Designing 3D Virtual Worlds as a Society of Agents

MAHER Mary Lou, SMITH Greg and GERO John S.
Key Centre of Design Computing and Cognition, University of Sydney

Keywords: Agents, 3D virtual world, agent communication, virtual architecture

Abstract: We consider virtual architecture as 3D virtual worlds able to support human activities and collaboration needs in digital virtual environments. 3D virtual worlds can go beyond the simulation of physical worlds to become dynamic, adaptable worlds by incorporating agents in the representation of the world. Agents are software systems that are capable of acting autonomously according to their own goals and beliefs. A society of agents accommodates agent communication and collaboration as part of the agent reasoning. In this paper we present a framework in which agents become the basis for the elements of a 3D virtual world. This framework is presented as having a model for an agent that can interact and reason about the 3D world, and as a model for agent communication. The model is illustrated by the design of a virtual conference room.

1 INTRODUCTION

3D virtual worlds are place-like environments that can support communication and collaboration. These 3D environments offer the professional community an alternative communication channel and can extend available office space for meetings. The design of these worlds is similar in many ways to the design of the physical world by developing a 3D model of the place and then constructing the design in a virtual world platform. The development platforms for 3D virtual worlds have mostly focused on efficient visualization and real time rendering to support activities such as walking, chatting, and interacting with other people in the world. Recent developments in virtual worlds allow the 3D environment to support more complex activities by associating programmed behaviors with elements of the world.

The design and implementation of computer games, online social worlds, and educational virtual worlds requires a significant effort in developing the infrastructure. This infrastructure then becomes old as technology changes because the design elements for virtual worlds are not well established and generally rely on custom programming. Most worlds are largely static, with any dynamic behaviour of objects restricted to triggering pre-programmed behaviours.

We propose to extend the idea of object-oriented virtual worlds to agent-oriented virtual worlds to overcome the establishment of static places that do not respond to their use. In this paper we present a flexible, object-oriented framework that we are developing in which agents become the basis for all elements of a world. By using agent models we can design intelligent, interactive virtual worlds that exhibit

various kinds of behaviours. We describe an ongoing implementation of the framework using a Java Native Interface to the Active Worlds[1] (AW) software development kit. We illustrate the framework with a virtual conference room society of agents. The 3D model of the conference room is illustrated in Figure 1.

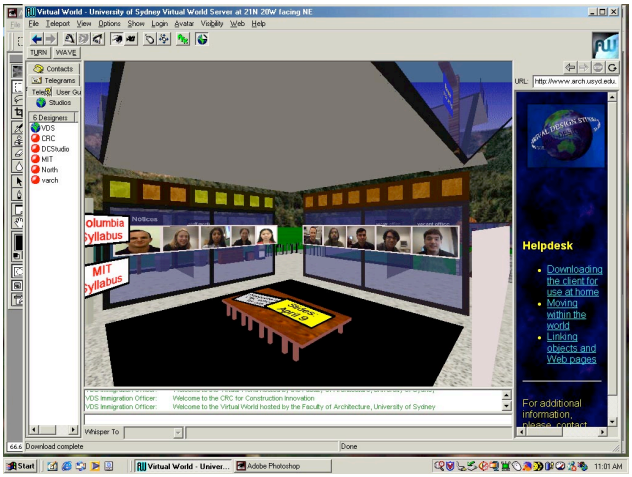

Figure 1 A 3D model of the virtual conference room

2 A FRAMEWORK FOR AN AGENT-ORIENTED 3D VIRTUAL WORLD

3D virtual worlds are networked, multi-user environments that support communication and collaboration in a place-like context. The virtual worlds that we are considering as the basis for our agent-based virtual world are object-oriented systems that associate a 3D model and a behaviour with each element of the world. Examples of such worlds include: Active Worlds and VirTools[2]. The design of these worlds has attracted the attention of the architectural community because of the similarity between the design of physical buildings and virtual worlds to support various human activities. Various design principles and examples have been explored for professional and educational activities. We would like to go beyond the design of the virtual world as a static 3D world to be an adaptable agent-based world. In order to develop an agent-based world we present a framework in which each object in the world is potentially an agent.

An agent is a computer system that is situated in some environment, and is capable of autonomous action in that environment in order to meet its design objectives (Wooldridge 1999). Agents receive input (sense-data) from their environments and

[1] http://www.activeworlds.com

[2] http://www.virtools.com

act to alter that environment using effectors (and effect-data). Autonomy is taken here to mean that each agent has it's own thread of control and that it decides by itself what actions to take (Jennings 2000).

Recent agent research has focussed on multi-agent or distributed agent systems and the interactions of agents (Ferber 1999; Wooldridge 2002). Distributed and multi-agent systems consider issues related to agent communication and collaboration. In design there has also been interest in distributed design agents. Combining agents and situatedness has lead to the development of situated design agents (Gero and Fujii 2000). A situated design agent is able to respond to its environment by constructing it own representation of the situation and reasoning about the current situation. The agent model presented in (Maher and Gero 2002) and shown in Figure 2 provides the underlying model for a single agent in a virtual world.

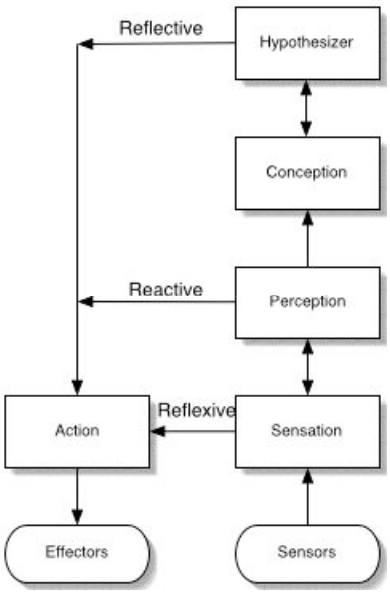

Figure 2 Model for a single agent in a 3D virtual world (from Maher and Gero 2002)

The agent is able to sense and have an effect on the virtual world through its *Sensor*s and *Effectors*. *Sensor*s receive uninterrupted messages from the virtual world and other agents and produce sense-data through the *Sensation* process. *Perception* is the process that interprets sense-data and recognise patterns as percepts. Percepts are interpreted by the *Conception* process which assigns specific meanings to the patterns for further reasoning. The *Hypothesizer* monitors the percepts and concepts for situations that relate to the agent's goals and identifies relevant goals to be achieved. The *Action Activator* observes the sense data for reflexive action, the percepts for reactive action and the goals identified by the Hypothesizer for

reflective action. The Action Activator is responsible for triggering the effectors to make changes to the virtual world environment. Further details of the agent model underlying this can be found in (Gero and Fujii 2000; Smith and Gero 2002; Maher and Gero 2002).

Our framework for designing a 3D virtual world as illustrated in Figure 3 is based around a set of abstract classes that form the generic architecture for constructing a society of agents. Central to the framework are the *Society*[1] and the *Agent*. A *Society* is an aggregation of *Agent*s that share a common connection with a virtual world. Normally these share some ontological connection, such as a *Room* agent plus a set of *Wall* agents that collectively comprise a virtual conference room. The *Society* manages computational resources, such as the connection to the virtual world, on behalf of the *Agent*s. *Agent*s within one *Society* communicate by sending messages from the *Effector* of one *Agent* to the *Sensor* of another without such messages necessarily going through the virtual world server. This allows such agents within a society to self-organize without flooding the world with indecipherable chat. An *Agent* communicates with an *Agent* from another *Society* or with citizens of the world by sending messages through the world server.

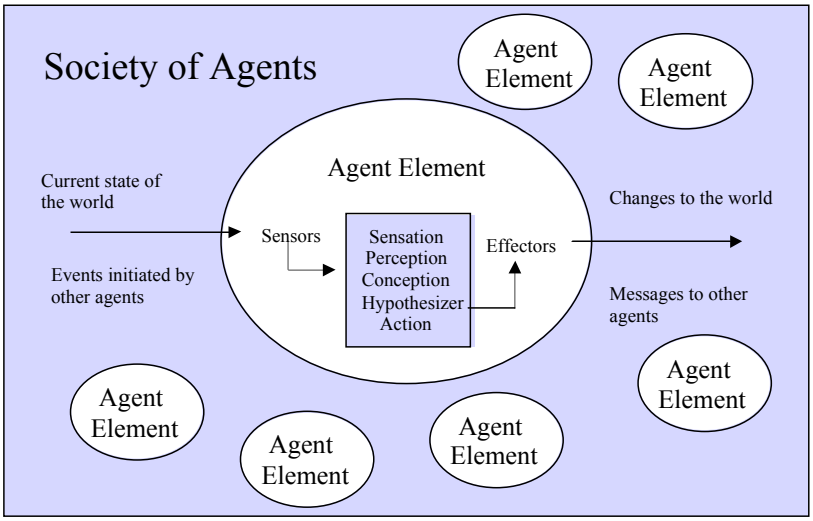

Figure 3. A society of agents for a 3D virtual world (Maher and Gero 2002)

Our framework assumes that an agent server and a 3D world server act as distinct components of an agent-based virtual world. The philosophy is the same as that underlying distributed operating systems: each server does one thing and does it

[1] In this paper, *Society* in an italic font refers to a computational entity such as a java class. Society in a normal font refers to a concept.

well, could be replaced by another with minimal impact on the rest of the system, isolates the impact of faults and service interruptions, and allows servers to reside on one or many processors (or even relocate or be usurped upon fault detection) without affecting users.

Each entity capable of reflexive, reactive or reflective behaviours is modelled as an agent (Maher and Gero 2002). The *Agent* has behaviours and an optional 3D representation, and can both dynamically change the 3D representation plus produce non-visual behaviours. The wall agents in the virtual conference room provide a visual and spatial boundary for the room agent, therefore collectively serving the function of providing a room-like space for activities that require privacy and proximity of the people in the world. The wall agents also have behaviors related to their function as elements of the world, for example a wall can function as a slide presentation screen, as an interface to information retrieval, or as a bulletin board An example of a behaviour that requires communication in the virtual conference room society is a “resize” behavior. The room agent is able to sense the number of citizen avatars within it and has as its goal to make the room just big enough to comfortably contain the avatars. In order to achieve this goal, the room agent will need to communicate with the wall agents when the size of the room is too small or too big.

3 INTER-AGENT COMMUNICATION

In developing a model for inter-agent communication, we need to address questions related to the content and intention of the communication. What should be communicated? Should it be a message that the receiving agent needs to interpret, or should it be more direct? Then we need to address the representation of the agent communication. One difference between an agent and an object is that an object encapsulates state and behaviour realization but not behaviour activation or action choice (Jennings 2000). Objects are “totally obedient to one another and do not have autonomy over their choice of action”. If a room agent dictated to a wall agent what it should do we would have a room agent and an aggregation of wall objects, not a room agent interacting with a wall agent. The room needs to be able to tell the wall what it wants without specifying how to do it. Further, agents shouldn't require knowledge of what the *Sensor*s of other agents look like. We require a simple and uniform communication mechanism that at the same time is expressive enough and does not unnecessarily constrain future agents.

For two agents to communicate requires a common language, ontology and protocol. The language should be powerful and expressive, but at the same time sensible by all *Agent*s. The ontology describes the concepts that agents have in common. The protocol we have employed in virtual conference room agent society is the Contract Net (Huhns and Stephens 1999) negotiation protocol. This well known protocol is based on contracting procedures from business, and it is well suited to cooperative agents. Figure 4 shows an example interaction for the conference room society. The agent with a problem to solve broadcasts a call for proposals, and agents that believe they can satisfy the call answer with proposals. One agent is then awarded a contract

to initiate their proposal.

The conceptual basis for agent communication languages (ACL) is the theory of speech acts (Austin 1962; Searle 1969; Cohen and Perrault 1979). Computationally, the two best known are KQML and ACL from the Foundation for Intelligent Physical Agents (FIPA) (Huhns and Stephens 1999). These are similar; the principal advantage of the FIPA ACL is that it has a well defined semantics (Wooldridge 2002). FIPA (2001b) describes 22 performatives that constitute the communicative acts defined in the FIPA ACL. Example performatives are Call-for-Proposal, Inform, Propose, and Request. FIPA (2001a) defines an abstract message structure, and content language specifications are provided for a number of languages. Each message contains a performative, message content, sender, receiver, and ontology. Each message may optionally also include language, reply-to, encoding, protocol, conversation-id, reply-with, in-reply-to, and reply-by elements.

Each communicative act should be implemented in accordance with a semantic definition of that act. FIPA (2001b) defines preconditions on communicative acts in terms of belief and intentions; that is, it says when it is acceptable to perform a particular communicative act. Our agent believes that *p* is true when it has a concept predicate p in working memory. In this sense our *Conception* process implements a belief revision function (Wooldridge1994). Intentions are partial plans. The hypothesizer uses goals and beliefs to generate these partial plans, and the action activator executes them.

In the virtual conference room example, *Room* has a *Sensor* of citizens' avatars. For an example of inter-agent communication, assume that the conference room contains the maximum of 20 citizens before citizen Greg enters. The room's *Sensor* constructs sense-data corresponding to "Greg is in the room" and a *Perception* process interprets this with the percept "the number of people in the room is 21". The *Conception* process uses forward chaining on percepts and expectations to recognize that the room is too small. The *Hypothesizer* then identifies goals such as "make the room bigger", "eject one citizen", "lock the room to additional avatars" and selects the goal to make the room bigger. The *ActionActivator* recognizes that it does not have an effector that can make the room bigger so it sends a message to the society to call for proposals to achieve the "make the room bigger" goal. The following message is sent by agent *Room* to other agents to call for proposals on making itself larger with respect to citizen Greg:

```
<performative type="call-for-proposal">
    <sender name="Room"/>
    <content>
        <predicate preposition="larger" object="Room" reference="Greg"/>
    </content>
</performative>
```

This message is a "call-for-proposals" performative type. The Room is the sender and any agent can reply. The content of the message is a belief of the Room: the

Room has a belief that the Room needs to be larger. The interaction diagram illustrating how this inter-agent communication propagates through the society is shown in Figure 4.

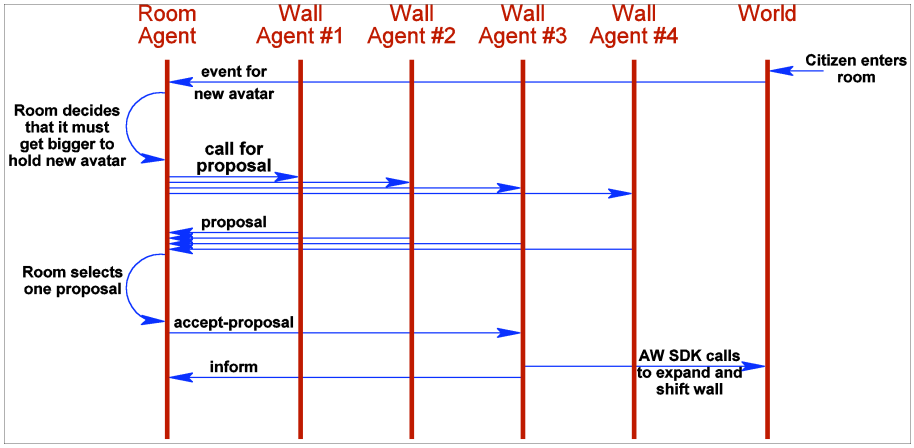

Figure 4 Interaction diagram showing the communication within the virtual conference room society when a new avatar enters the Room; AW SDK= Active World's Software Development Kit.

The question of what to communicate between agents is of particular interest. Should *Room* say to the *Wall*s "I would like someone to change so that I am larger", or "I would like someone to move such that the position of object X which is now outside becomes inside", or should it say to a particular wall "I would like you to move in direction D"? The first requires that both the room and walls know how to calculate room size, the second exposes the rooms goals, and the third has the room telling the wall too directly what to do. For the illustration described in this paper we adopted the first: a call for proposal requests that *Room* get bigger, with *object* and *reference* providing enough information that each *Wall* can submit a sensible proposal. Thereafter one proposal will be accepted, activated, and the room will get bigger. Suppose that one wall agent's proposal to move is accepted. That wall sends messages to the agents in its *Society* that it is moving and the other wall agents can either respond by lengthening themselves to maintain the connection with the wall that is moving or by leaving a gap.

4 IMPLEMENTATION

We have implemented the agent society as a software system with an interface to the Active Worlds server. Active Worlds provide a software development kit for Microsoft Windows platforms that is used to program bots that can enter the world as software controlled avatars and interact with human controlled avatars and

objects. Using the SDK directly means writing a C/C++ program to interface to a dynamic link library.

The procedure for inserting a new 3D object to an AW world is to copy an existing object, move it as required, and edit a dialog box to configure it. This is shown in Figure 5. It is reminiscent of the component based approach to building java programs that is facilitated by the use of java beans. In designing and implementing the agent framework, we want to preserve the object-oriented approach to copying an existing agent and editing it to configure it as a new agent. Our long term aim is to be able to insert and configure an agent from components with a minimum of programming.

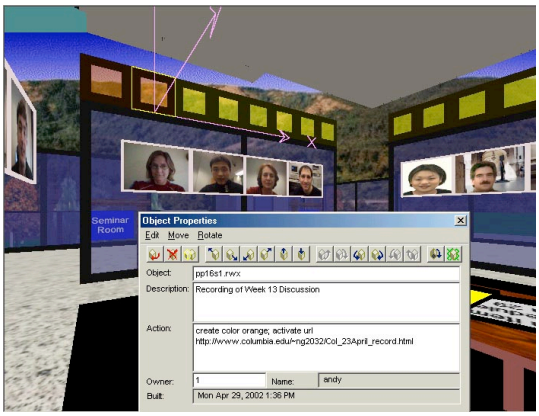

Figure 5 Inserting an object into an AW world.

A *Society* of agents is loaded from an XML configuration that specifies the agents and their sensors, effectors, and the file that specifies the agent's rules for reasoning about percepts, concepts, goals, and action activation. Classes that comprise an agent and society are not fixed at compile time. This approach has an added advantage in that we should, eventually, be able to stop, reconfigure and start agents without needing to recompile or restart the AW server. To make this work we are building a library of components that can be used to configure an agent.

We implement *Sensors* and *Effectors* in java using a Java Native Interface. This encapsulates of the SDK within a set of standard configurable components. Agents see only *SenseData* and effect-data corresponding to *Sensor*s and *Effector*s. It will mean that world builders can write agents using higher level languages. An important question is whether agents need to reason using AI techniques such as forward chaining, or whether coding directly in java is sufficient. Currently we have an agent class that uses the rule-based language Jess[1] for forward and backward chaining. This rule-based agent uses component *Sensor*s written in java and asserts

[1] http://herzberg.ca.sandia.gov/jess/

java bean sense-data into Jess working memory, and *Effectors* are driven from rules.

We are currently writing a component library of standard sensors and effectors. Some examples include

- *AWChatSensor* , which receives chat text from a world.
- *AWAvatarSensor*, which senses avatar additions, deletions and changes.
- *AW3DObjectSensor*, which identifies what AW 3D objects are near the agent.
- *ACLSensor,* which senses agent communication from the society.
- *AWChatEffector*, to send chat text into an AW world.
- *AW3DObjectEffector*, to add, change or delete a 3D object in a world.
- *ACLEffector*, which sends agent communication to the society.

Existing implementations of the FIPA ACL are targeted at distributed systems of agents. In our agent society package we have encapsulated the ACL within *ACLSensor* and *ACLEffector.* The agents communicate using XML. We use our own simple XML grammar, whilst adopting the FIPA ACL abstract message structure (FIPA 2001a) and semantics (FIPA 2001b). We chose XML because any XML parser can parse any XML data, and because anything for which a grammar can be defined can also be encoded in XML (Decker et al. 2000).

5 CONCLUSION

A virtual world as a society of agents provides the means for constructing flexible, dynamic worlds that adapt to users needs. The framework described will, when, fully implemented, enable "non-programmers" to construct such worlds by configuring a library of components. The framework can be extended to reason with various paradigms, as has been demonstrated with the use of Jess. Our agents communicate with each other and with an AW world, and collectively they provide desirable behaviours. Such communication is facilitated without affecting the performance of non-agent objects in that world. As a consequence virtual worlds become adaptable based on their use.

Acknowledgements

This work is supported by a grant from the Australian Research Council.

REFERENCES

Austin, J. L. (1962). *How to do Things with Words*. London: Oxford University Press.

Cohen, P. R. & C. R. Perrault. (1979). Elements of a plan based theory of speech acts. *Cognitive Science* 3(3): 177–212.

Decker, S., S. Melnik, F. van Harmelen, D. Fensel, M. Hlein, J. Broekstra, M. Erdmann, and I. Horrocks. (2000). The semantic web: The roles of XML and RDF. *IEEE Internet Computing* 4(5): 63-74.

Ferber, J. (1999). *Multi-Agent Systems: An Introduction to Distributed Artificial Intelligence*. Harlow: Addison Wesley.

FIPA. (2001a). *FIPA ACL message structure specification.* Document number XC00061E. http://www.fipa.org

FIPA. (2001b). *FIPA content language library specification communicative act library specification.* Document number XC00037H. http://www.fipa.org

Gamma, E., R. Helm, R. Johnson, and J. Vlissides. (1994). *Design Patterns: Elements of Reusable Object-Oriented Software*. Reading, MS: Addison Wesley.

Gero, J. S. and H. Fujii. (2000). A computational framework for concept formation for a situated design agent. *Knowledge-based Systems* 13:361-368.

Huhns, M. B. and L. M. Stephens. (1999). Multiagent systems and societies of agents. In *Multiagent Systems: A Modern Approach to Distributed Artifcial Intelligence*, ed. G. Weiss, 79-120. Cambridge, MA: MIT Press,.

Jennings, N. R. (2000). On agent-based software engineering. *Artificial Intelligence* 117:277-296.

Maher, M. L., and J. S. Gero. (2002). Agent Models of 3D Virtual Worlds. *ACADIA* (to appear).

Wegner, P. (1997). Why interaction is more powerful than algorithms. *Communications of the ACM* 40 (5): 81-91.

Searle, J. R. (1969). *Speech Acts.* Cambridge: Cambridge University Press.

Smith, G. J. and J. S. Gero. (2002). Interaction and experience: Situated agents and sketching, *in* JS Gero and F Brazier (eds), *Agents in Design 2002*, Key Centre of Design Computing and Cognition, University of Sydney, Australia, pp. 115-132.

Wooldridge, M. (1994). This is MyWorld: The logic of an agent-oriented DAI testbed. In *Proceedings of Intelligent Agents, Workshop on Agent Theories, Architectures and Languages, LNAI890*, ed. M. Wooldridge & N. Jennings, 160-178. Springer-Verlag.

Wooldridge, M. (1999). Intelligent Agents. In *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*, ed. G. Weiss, 27-77. Cambridge MA: MIT Press.

Wooldridge, M. (2002). *An Introduction to MultiAgent* Systems. Chichester, England: John Wiley and Sons.

This is a copy of the paper: Maher, ML, Smith, G and Gero, JS (2003) Designing 3D virtual worlds as society of agents, *CAADFutures* (to appear)