigation agents for visitors to our Meta-Museum project. We also want to incorporate learning mechanisms into our hierarchy, so that agents will improve or adapt their actions over time based on user input and the actions of other agents.

References

[Blumberg and Galyean, 1995] B. M. Blumberg and T. A. Galyean. Multi-level direction of autonomous creatures for real-time virtual environments. In SIG-GRAPH 95, pages 47–54, August 1995.
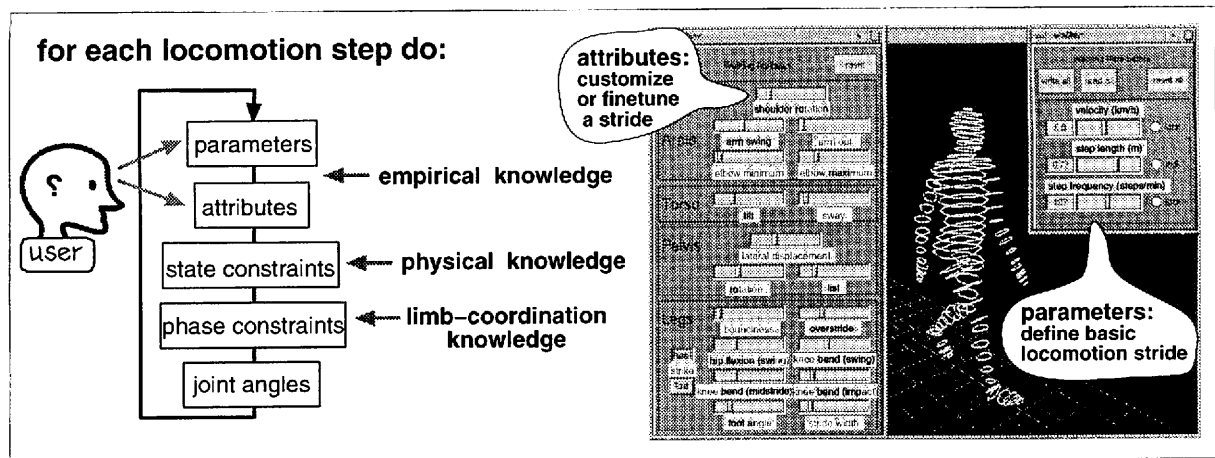
Figure 5: Direct manipulation of parameters for human locomotion.

[Bonasso et al., 1995] R. P. Bonasso, D. Kortenkamp, D. P. Miller, and M. Slack. Experiences with an architecture for intelligent, reactive agents. In M. Woolridge, J. P. Müller, and M. Tambe, editors, Intelligent Agents II, pages 187–202. Springer, 1995.

[Brooks, 1986] R. A Brooks. A robust layered control system for a mobile robot. IEEE Journal of Robotics and Automation, 2(1):14–23, April 1986.

[Bruderlin and Calvert, 1996] Armin Bruderlin and Tom Calvert. Knowledge-driven, interactive animation of human running. In Graphics Interface '96, Proceedings, pages 213–221, May 1996.

[Esser et al., 1997] S. Esser, S. Fels, A. Bruderlin, and K. Mase. Inventcl: A 3d mega-widget using open inventor and tcl/tk. submitted for publication in Usenix, Tcl/Tk workshop, 1997.

[Ferguson, 1992] I. A. Ferguson. TouringMachines: An Architecture for Dynamic, Rational, Mobile Agents. PhD thesis, University of Cambridge, UK, 1992.

[Fischer et al., 1995] K. Fischer, J. P. Müller, and M. Pischel. A pragmatic bdi architecture. In M. Woolridge, J.P. Müller, and M. Tambe, editors, Intelligent Agents II, pages 203–218. Springer, 1995.

[Koestler, 1990] G. Koestler. The Act of Creation. reissued edition, Arkana, 1990.

[Müller et al., 1994] J. P. Müller, M. Pischel, and M. Thiel. Modelling reactive behaviour in vertically layered agent architectures. In M. Woolridge and N. R. Jennings, editors, Intelligent Agents, pages 261–276. Springer-Verlag, 1994.

[Norman, 1988] D. Norman. The Psychology of Everyday Things. Basic Books Inc. New York, 1988.

[Simon, 1969] H.A. Simon. The Sciences of the Artificial. MIT Press, Cambridge, MA, 1969.