Aspects of interactive autonomy and perception

Madsen, Claus B.; Granum, Erik

Published in:
Virtual interaction : interaction in virtual inhabited 3D worlds

Publication date:
2001

Document Version
Accepted author manuscript, peer reviewed version

Aspects of Interactive Autonomy and Perception

Claus B. Madsen and Erik Granum
Laboratory of Computer Vision and Media Technology
Intermedia Aalborg
Aalborg University
Denmark

June 29, 2000

Contents

1 Introduction

The purpose of this paper is to approach the issue of autonomous agents from a more technical, pragmatic and concrete point of view than that of the other papers in the book. Especially this paper offers a contrasting perspective compared to the preceding paper "Agents as Actors" by Andersen and Callesen.

Andersen and Callesen use concepts from fields such as literature, film, theatre and language theory to set some requirements for the design, or the architecture as it were, of autonomous agents. In this context it can be termed a *top-down* approach to agent design. As Andersen and Callesen point out, others have used inspiration from biology and ethology to design autonomous agents. Again a top-down approach.

The present paper takes a *bottom-up* approach. In effect we have set out to create some concrete examples of interactive autonomous agents in order to, from experience, "learn" how they can be designed. Especially how practical issues inform the design of the agent architecture, the elements in it, and how they interact.

This is not to say that our work with autonomous agents is not based on any philosophy. But the philosophy is quite simple: the agents must be truly autonomous, capable of relating in a sensible manner to all aspects of a seamless, continuous, simulated virtual world. That is, the agents must relate to the spatial properties of the virtual world, the dynamics of it, and the events that take place in it, in a continuous and consistent manner. Here we mean as continuously as can be implemented on a computer, and consistent means not violating any expectations concerning the structure of time and space which the simulated world suggests in the observer.

What we are aiming at is a unified framework for how an agent can relate to (perceive and act on) a simulated world. That is, we are not willing to accept any kind of special cases. For example, a sheep agent and a troll agent should be based on exactly the same formalism, only "running different scripts" (playing different roles). Additionally, the mechanisms with which an agent relates to all objects in the simulated world should be the same,

independently of whether the object is a house, another agent, or a real human represented by an avatar. I.e., an agent should perceive and interact with an avatar exactly as it would with any other agent. No special cases.

Franklin & Graesser (1996) formulate an autonomous agent definition which is generally accepted within the community:

> An autonomous agent is a system situated within, and a part of, an environment, which senses that environment and acts on it, over time, in pursuit of its own agenda, and so as to affect what it senses in the future.

In addition to this definition, Franklin & Graesser (1996) present a set of properties which can be associated to autonomous agents, according to their functionality. The agents we are interested in have three properties from this set, i.e., an agent must: 1) be proactive – the agent does not simply act in response to the environment, they should also initiate actions on their own accord, 2) be communicative – the agent communicates with other agents (possibly including people), and 3) have character – the agent has believable "personality" and emotional states.

This paper by no means describes a completely developed agent architecture which lives up to the above goals. Rather this paper can be viewed as a status report on ongoing work towards creating an architecture for a flexible autonomous agent, and a simulated virtual world in which the agent can live. We have looked in some depth at a number of issues relating to the design of autonomous agents, and this paper attempts to present some of the experiences we have made, and how these experiences have driven our efforts.

To forego one of the messages of the paper, it has turned out that much of our work thus far has focused on the perceptual system of agents. In fact we have discovered how important it is to have a thoroughly designed perceptual functionality. There may be two reasons for this:

- All people behind the work in this paper have some background in computer vision (computerized analysis of video images), and the application thereof for controlling robots.
- An autonomous agent will by its very nature run some kind of continuous sense-plan-act loop. Thus sensing capabilities are important, and in some manner come before much else can be investigated.

The latter issue seems to be inherent in the study of autonomous agents, whereas the former serves as information to the reader of the particular background we have for working with this problem. For years we have been studying vision controlled robots interacting with the real world; now we have taken an interest in working with robots with personality, living in a virtual world.

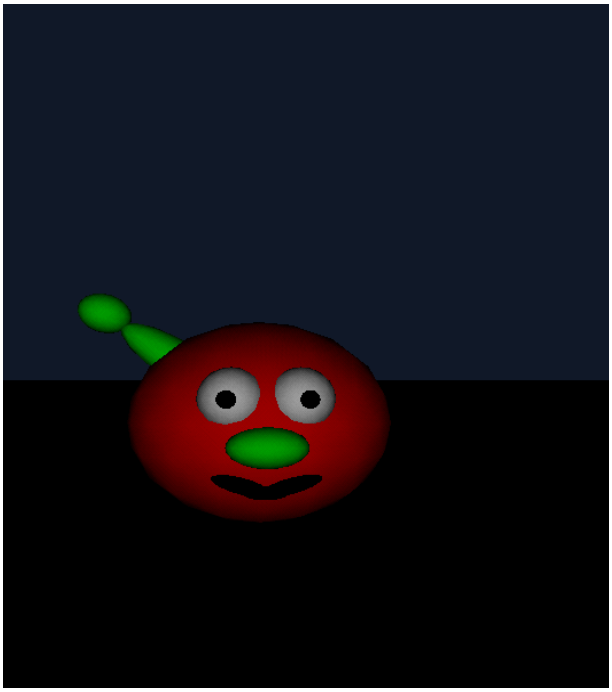

Figure 1: Bouncy, an interactive dog-like bouncing ball, is a prototypical example of an interactive autonomous agent.

The structure of the paper is as follows. First we describe a particular autonomous agent in some detail. The agent is a character called Bouncy, figure 1, and it was our first attempt at designing and implementing autonomous agents, (Madsen, Pirjanian & Granum 1999). In section 3 we then describe some alternative formalisms, or architectures, for agents. Based on the experience gained from designing Bouncy, and the discussion of alternative architectures, we argue for dividing agents into two layers: a high and a low level layer, or a subjective and an objective layer. This is discussed in section 4. From here on the paper focuses on the low level, objective layer, and presents in some depth our work with issues such as agent perception, sensory input and (spatial) memory, section 5, and the use of memory for spatial reasoning, section 6.

2 Bouncy – a simple interactive agent

The main purpose of designing and implementing the Bouncy character was to gain insight into the computational models for autonomous agents, capable of partaking in believable and engaging interaction with humans (and other agents). Thus the main focus behind the design of Bouncy was *interaction*, and the entire character, or role, of Bouncy was centered around a carefully planned interaction.

User's behaviour		Bouncy's perception of user
glove input	microphone input	
none	shouting	"Master is calling me"
pointing	shouting	"Master is scolding me"
open hand	talking	"Master is petting/comforting me"

Table 1: The table show the relationship between what the user does and how Bouncy interprets the intention of what the user does.

Subsequently we describe the design of Bouncy focusing on the control of the agent's innate behaviours. Specifically how the notion of time plus perceived stimuli affect the internal state of the agent, which in turn affect agent behaviour. Bouncy is implemented as a 3D animated character. Bouncy has been demonstrated to hundreds of people, and though conceptually simple, he has been wholeheartedly accepted by people in all age groups.

2.1 Bouncy scenario and interaction

The interaction scenario comprises Bouncy in a simulated 3D virtual environment, and a single human user in the real world. Bouncy moves around by bouncing up and down (hence the name Bouncy). He is able to express various emotions (happy, sad, etc.) by actuating his mouth, eyes, tail, and by the intensity with which he bounces.

Interaction with Bouncy is facilitated by several interface modalities. Bouncy is displayed to the user on a graphical display. Bouncy also produces a "bounce sound" while bouncing. Conversely, the user can communicate himself to Bouncy through a microphone and a data glove, which is used to characterize the posture of the user's hand. The actions of the user are perceived by Bouncy as described in table 1.

2.2 Bouncy behaviour repertoire, control and personality model

Bouncy has three main behaviours, one of which is further divided into three sub-behaviours:

- **PLAY**: causes Bouncy to wander about and play
- **SLEEP**: causes Bouncy to cease any current activity, lie down and sleep
- **INTERACT**: causes Bouncy to go to, and follow, the user plus engage in interaction. Sub-behaviours:
  - **tease**: pay attention to user but do not approach
  - **please**: approach user, bounce, wag tail and smile

– **have-the-blues**: stop bouncing, break eye-contact, look sad

These behaviours constitute the states in a Finite State Automaton, figure 2, in which state transitions occur based on events generated in Bouncy's personality model. The above description of Bouncy's "states" clearly illustrate, that the behavioural repertoire is very limited. In fact it should be quite easy to picture the range of the interactions one can have with Bouncy.

Typically, Bouncy will be bouncing around, playing, and when the user calls him, he approaches. Then if the user speaks loudly, while pointing his finger, Bouncy will gradually become sad and enter the **have-the-blues** behaviour. If the user then speaks while keeping an open hand, Bouncy will enter the **please** behaviour again. If the user does not do anything, Bouncy will after a while get bored, and enter the **play** behaviour, thus move away from the user (away from the screen) and start playing. At any point in time, Bouncy may become tired enough to go to **sleep**, a behaviour from which he wakes up after a while.

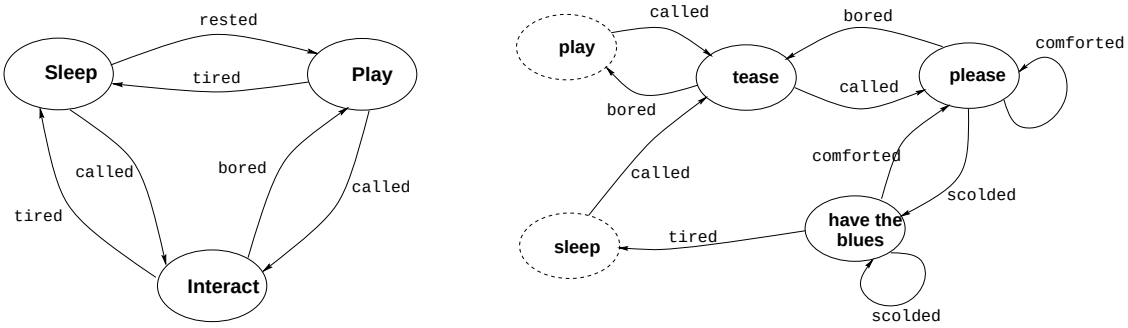

Figure 2: Left: the super-ordinate Finite State Automaton (FSA) of Bouncy, and right: the FSA including the sub-states of the **interact** state.

The overall behaviour of Bouncy, or its dramaturgic frame, is fully described by the FSA in figure 2. Yet, the mechanism for determining when and why to switch from one behaviour to another is a different matter all together. In all autonomous agents this mechanism is related to the personality and the mood of the agent. These concepts are normally characterized by their static/dynamic nature:

- Personality – describes the static traits of a character, and undergo little or no change over time
- Mood – describes the current mental state of the character at any point in time, and responds directly to changes in the character's situation

Concerning personality traits work on autonomous agents often borrow from psychology. An often used taxonomy is the socalled five-factor model, (McCrae & Costa 1989).

	excitedness	sleepiness	hapiness
rested		low	
tired	low	**high**	low
bored	low		
comforted	low	**not low**	
scolded		high	**low**

Table 2: Generating internal events from qualitative characterizations of the combined state of Bouncy's internal mood variables. The events drive the behaviour FSA of Bouncy, see figure 2. The bold-faced entries indicate that the value of the corresponding variable is weighted higher than the other variables, i.e., is the determining factor.

The model comprises extroversion, agreeableness, conscientiousness, neuroticism and openness. While it can be useful to apply these factors in characterizing an agent, it is more doubtful whether they map well onto a computational scheme. Autonomous agents that make extensive use of this are yet to be seen, although some attempt at using personality traits explicitly in autonomous agents is demonstrated by Silva, Siebra, Valadares, Almeida, Frery & Ramalho (1999).

On the other hand an often used computational model for an agent's mood is a set of numeric variables, which change values over time. In the design of Bouncy we used three such variables: excitedness, sleepiness, and happiness. All three variables range between -1 and 1. For example a combination of excitedness being 1 and happiness being -1, would correspond to an agitated, angry Bouncy. Bouncy does not have such a state, and thus this can be characterized as a personality trait of Bouncy: it cannot be angry. Instead scolding makes it sad, i.e., it enters the have-the-blues state.

The FSA of Bouncy is controlled by events generated based on the values of the three described mood variables, table 2. An event corresponds to "how Bouncy feels". For example, referring to the table, if excitedness is low, happiness is low, and sleepiness is high, then a **tired** event will be generated. The higher excitedness and happiness are, the higher sleepiness needs to be, in order to generate the event. Depending on the current state of the FSA (figure 2), the event will trigger a jump to the **sleep** state.

Naturally, the mood variables must change values over time. Table 3 summarizes what happens to the mood variables as a function of time and user action, and as a function of what the current state of the FSA is. Thus, if the current state is **sleep** the excitedness variable will gradually be reset to zero, as will happiness, i.e., the **sleep** state is used to "neutralize" Bouncy's mood. And of course sleepiness is gradually reduced as a function of time. Also, if in the **sleep** state, any sound from the user will cause an increment in the excitedness. This is used to make it possible for the user to "wake up" Bouncy when he is sleeping, because the **rested** event is also influenced by the value of the excitedness variable, table 2. The higher the excitedness, the lower sleepiness needs to be in order to

	Sleep		**Blues**			**Play**		**Please**		
	sound	time	scold	pet	time	sound	time	scold	pet	time
excitedness	↑	→ 0	↓			↑	↓		↑	
sleepiness		↓			↑		↑			↑
happiness		→ 0	↓	↑	↓		→ 0	↓	↑	

Nomenclature

Increment	:	↑
Decrement	:	↓
Reset	:	→ 0

Table 3: Qualitative schemes for incrementing/decrementing Bouncy's internal mood variables as a function of time, and as a function of the user's actions (sound, pet = sound plus open hand, scold = sound plus pointed finger). The **tease** state has been omitted to make the table less wide.

generate a **rested** event).

To summarize the behaviour control of Bouncy:

- time and user actions influence Bouncy's internal mood variables
- various combinations of mood variable values generate events corresponding to "how Bouncy feels", e.g., rested, comforted, etc.
- the events trigger state changes in the overall FSA, causing Bouncy to switch behaviour
- the current state determines how time and user action influence the change in the mood parameters

While the use of an FSA may seem to result in very abrupt changes in Bouncy's behaviour, the user does not really notice these changes, as Bouncy's facial expression is linked to the exact values of the happiness and excitedness, figure 3.

2.3 Extensions to the original Bouncy concept

In the above we have described the initial version of the Bouncy agent. Later work involved additions to the basic concepts, in terms of further developing Bouncy's ability to perceive the intentions of the user. Specifically, a speech processing module was developed which enabled an analysis of the pitch and rate-of-speech of the user, in order to determine of the user utterance was intended as "approval" or "disapproval", (Brøndsted, Nielsen & Ortega 1999). Sofar no actual speech recognition (speech to text) module has been developed for Bouncy.

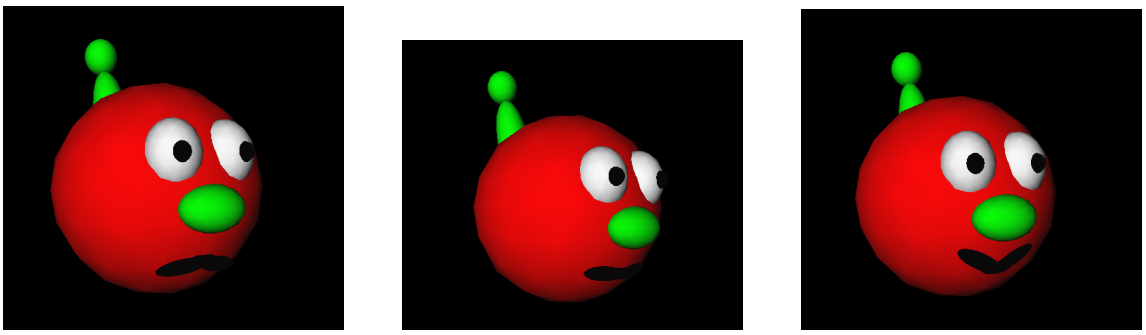

Figure 3: The facial expression of Bouncy is linked directly to the happiness and excitedness variables, and thus changes smoothly.

Additionally, we have worked on extending Bouncy's ability to act in virtual world with obstacles. The previously described behaviours affect the agent's "physical" degrees of freedom, in a manner where they do not interfere with each other. E.g., Bouncy can smile both while bouncing and at rest. Yet there are situations where a compromise has to be made between what two or more behaviours dictate. E.g. when trying to approach the user (user's avatar), but the direct path is obstructed by an obstacle.

To solve this problem we have designed a formalism, Multiple Objective Actions Selection, (Pirjanian 1998), for simultaneously satisfying the demands of several parallel behaviours requiring control of the agent's degrees of freedom.

Thus, Bouncy can act in an environment with obstacles. Furthermore, we have experimented with flocks of Bouncys, where one Bouncy acts as leader dog, and the others merely follow, (Pirjanian, Madsen & Granum 1998). Because they all are independent autonomous agents, follow-dogs may be forced to take alternative paths in order to avoid obstacles, but they will always attempt to catch up with the leader.

3 Alternative autonomous agent architectures

In the previous section we described the architecture behind the Bouncy interactive agent. Two elements were central in this architecture: a small set of numerical mood variables, and a Finite State Automaton for modelling the overall behaviour of the character. While the use of numeric mood parameters is standard in all autonomous agent, (though their meaning may vary), there are alternative architectures. Two such alternatives are described in the subsequent sections.

3.1 The Blumberg agents architecture

An agent architecture has been developed by Blumberg (1997) with the explicit purpose of creating interactive characters for entertainment and educational purposes. The archi-

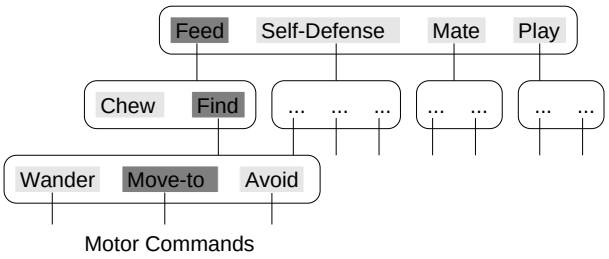

Figure 4: The behaviours in the top level group compete for the control of the character at every time step. The winning behaviour may have a child group. Inside this child group all behaviours are allowed to compete. This continues all the way to the bottom of the tree. In this example the top level behaviour feed has won (shaded dark), allowing chew and find to compete, where find has won, allowing wander, move-to, and avoid to compete, and among those move-to has won.

tecture has many similarities to the FSA approach of Bouncy, only instead of an FSA, Blumberg's agents employ a hierarchical tree structure of behaviours the agent may engage in. It is the task of each behaviour to determine its "Level of Interest" at every point in time. An action selection mechanism then chooses which behaviour is most appropriate, figure 4. Behaviours are organized in behaviour group, and within such a group all behaviours are considered equally important. Thus, all behaviours in a group compete.

The computation of the Level of Interest of a behaviour is primarily based on values produced by something called Releasing Mechanisms, (RM). An RM filters the agent's sensory input so as to search for objects or events that help determine the appropriateness of a behaviour. Thus, an RM is object/event dependent and output a continuous value. There are three primary phases to producing such an RM output value:

1. find phase: search through the sensory input for the presence of a particular object or event

2. filter phase: determine whether the object or event satisfy some requirements

3. weighting phase: determine the output value based on some criteria

An example RM could be a "mouse detector" used for triggering the feed behaviour of a cat, which only likes white mice: 1) sift through the sensory input to see if there are any mice around, 2) if there is a mouse, determine if it is white, and 3) dependent on the distance to the white mouse compute an output value. The output value of such an RM is combined with the values of the agent's internal mood variables, for example hungriness, to produce the Level of Interest for the feed behaviour.

The description given here of the Blumberg agent architecture is quite simplified, as it contains many subtle aspects. Yet, the important issues are the hierarchical organization

of competing behaviours, where the competition is based on computations of behaviour Level of Interest, which in turn is based on output values from Releasing Mechanisms.

3.2 The JAM agent architecture

The Finite State Automaton approach of Bouncy, and the agent architecture of Blumberg, have their roots in behaviour-based systems, and the reactive systems thinking, which not at least Rodney Brooks has advocated for many years, (Brooks 1986, Brooks 1991).

The JAM agent architecture, (Huber 1999), has its roots in more traditional AI, operating with explicit goals and plans, and reasoning about pre- and post-conditions of plans. The JAM architecture is shown schematically in figure 5. The world model is a database that represents the current beliefs of the agent. The plan library is a collection of plans that the agent can use to achieve its goals. The interpreter is the agent's "brains" that reason about what the agent should do and when it should do it. The intention structure is an internal model of the goals and activities the agent currently has and keeps track of progress the agent has made towards accomplishing those goals. The observer is a light-weight plan that the agent executes between plan steps in order to perform functionality outside of the scope of its normal goal/plan-based reasoning, e.g., update mood variables and process sensory input.

Changes to the world model or posting of new goals trigger reasoning to search for plans that might be applied to the situation, (this list of plans is called the Applicable Plans List, or APL). The JAM interpreter selects one plan from the APL and *intends* it, i.e., commits itself to execute the plan. The act of intending the plan places the now-instantiated plan onto the agent's intention structure, where it becomes one of possibly several intentions. The newly intended plan may or may not be immediately executed, depending on the plan's utility relative to that of the intentions already on the intention structure.

4 Dividing agents into layers

In the previous two sections we have described three different agent architectures: the Finite State Automaton (FSA) approach of Bouncy, the Blumberg agent architecture, and the JAM agent architecture. Through various projects, and over a period of approximately one year, we have accumulated experience with: 1) the FSA approach from the work with Bouncy, 2) the Blumberg architecture from implementing three specific agents using this architecture, and 3) designing and implementing a specific agent using a home-grown, less formal architecture not described here.

During this work we have made some observations that have caused us to start specifically studying the interplay between an agent's perceptual system, and its use of a memory for reasoning about the spatial properties of the virtual world. In fact we are currently working from the hypothesis that an agent can be divided into two layers: a subjective

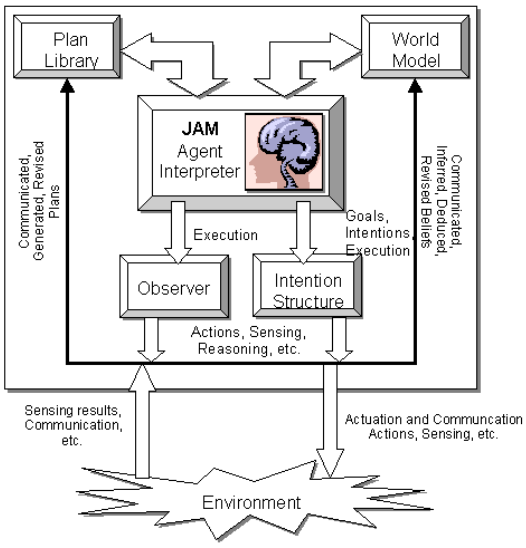

Figure 5: Schematic overview of the JAM agent architecture.

and an objective layer, where the objective layer is the current focus of our activities. In the subsequent section we shall attempt a brief presentation of the gist in the observations that have driven us onto that path.

4.1 Scripting of behaviour in agent architectures

Independently of which basic architecture is used for an agent the design of a *particular* agent involves writing little "scripts", or pieces of stories in which the agent can be involved. These script pieces are the behaviours, and the collection of behaviours in some manner defines the role which the agent can play, or its personality. In relation to this discussion there are two elements to designing such behaviours:

- a characterization of what triggers a certain behaviour, for example a way for the agent to detect that a certain situation has occurred in which a particular behaviour is relevant
- a careful definition of what the behaviour entails, i.e., what the agent should *do* in order to exhibit or communicate this behaviour

Using this point of view the main difference between the three described agent architectures lies in the triggering of behaviours. In the FSA approach of Bouncy, the triggering is done using characteristic sentiments defined from combinations of the mood variables. In the Blumberg architecture triggering is done via the Releasing Mechanisms and their

influence on each behaviour's Level of Interest. In the JAM architecture each plan has a set of pre-conditions that must be met for the plan to be relevant.

In principle every computer program can be modelled as an FSA, and thus it cannot be argued that the JAM architecture nor Blumberg's architectures can realize agents, which an FSA approach cannot. Nevertheless, there are some valid arguments against using an FSA as the underlying design tool for agents that are more complex than the Bouncy example.

One way to think about this is the following. Blumberg and JAM agents can be considered as FSA's where there is a potential connection from (almost) any state to (almost) any other state. This means that the resulting agent has more freedom to switch between behaviours. This gives the Blumberg and JAM agents a higher chance of emergence, i.e., an overall behaviour of the character, which is more intriguing, unexpected and multi-faceted than with a strict FSA. Something other than what the designer expected or explicitly planned, but nevertheless in accordance with the character's personality. Thus the latter two approaches are more flexible and extendible: additional behaviours can be added to the personality of an agent, without explicitly designing how the new behaviours interact with the behaviours already present.

One way to summarize the above is to say that an FSA based design emphasizes strict scripting, by defining what the agent *should* do in a certain situation. The Blumberg and JAM agent designs emphasize emergence, by focusing on what the agent *could* do.

Nevertheless a combination of strict scripting on one hand and the chance of emergence on the other is to be preferred. Therefore scripted, fixed linear sequences must be built into Blumberg or JAM agents. This is because there sometimes is a need for being able to *guarantee* a fixed sequence of events in order for a behaviour to make sense. Consider a get-acquainted-with behaviour of a dog-like creature. Such a behaviour could be triggered when an unknown object entered the dog's knowledge of the world, and a possible "script" of such a behaviour might be:

1. face the new object
2. slowly approach
3. circle the object at a safe distance
4. move close and sniff
5. engage in an appropriate interaction, for example eat, bite, lick, flee etc.

4.2 Agent perception and memory

Another main area of observations we have made relates to the issues of agent perception, the use of memory for storing perceptual information, and for reasoning about the agent's

relationship to the world. In this context there are at least two relevant types of perception and thus also memory:

- spatial perception and memory – this deals primarily with the 3D structure of the virtual world and the objects in it, and is associated with the normal senses such as vision, hearing (audio), touching (tactile) etc.
- episodic perception and memory – as opposed to the spatial perception the episodic perception has a much stronger relationship to a notion of time, the linear progression of events over time and their causal relationships, e.g., "one event leading to another"

While we shall not pretend to have much in-depth knowledge about the latter, there are a few comments to associate with the presented agent architectures. Each of the three architectures have the possibility to maintain some measure of episodic memory. For example it is easy to extend the Bouncy concept so that it will wonder off and start playing if the user has made it sad and happy three times in a row. I.e., each agent can keep track of its *own* decisions, and what characterized the situations leading to these decisions. Similarly, all three agent architectures operate with dynamic internal mood variables, which are influenced by the passing of time (hunger increases as a function of time and activity). Thus, there is some measure of explicit use of a temporal dimension. Nevertheless, to our knowledge there is still no autonomous agent research which deals with episodic perception and memory in a unified, structured manner. Especially not events or episodes relating to other agents and their decisions. The main problems here are how to represent "episodes", how to identify/recognize them, and how to reason about them.

Other than what is pointed out in this section we have not yet performed thorough investigations into the area of episodic and temporal issues. This is not to say that episodic memory is not important! Even a rudimentary interplay between spatial and episodic memory is absolutely essential to create the simplest believability enhancing effects. Consider for example the get-acquainted-with behaviour described in section 4.1. Imagine the following scenario:

- a dog agent sees a cat for the first time and decides to get acquainted with it
- after having gotten to the circling stage of the behaviour the dog agent notices a human off to one side
- the partial acquaintance with the cat, and the novelty of the human might trigger the dog to start a get-acquainted-with behaviour on the human
- the dog may be in a dilemma as to which object to further investigate first, and thus may go back and forth between the cat and the human
- the dog may finally decide that the cat poses no threat and the human offers some promise of entertainment if not food

In order to enable the above simple scenario the dog agent needs both a spatial memory and some measure of episodic memory. Spatial memory because it needs to remember the presence of the cat when it turns to the human and vice versa,– and episodic memory because it needs to remember how far into the get-acquainted-with ritual it has gotten with each object when switching back and forth in the dilemma situation. Having pointed this out we shall abandon the issue of episodic memory altogether, and from here on we shall deal exclusively with the spatial perception based on models of senses such as vision and audio, and the relationship of the sensory input to a spatial memory.

Concerning spatial perception none of the presented agent architectures make any bindings on how the agent perceives the virtual world. All three architectures assume that the agent has access to whatever knowledge of the virtual world is relevant for the agent and its behaviour repertoire. Thus this is an open issue, with no constraints imposed by the architectures, and this is one reason why we have studied this area in some detail, as will be described in later sections.

Concerning spatial memory only one of the described architectures explicitly models and employs such a functionality namely the JAM agent architecture. JAM agents have a world model which stores the beliefs of the agent. On the other hand the two other architectures base their behaviour triggering on sensory information, and for that purpose there need not be any difference between what is actual, current sensory information, and what is memories of objects perceived in the past.

In summary it appears that no agent architectures impose any constraints on how the perceptual system of the agent should work and how it should be backed up by a spatial memory.

4.3 The objective versus the subjective layer

In the previous two sections we have highlighted the areas of agent scripting and perception/memory, and pointed out our main observations regarding them. During working with developing autonomous agents we found ourselves spending a lot of time programming purpose-made sensing capabilities, the planning of agent movements etc. As a consequence actual agent behaviour design, or personality/role issues suffered. I.e., every time we wanted to investigate something, too much time was wasted on re-implementing something which was already made in an other context for another agent.

This lead us to adopt the working hypothesis that agents could be split into two separate layers: a subjective and an objective:

- subjective layer – special to a character, or an agent exhibiting a particular personality, playing a specific role
- objective layer – common to all characters, a platform providing functionalities which all agent require

Initially the purpose of designing and implementing the objective layer was to save time later when experimenting with various solutions for the subjective layer. Later the work on the objective layer took on a purpose of its own: to learn about which functionalities are the general ones, i.e., how high in the internal agent hierarchy can we go before it becomes personality or role dependent?

We have later learned, (see the proceeding paper by Andersen, Madsen and Granum, which pinpoints some issues concerning engineer vs. humanist points of view), that the borderline between the objective and the subjective layer is fluent/dynamic. We have therefore decided to use the terms High Level Agent layer (HLA) and Low Level Agent (LLA). I.e., the LLA is a common platform on which HLA's are built; all HLA's have an LLA beneath, but each of these LLA's are identical, the only difference being some parameters that are set concerning for example the 3D shape of the agent and how it moves.

The LLA provides the HLA with functionalities that deal with sensory input, continuous maintenance of a spatial memory, and spatial reasoning capabilities. Examples of such reasoning capabilities are path planning for moving to various places and planning of how to hide from other agents. The LLA accepts action commands from the HLA and executes them, for example "moveTo X", "follow X", "find X", "fleeFrom X", and "kill X". Essentially everything that involves physical movements in the virtual world, plus making utterances (on request from the HLA).

By nature the LLA is tightly connected to the virtual world. The actual relationship, and the architecture of the LLA will be expanded upon in section 5.2. First we describe three different approaches to sensory perception, where the latter is the one we are currently pursuing.

5 Maintaining a spatial memory within the agent

As should be evident from the discussion in the paper thus far it is vital that agents have (access to) functionalities providing them with sensory information. They need to be able to sense the state of the virtual world in order to relate to it and act on it. On our work we have investigated three different ways of providing agents with sensory information: 1) giving the agent direct access to the geometry database of the virtual world, 2) providing agents with iconic sensory data, e.g., digital images in the case of the vision sense, and 3) higher level partly interpreted sensory information, the socalled percepts.

We applied the first approach in the case of the Bouncy agent. Bouncy had direct access to the geometry database, plus direct access to the microphone input, and the input from the dataglove. In fact the programming code for Bouncy was mixed in with the geometry handling and visualization. While this is an easy and rapid approach for implementing one particular agent, it is not a viable approach for general work on researching autonomous agents.

In the subsequent section we describe some work regarding the two other approaches to generating sensory information for agents. Regardless of the approach it has been imperative for us, that the agent's senses emulate what is intuitively expected of sensory perception, e.g., the agent should have limited field-of-vision, and limited range of hearing. We consider this an important element in facilitating believability and make it intuitive for the observer (and other agents) to identify with an agent's situation (what it knows and how this might influence its decisions). For this we need a sensory system based on an understandable, realistic functionality: What You Have Sensed Is What You Know – WYHSIWYK.

5.1 Iconic representation of sensory information

A 2D digitized image, and array of pixels, is an iconic representation of visual information. Similarly audio samples is an iconic representation of sound. Especially in the case of visual information a substantial body of research has investigated the use of iconic representations for agent sensory input, (Blumberg 1997, Monsieurs, Coninx & Flerackers 1999). That is, presenting the agent with actual images of the virtual world, and let the agent itself recognize objects, judge distances to things, etc.

To investigate this issue, in the case of the vision sense, we developed a virtual reality computer game taking place in a maze. It was a multi-player game, but the world was also populated by autonomous agents, and there was no exterior way of determining whether a character was an avatar of a human player, or an agent. The objective of the game was survival in a world where there are resources (food and ammunition) and everyone can shoot at everyone.

The agents were based on a simple version of Blumberg's architecture with internal mood variables such as anger, health, and hunger. The agents had four basic behaviours: fleeFrom X, hunt X, findResource (food/ammunition), and goBeserk! The latter was a behaviour employed when the situation looked hopeless, and self-preservation seemed futile.

The system could perform visual rendering of the dynamic world from the point of view of each character, agents and avatars alike. Additionally the system could provide an overview of the entire world. Examples of such renderings are shown in figure 6.

The essence here is, that each agent had the 2D iconic images provided to it at a rate of approximately 5 images per second. From these images the agent was able to recognize objects such as doors, walls, characters and resource items, based on shape, size and color information, i.e., typical computer vision techniques. The agent also had simulated stereo vision in order to judge distance in the virtual world. The stereo vision was emulated by utilizing the z-buffering of the graphics engine, similarly to what Monsieurs et al. (1999) proposed.

From the visual perception the agent maintains a spatial database at two hierarchical levels:

Figure 6: The large window shows a scenario overview, and one of the rooms contains an avatar, an agent, and a resource (ammunition) item. The top-most of the smaller windows show what the avatar sees: part of the room and a character (in this case an agent). The other small window shows what one agent sees: part of the room, a character (the avatar) and a resource item.

- inside-room level – while moving about in a room the agent builds a detailed map of the room
- at room-to-room level – every room that the agent discovers and maps while roaming about gets inserted in a topological map of the world

The room-to-room level is used for planning at world level, for example going back to the place where the nearest food has been seen. The inside-room level is used for detailed interaction with other characters, for example fleeing from a character, see section 6.3.

From a philosophical point of view the use of iconic representations of sensory data is highly attractive. For example it facilitates a natural element of sensory imperfection, such as not being able to recognize an object if it is partially hidden by some other object. In fact this approach emulates real-world sensing as closely as possible: a 3D world is projected to 2D images (e.g. on a human retina), and the 2D images are then analyzed to create a *model* of the 3D world.

Nevertheless technical problems exist. Primarily there is a substantial computational overhead associated with visual rendering of the virtual world from the point of view of all agents. Secondarily it is difficult to achieve suitable performances of the computer vision techniques for analyzing the images and providing the agent with a perceived model of the environment. Especially if the vision sense is to work properly in all types of environments with any kind of objects and lighting. Because of these technical problems with this iconic approach we have abandoned it in favour of using a more high level abstract representation for sensory data.

5.2 Higher level sensory information – percepts

In the above described work it was the task of a Virtual Environment Server (VE server) to provide each agent (and human user) with images of what they could see at any point in time. Each agent would log onto this VE server in order to subscribe to this functionality, and in turn the VE server would accept and carry out requests for movements from each agent. As such the VE server plays a very important role, in that it simulates an entire virtual world. In fact VE servers are an entire research area on its own, and cannot completely be separated from the autonomous agents, since they communicate at such an extensive level.

This paper will not address VE server issues per se, but it is impossible to describe agent perception and reasoning/action without giving an overview of the relationship between the VE server and the agent. Having abandoned the concept of iconic representations of sensory data, our currently ongoing work focuses on the use of socalled percepts. According to Webster's Dictionary a percept is "an impression of an object obtained by use of the senses". The word is related to the word sense-datum, which means "an immediate unanalyzable private object of sensation". We have designed and built a VE server which Low Level

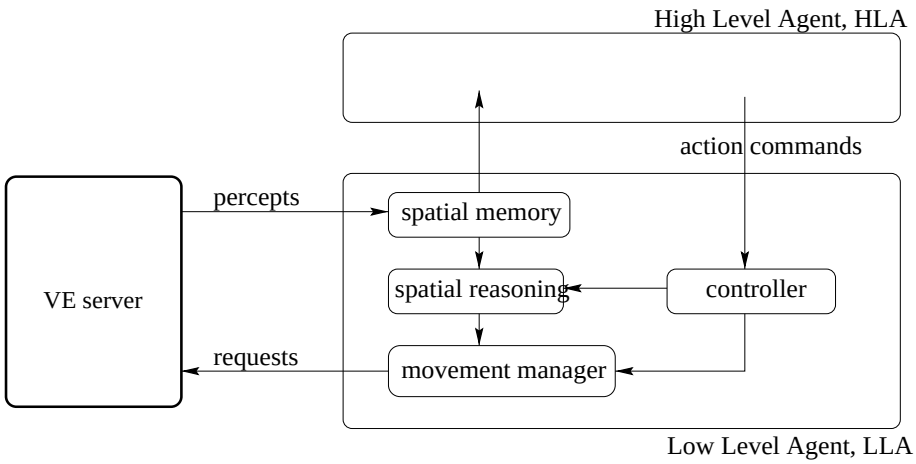

Figure 7: Overview of relationship between virtual world, Low Level Agent, and High Level Agent

Agents (LLA's) can log onto, and from which LLA's can get sensory data in the form of percepts. The VE server thus simulates for example the vision and the auditory senses of each agent, and gives the agent information (percepts) concerning the objects that have been sensed.

Before going into more detail with the content of percepts and the generation thereof, the relationship between the VE server, the LLA and the HLA is depicted schematically in figure 7. On an overall scale the LLA is connected to the VE server, and the HLA is connected to the LLA. The LLA receives percepts from the VE server, and in turn sends movement and sound requests to the VE server. That is, if the agent wants to move, or make a sound, it must request the VE server to carry out these actions. Inside the LLA percepts are used to keep the contents of a spatial memory up to date, and this spatial memory is used for planning the movements of the agent, for example planning a path from the current position to some desired position. I.e., the HLA can send an action command to the LLA: "moveTo well". The LLA will then plan a path to the well, and follow the path until it arrives at the well. Section 6 provides more detail on the spatial reasoning.

As seen from figure 7 the spatial memory of the LLA is a service provided to the HLA. The LLA uses the memory for reasoning about the geometry of the world, e.g., to avoid running into things. But the HLA uses the same memory for "affective reasoning", i.e., figuring out what a particular object in the virtual world "means" to the particular agent. Popularly speaking a duck in the virtual world merely represents an obstacle or a special location to the LLA. But to the HLA the same duck may represent potential lunch (for a fox agent) or a playmate (for another duck agent). The LLA is the objective layer, whereas the HLA is the subjective, character dependent layer. The HLA utilizes the LLA to carry out action commands, for example "hunt duck" or "playWith duck".

Returning to the issue of percepts. The VE server generates percepts for all agents, i.e., the front-end sensory apparatus of an agent resides in the VE server. The designed platform supports arbitrary senses, but currently 4 senses are implemented: 1) vision, 2) audio, 3) tactile, and 4) sixth sense, see figure 8. The former three are emulations of the natural senses. The latter, the sixth sense, reflects a need for an agent to "feel" the presence of nearby objects, even if they are outside the agent's field of view, and even if they are not making noises. Without this sixth sense an agent would have to turn constantly to monitor the position of objects inside the agent's "personal space".

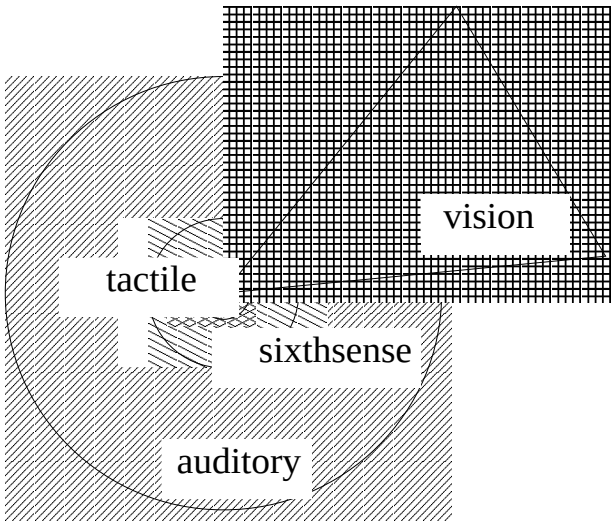

Figure 8: Four senses are currently implemented. The spatial properties, e.g., range and field-of-view, of each sense can be modified by the agent itself.

Figure 9 shows a screen shot from a farm scenario loaded into the VE server, and with a number of LLA's logged on (cows, some sheep in the background, a dog, a farmer, and a sheep in the foreground).

Having thus described the overall mechanism for generating and using percepts in our current system, we can be more specific on the contents of a percept. It is though important to stress that percept content can be anything. It is a completely flexible concept (also in our implementation), and the content listed below merely represents what we sofar have identified as useful information for the agents *we* are working with.

Percept content:

- primarily for objective, spatial reasoning
  - ID – the name of the sensed object, e.g., "Duck01" or "Donald"
  - bounding box – the coordinates of the object's bounding box

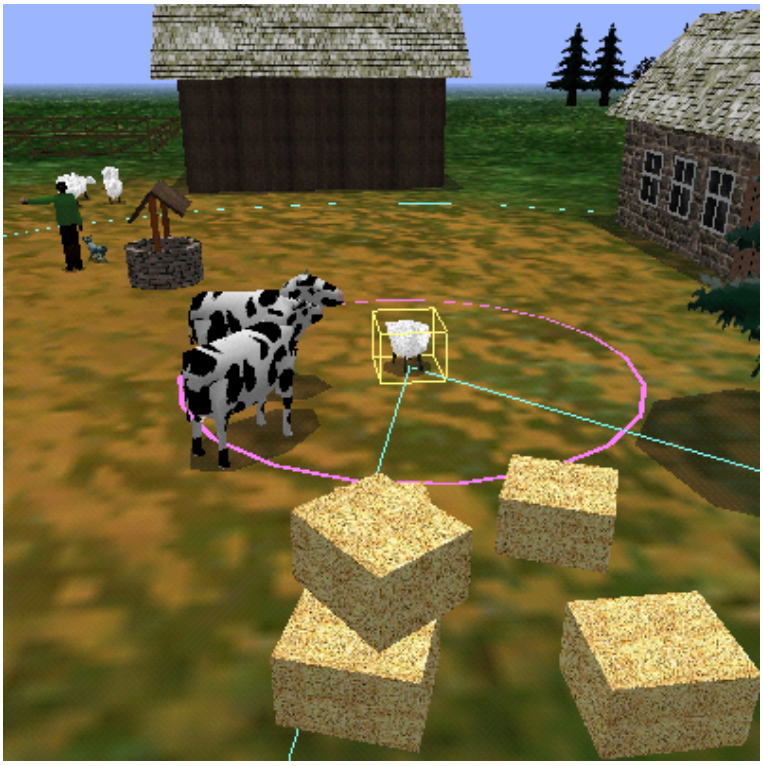

Figure 9: Visualization of the range of the agent's senses. The field of view of the vision sense is illustrated as the two lines emanating from the agent, so in the situation shown, the sheep agent is "seeing" the hay bales. Similarly, it is "aware" of the presence of the two cows, since the cows are inside the range of the sixth sense. Furthermore, the farmer and the dog in the back of the scene are within "hearing" range. The tactile sense is triggered if an object comes inside the bounding box around the agent.

  - position – the location of the object in the virtual world
  - orientation – how the object is oriented in the virtual world

- primarily for subjective, affective reasoning
  - sensory modality – indicates what sense caused the generation of the percept, e.g., vision
  - types – each object can belong to a number of object categories, for example "animal", "duck", and "herbivore"
  - adjectives – each object can be associated with several qualities, for example "angry", "hungry", etc.
  - current activity – the current activity of the sensed object, for example "waving", "eating" or "greeting"

– speech act – indicates the "meaning" of the sound if the object was sensed with the auditory sensing modality, e.g., the sound could mean "refusal", "acceptance", "threat", etc.

The percept fields of types, adjectives, current activity, and speech act can be dynamically modified by an agent. For example a fox agent can tell the VE server, that he is currently "eating". This information will then be passed via percepts to all other agents that sense (see, hear, etc.) the fox. In this context it is up to the percept receiving agent to only use as much information from the percept as is relevant given the sense modality that generated the percept. For example an agent will be notified of the position of an object when hearing it, but it should only use the knowledge about the direction to the object that produced the sound.

In our implemented system percepts are generated at a rate of 10 times per second. This high percept rate is required for the LLA to be able to maneuver consistently in the virtual world. For example to be able to follow the farmer, the agent will get an update of the position of the farmer ten times a second, so that it can keep moving in the right direction in a smooth manner. This naturally presumes that the agent can "see" the farmer, i.e., that the farmer is inside the agent's field-of-view, or is close enough to be picked up by the sixth sense.

5.3 Spatial memory as a database with entries and queries

As seen from figure 7 the Low Level Agent maintains a spatial memory; in effect by integrating all the information coming in as percepts. The very first time an object is sensed, i.e., the first time a percept is received concerning some object, the object is created as an instance in a database – the spatial memory. All the available information about the object is stored in the database.

Any subsequent percepts concerning the same object are used to update the information concerning the object in question. For example update the knowledge of the object's position, or its activities. Thus, the LLA is "born" with an empty spatial memory, which gradually gets to contain more and more entries as time passes and the agent has sensed more and more objects. There is no limit to how many objects the LLA can have in the spatial memory, and currently we have not imposed any mechanism for forgetting objects.

The spatial memory supports various queries for information. For example the spatial memory can return the identity, and any other information, concerning an object of a certain type. This can be used by the High Level Agent to ask the spatial memory questions such as "what is the nearest object of type **hay**?"

The spatial memory also supports various events to be triggered. For example the HLA can ask the spatial memory to generate an event, if an object of a certain identity comes within a certain range.

As explained the contents of percepts, and thus the exact information concerning each

object in the spatial memory, can be anything. It is up to the designer to decide what is useful information for various types of agent reasoning. As mentioned previously in the paper, we are currently focusing on the Low Level Agent and its perceptual capabilities, and thus the following section describes some concrete issues relating to spatial reasoning.

6 Use of spatial memory for spatial reasoning

We shall discuss three issues in this context: motion/path planning, avoiding hallucinations, and empathetic sensing. The path planning issue is a concrete task that an LLA needs to be able to perform. The issue of avoiding hallucination deals with making the LLA able to notice, when the state of the virtual world does not correspond to what is recorded in the spatial memory. And empathetic sensing is a term we use to describe the fact, that an agent can actually "imagine" what some other agent should be sensing, i.e., "picture the world from another character's point of view".

6.1 Path planning

In order to be able to figure out how to move around in the virtual world an agent needs path planning capabilities. In our system design this functionality resides in the LLA, more specifically in the spatial reasoning module of the LLA, see figure 7. In essence path planning involves computing an obstacle free path/trajectory from the current position to some desired goal point. A goal point can be the location of some particular object, for example the stable.

Decades of path planning research in the area of mobile robotics have proven that the most flexible and efficient way to perform path planning is to use numerical potential fields composed from considering particular object in the world to exhibit attractive or repulsive forces, (Latombe 1991).

In the previous paper by Andersen and Callesen the concept of COMPULSION schemas was described, which exactly is the concept of numerical potential fields. Once a potential field has been computed, the agent can search for a path, guided by the changing values of the potential field. In the simplest case there is only one attractive force (the goal point), and no repulsive forces. The value of the potential field at any location in the virtual world is then proportional to the distance from the location to the goal point. But note: it is not the line-of-sight distance, it is a distance that takes into account the obstacles in the world. In this case a simple gradient descent search strategy can be used to find a path to some goal point.

Figure 10 shows two scenarios where a pig agent has been asked to move to one of the two cows behind the fenced pen. In the first case the shortest path is left of the pen, and in the second case the shortest path is right of the pen, by passing between the farm house and the tree in the middle of the image. Figure 11 shows a longer, more complicated

path generated by the pig agent. Every time a path is generated, the entire content of the agent's spatial memory is used to construct a map of the virtual world (as known to the agent at that time), and then the distance-based potential field is computed. Finally a search algorithm is used for finding the path to the desired goal, and the agent then follows the path. If the path is obstructed (e.g., because an object moved in the way), a new path is generated.

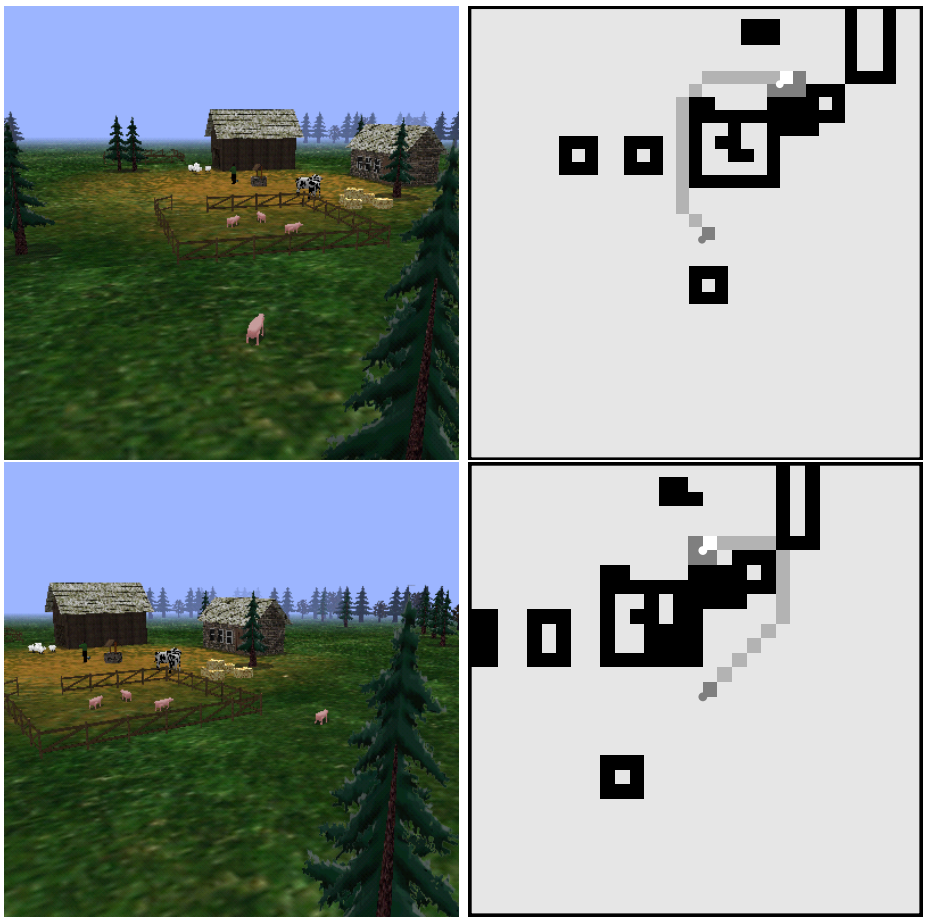

Figure 10: Three world snapshots and paths generated by the LLA, in this case a pig. The maps shown on the left are generated by the agent to find the path. In each map light grey represents free space, whereas black represents obstacles. The path is shown in dark grey, and the goal point is indicated by a small white square. In all paths the initial position of the agent is in the center of the map.

Regardless of the fact that potential fields are generally considered the most efficient tool for path planning, some researchers in the autonomous agent community employ other techniques, e.g., Monsieurs et al. (1999). We still claim that such techniques cannot solve path planning problems in the case where there are multiple constraints on the path, i.e.,

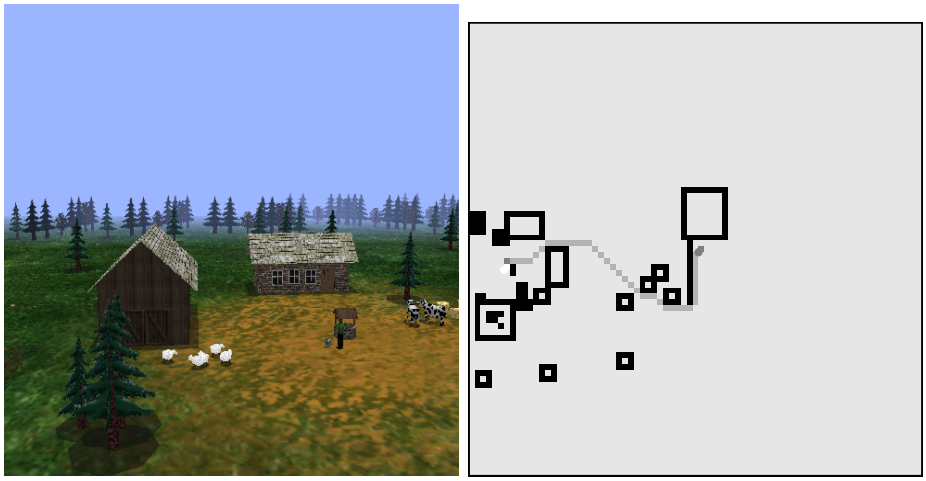

Figure 11: In this case the pig agent is hiding in the forest in the rear of the scene, and has planned a path to the dog near the well in the farmyard.

that it should not only be the shortest. This case is also discussed in the previous paper by Andersen and Callesen. For example, if a duck wants to go to the duck pond, while simultaneously having to make sure it does not get too close to the dog. In this case the duck pond will act as an attractor, and the dog as a repulsive force.

If multiple attractive and repulsive forces exist, the potential field can have local minima, and then a gradient descent search strategy is not sufficient. In this case a more complicated search strategy is required, and the classical A* search algorithm is a suitable choice. Descriptions of the A* algorithm may be found in Andersen, Madsen, Sørensen, Kirkeby, Jones & Christensen (1992), Latombe (1991), and Rich (1983).

We have not yet implemented the facility for combining multiple constraints when computing a path, so we cannot demonstrate this with a concrete example. But figure 12 shows a case, where instead of planning a path *to* some object, the agent has been asked to plan a path *away from* some object. This path is the one which most quickly maximizes the distance to the object, and as such it can be used for planning. e.g., how to flee from another agent in the virtual world.

6.2 Avoiding hallucinations

The spatial memory of the LLA was described in some detail in sections 5.2 and 5.3. But if one is not careful, this kind of spatial memory can lead to "hallucinations". Consider the following example scenario, (figure 9 can aid in visualizing this): a sheep agent sees a dog near the well in the farmyard, and the dog is entered into the sheep's spatial memory. Then the sheep goes behind the farm house, and thus can no longer see the dog. Then the dog moves away from the well, and goes somewhere else, but since the sheep cannot see

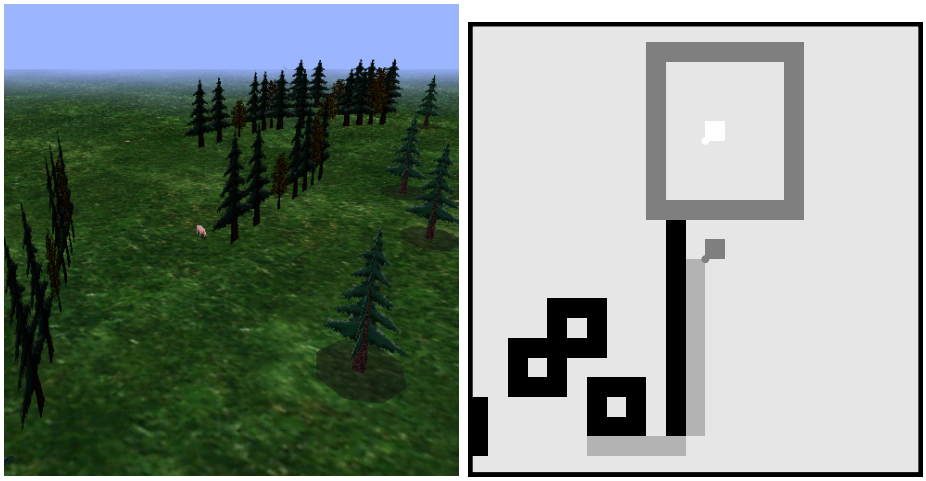

Figure 12: In the situation shown the pig has been ordered figure out a path that will take it *away* from the group of trees on the left, i.e., "fleeFrom forest". The large grey square in the map is the space taken up by the group of trees the agent is trying to escape from. This is because of the bounding box of the tree group being so large, which is due to a minor technical imperfection in our current VE server.

the dog, the sheep does not know this. But this is as it should be, because that is the way sensing works in real life. The problem arises when the sheep returns to the well to play with the dog. According to the sheep's spatial memory the dog is at the well. So when the sheep comes back to the well it should have some mechanism for detecting that the dog has vanished. Otherwise it will be hallucinating the dog.

To solve this problem the LLA is capable of determining what it would *expect* to sense at any point in time. That is: according to the contents of the spatial memory, what would I expect to sense right now? This corresponds to people closing there eyes before entering into a room, saying: "I expect to find my computer in this room". If the computer turns out not to be there, this is noticed.

In our LLA's there is a visual imagination (an inner eye) which works on the spatial memory. Continuously this inner eye determines which percepts are expected right now. For example when the sheep agent returns to the well it *expects* to see the dog. If the expected percepts concerning the dog do not arrive from the VE server, the object is marked as 'vanished'. It is not removed from the spatial memory, but only marked as having unknown location and other properties. If later a percept concerning the object is received, the 'vanished' indicator is removed.

6.3 Empathetic sensing

The visual imagination, the inner eye, as described in the previous section can be used for different purposes. It can be used to estimate what *other* agents can sense. Figure 13 shows an example, where an agent is using its own spatial memory to predict which areas are invisible to another agent.

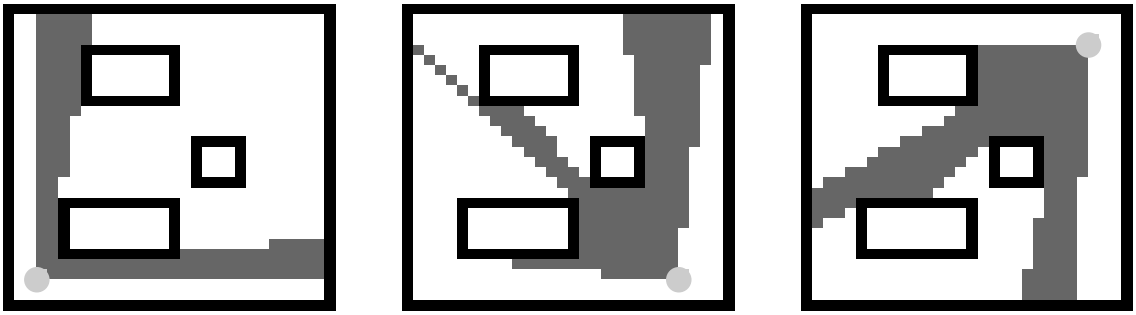

Figure 13: Using the internally simulated vision sense, the agent can figure out what areas of a scene are hidden from the field of view of another agent, and thus can be used for hiding. In the three examples the white area is what the other agent cannot see, and the grey area is what the other agent *can* see.

Of course an agent cannot know exactly what another agent knows, but it is possible to make plans from the concept "according to my knowledge of the world, the other guy can or cannot sense this or that". This is a fundamental prerequisite in supporting plot structures or narratives based on elements of scheming and conniving. I.e., an agent can essentially reason about "I want to make sure he sees this", or "I want to make sure he does not see this". This is a concrete step towards using empathy, or putting one self in the position of others, which is critical to creating suspense and interesting plots as pointed out in the previous paper by Andersen and Callesen.

7 Discussion

In this paper we have described how our desire to investigate architectures for autonomous agents in interactive virtual environments has led us to divide autonomous agents into to two levels: the Low Level Agent and the High Level Agent. The paper has described some important characteristics of the LLA, i.e., its spatial memory, how it is maintained, and how it is exploited for spatial reasoning. This entire development was spurred by an interest in having a platform for experimenting with the higher character level of autonomous agents – agents as actors, as it were. The purpose of the LLA is to serve as an objective layer of services, which the HLA can build upon. In the following sections we shall briefly touch

upon topics concerning the use of the LLA, and finally we point out a few current activities building upon what has been described in this paper.

7.1 Aspects of the interface between High and Low Level Agent

The Low Level Agent allows the High Level Agent counterpart access to the spatial memory, and allows the HLA to query information from it. As described the spatial memory also contains information about objects, which is solely intended for affective/subjective reasoning, and as such is only relevant to the HLA. For example the type of an object is not relevant to the LLA, but highly so to the HLA.

Conversely, the HLA may issue action commands to the LLA. These action commands are of the type "face object" (turn head in direction of object), "moveTo object", "hideFrom object", "hunt object", "playSound", etc.

Within the LLA these action commands are executed according to simple, built in scripts, which are in fact simple Finite State Automata, FSA's. These scripts form a hierarchy, which can be combined in arbitrary ways. In fact all action commands are executed using only three basic actions: moveForwards, turn, and playSound. In the near future the LLA will also be able to, upon request, play various animations, i.e., make somersaults, look threatening, look sad, wave, etc. The hierarchy allows the LLA to offer a specific way of carrying out an action. For example the "hunt object" action can be scripted as a sequence of sub-actions: "hideFrom object", "face object", "moveTo object" (when "object" is facing the other way). Any such sub-actions utilize the moveForwards, turn, playSound, and playAnimation action primitives. In this manner the LLA offers an objective way of carrying out higher level actions, by making combinations of more primitive ones. Still the HLA has access to all levels in this action command hierarchy, if for example it is an illustrative feature of a character's personality to carry out an action in a certain manner. So an HLA can carry out a "hunt object" in another manner than the one pre-scripted into the LLA, should it so desire. This is a clear example of the fact, that the borderline between the objective and subjective agent level has to be fluent/dynamic. Sometimes the way something is done is at least as important as the fact that it is done (see the next paper for more detail on this).

7.2 What does it mean to be an avatar?

Normally the avatar concept relates to the representation of a user in a virtual world. As such an avatar does not have any capabilities in terms of sensing and acting on its own. All avatar actions are completely, and directly controlled by the user via some interface, for example the mouse.

In our work an avatar is an agent where some functionality has been taken over by a human. One way of using an agent as an avatar is for the user to control the *movements* of the avatar, again using some interface. Another possibility is to control the agent's

movements using the same level of abstraction as the HLA communicates to the LLA, i.e., using action commands such as "moveTo object". These two possibilities are currently supported in our implementation, but we are actively investigating alternative methods for avatar control.

Regardless of the abstraction level for avatar control, the agent is still "alive" when it is being used as an avatar. That is, it still receives percepts as it would if it were moving around on its own accord. Consequently it is continuously maintaining its spatial memory. Thus, at any point in time the user can release the agent from the avatar function, and the agent can continue "as if nothing happened". When the agent is released from the avatar mode it will have a spatial memory, which is consistent with what the agent sensed during its stint as avatar. This is a feature we are using in ongoing research projects, where the user is allowed to choose any character in the virtual world and use as avatar, at any point in time. When an agent is chosen as avatar, the "virtual camera" being used to visualize the virtual world to the user is attached to the head position of the chosen agent, such that the user sees the world from the point of view of the avatar.

7.3 Ongoing and future work

We are continuously building upon the work described in this paper. Two of the more important active directions of work are investigating architectures for High Level Agents, and expanding the interactive capabilities of agents to include a linguistic modality.

We are currently designing a High Level Agent based on many of the ideas behind the agent architecture proposed by Blumberg (1997), (see also section 3.1). Drawing upon our experience as presented in the present paper, we are specifically focusing on designing an agent architecture, which actively uses the spatial memory and associated functionalities. This is an aspect we have found to be missing in the original work by Blumberg, and others for that matter.

While working on High Level Agent architectures we are also making provisions for incorporating spoken language understanding into agents. Together with a research group at the Center for Language Technology in Copenhagen we have designed a way for a number of autonomous agents to share a module providing a speech recognizer, a parser and a dialogue manager. Ideally each agent would have a separate module such as that, but this is not realistic from a computational point of view. So we have designed a system such that each agent uses the same speech module as a resource which translates whatever is said by the user into something meaningful for the agents. This translation is agent specific, and only the agents within hearing range of the user's avatar will hear what is said. Conversely each agent will be able "speak" by sending text to a text-to-speech generator.

In conclusion we are working towards designing autonomous agents using a bottom-up approach, where we maintain a strong focus on ensuring, that the agents do not cheat – What You Have Sensed Is What You Know. There is no such thing as a user; there may be one of the other agents around who is in fact controlled by a human, but this is just a

coincidence.

8 Acknowledgements

The work described in this paper is all done in fruitful collaboration with numerous people. To mention but a few of the more important, the authors wish to thank Bo Cordes Petersen, Rasmus Agerholm, and Paolo Pirjanian for their invaluable contributions.

Finally the support of the European research project, PUPPET, ESPRIT Long Term Research EP 29335 under the i3 Early School Environments programme, and the Danish Research Council research project STAGING is gratefully acknowledged.

References

Andersen, C. S., Madsen, C. B., Sørensen, J. J., Kirkeby, N. O. S., Jones, J. P. & Christensen, H. I. (1992), 'Navigation using range images on a mobile robot', *Robotics and Autonomous Systems* (10), 147 – 160.

Blumberg, B. (1997), Old Trickes, New Dogs: Ethology and Interactive Creatures, PhD thesis, Massachusetts Institute of Technology, Program in Media Arts and Sciences.

Brøndsted, T., Nielsen, T. D. & Ortega, S. (1999), Affective multi-modal interaction with a 3d agent, *in* 'Proceedings: Eighth International Workshop on the Cognitive Science of Natural Language Processing, Galway, Scotland', pp. 102 – 109.

Brooks, R. (1986), 'A robust layered control system for a mobile robot', *IEEE Journal of Robotics and Autonomation* **2**(1), 14 – 23.

Brooks, R. (1991), Intelligence without reason, computers and thought lecture, *in* 'Proceedings: International Joint Conference on Artificial Intelligence, Sidney, Australia'.

Franklin, S. & Graesser, A. (1996), Is it an agent, or just a program?: A taxonomy for autonomous agents, *in* 'Proceedings: Third International Workshop on Agent Theories, Architectures, and Languages', Springer-Verlag.

Huber, M. J. (1999), Jam: A bdi-theoretic mobile agent architecture, *in* 'Proceedings: Third International Conference on Autonomous Agents, Seattle, Washinton', pp. 236 – 243.

Latombe, J.-C. (1991), *Robot Motion Planning*, Kluwer, Boston.

Madsen, C. B., Pirjanian, P. & Granum, E. (1999), Can finite state automata, numeric mood parameters and reactive behaviours become alive?, *in* 'Proceedings: Workshop

on Behavior Planning for Life-Like Characters and Avatars, held in conjunction with the I3 Spring Days, Sitges, Spain', p. (4).

McCrae, R. R. & Costa, P. T. (1989), 'The structure of the interpersonal traits: Wiggin's circumplex and the five-factor model', *Journal of Personality and Social Phsychology* **56**(4), 586 – 595.

Monsieurs, P., Coninx, K. & Flerackers, E. (1999), Collision avoidance and map construction using synthetic vision, *in* 'Proceedings: Workshop on Intelligent Virtual Agents, Salford, UK'.

Pirjanian, P. (1998), Multiple Objective Action Selection and Behaviour Fusion Using Voting, PhD thesis, Department of Medical Informatics and Image Analysis, Aalborg University, Denmark.

Pirjanian, P., Madsen, C. B. & Granum, E. (1998), Behaviour-based control of an interactive life-like pet, Technical report, Laboratory of Image Analysis. Not published.

Rich, E. (1983), *Artificial Intelligence*, McGraw-Hill International Editions.

Silva, D., Siebra, C., Valadares, J., Almeida, A., Frery, A. & Ramalho, G. (1999), Personality-centered agents for virtual computer games, *in* 'Proceedings: Workshop on Intelligent Virtual Agents, Salford, UK'.