Designing Interactive Multi-agent Systems with Semantic 3D Environments

Zhiqiang Gao

Department of Computer Science and Engineering, Southeast University, China
gao_zhiqiang@yahoo.com

Abstract

3D virtual spaces hosting multi-agents are constructed by shapes, terrains, paths, textures, colors, lights and fogs in the scene. These 3D models and features can be displayed by browsers just as HTML by IE, and perceived by *human users* easily. However, it is difficult for *agents* to interact with 3D objects without *semantic annotations* such as contents, design purposes and effectives. This is especially true for interactive multi-agent system design, because environments and scenarios of agents are often specified by different humans. So, we address the following issues in this paper: 1) How to design interactive multi-agent systems? 2) What kind of semantic information is required for 3D environments annotation? 3) How to annotate 3D environments?

1. Introduction

Interoperability among agents and agent systems has been intensively studied by FIPA (www.fipa.org) and Agentcities (www.agentcities.org), which facilitate the inter-working of agents and agent systems across multiple vendors' platforms (Dale, Willmot and Burg, 2002). Ishida (2002a, 2002b) has emphasized the interaction design between humans and agents by introducing scenarios to describe a human request to a large number of agents with different roles. The scenarios also establish a bridge between agent designers and application designers (Zhiqiang, 2002a and 2002b). The Semantic Web brings structure to the meaningful contents of Web pages, creating an environment where software agents roaming from page to page (Berners Lee, Hendler and Lassila, 2001. DAML+OIL Revised Specification, 2001). However, little attention has been paid to semantic annotation of non-text information, such as video, audio, image and graphics. The goal of MPEG-7 standard is to develop a rich set of standardized tools to enable both humans and machines to generate and understand audiovisual descriptions (Martinez, 2001). The idea of "inverse causality" was introduced whereby objects in the environment told an animated agent how it should interact with them (Goldberg, 1997). Doyle (2002) introduced the concept of annotated environment in text-based MUD world, an environment which contains structured representations of its content and its purpose, authored by the creator of the environment.

The idea of this paper comes mainly from "inverse causality" of Goldberg (1997). However, our concern is 3D environments, which are much more difficult to tackle with when compared to texts. The following of this paper is organized as: We first design an interactive anti-terrorist shooter training simulation system, discuss the requirements for 3D space annotation, and then illustrate example ontology visually. At last, part of the implementation for the multi-agent system is presented.

2. Designing Interactive Multi-agent Systems

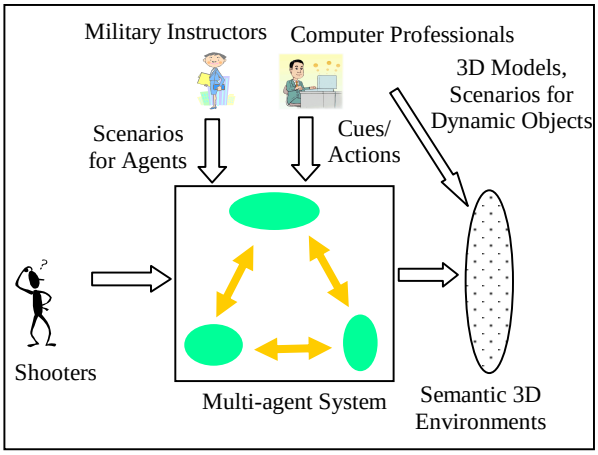

Figure 1. Interactive multi-agent system for anti-terrorist shooter training simulation

Environment creators and scenario designers of interactive multi-agent systems are often different people. Given an example of Digital City Kyoto (Ishida, 2002a), it is computer professionals and architects to build the 3D model of Kyoto Station, and social scientists do evacuation simulation by specifying scenarios for agents. In the case of shooter training simulation, researchers construct 3D spaces and implement *cue*s and *action*s (which are primitives of *Q* language, see Ishida 2002a) of terrorists and pedestrians (agents), including fire, jump, move, hear, etc. The purpose and effectives of objects in the scene are annotated in order for agents to interact with smartly. The

scenarios of agents are designed by instructors, and may by modified again and again. Shooters walk and fire in real world with their positions and intentions recognized via computer vision technique. See Figure 1.

Describing Scenarios for Agents

An indoor anti-terrorist shooter training simulation process is described below. An elevator runs up and down with neon light indicating floor. After it reaches the first floor, its door slides open and a target (passenger) will drop off. A shooter in real world then asks him to show his identification card. The target either shows it, or pulls out a pistol to fire at the shooter. The scenario of the target is specified by military instructors, as shown in Figure 2. Note that corruption of target scenario execution due to shooter firing is treated by its reactive layer.

```
(defagent terrorist
            :scenario 'Fire
            :weapon: 'PISTOL
            :sensitivity 'HIGH
            :position 'Elevator
            :orientation 'Door
            :gentle 'MALE
            :cloth BLACK
            : trouser GREY)

(defscenario fire
 (scene1 ((?see :what '(door opened))
           (!walk :direction 'door) (go scene2))
          (otherwise
           (go scene1)))
 (scene2 ((?hear :what "Show your identification card!")
           (!fire :at 'shooter)
           (!finish)))
          (otherwise
           (!walk :direction 'door)
           (!finish))))
```

Figure 2. Definition and scenarios for agents

Requirements for Annotating 3D Environments

Objects in 3D virtual spaces are classified as dynamic objects and static objects. The appearances and shapes of dynamic objects change without the interaction of agents and avatars. For example, the neon light of the elevator changes its number in time sequence. The door of the elevator has two states as opened or closed, and become opened after the elevator gets its level. Dynamic objects are controlled by scenarios like agents. Static objects have only one state, including ground of the elevator, identification card and pistols. They may change their positions passively by following the action of agent or due to gravity, and semantic annotations are mainly designed for them.

Annotations are characterized along three high-level dimensions similar to Doyle (2002).

{descriptive, directive}

Descriptive annotations provide factual information with no indication about how that information should be employed, and directive annotations provide the agent with an explicit recommendation with regard to its behavior. The geometry of the door of elevator is descriptive, with its purpose for agent to get out being directive.

{object, relationship, operation}

An object is primitive element in the virtual space such as identification card and pistol. Relationships can exist between objects. For example, the elevator unit consists of door and side walls. Operations are actions that can be performed by the agent, which are mainly annotated for collision check.

{content, context}

Content describes elements and happenings in the environment, without providing any underlying justification for why these annotations are present or how they should be employed, such as current action of the shooter. Context annotations indicate that certain knowledge is only relevant in some circumstances. A pistol can be fired only after it is raised by the shooter.

Annotating 3D Environments

As mentioned above, we take indoor anti-terrorist shooter training simulation as an example, which happens near an elevator. Its ontology is annotated by DAML+OIL and shown visually in Figure 3. There are six static objects in 3D spaces, which are the ground, the four side walls, an identification card and a pistol. There are two dynamic objects in the situation, which are the door and the neon light of the elevator. Geometry properties are designed for collision check, and purpose properties are annotated for the agent to act, such as the purpose of the door is for agent to drop off the elevator, with the ground for it to stand on.

3. Implementing Interactive Multi-agent Systems

Cues and Actions for Agents

Human-like agents of terrorists and pedestrians move realistically, respond to scenarios specified, and travel about the environment as directed. In order to simplify the task of adding life-like human characters to real-time interactive simulations and allow instructors to concentrate on telling agents where to go and what to do, we use *Q* scenarios to constrain the behavior of agents. *Cue*s and *action*s for agents are summarized as:

Cues

?hear	?see

Actions

!stand-ready	!kneel-ready	!prone-ready
!stand-aim	!kneel-aim	!prone-aim

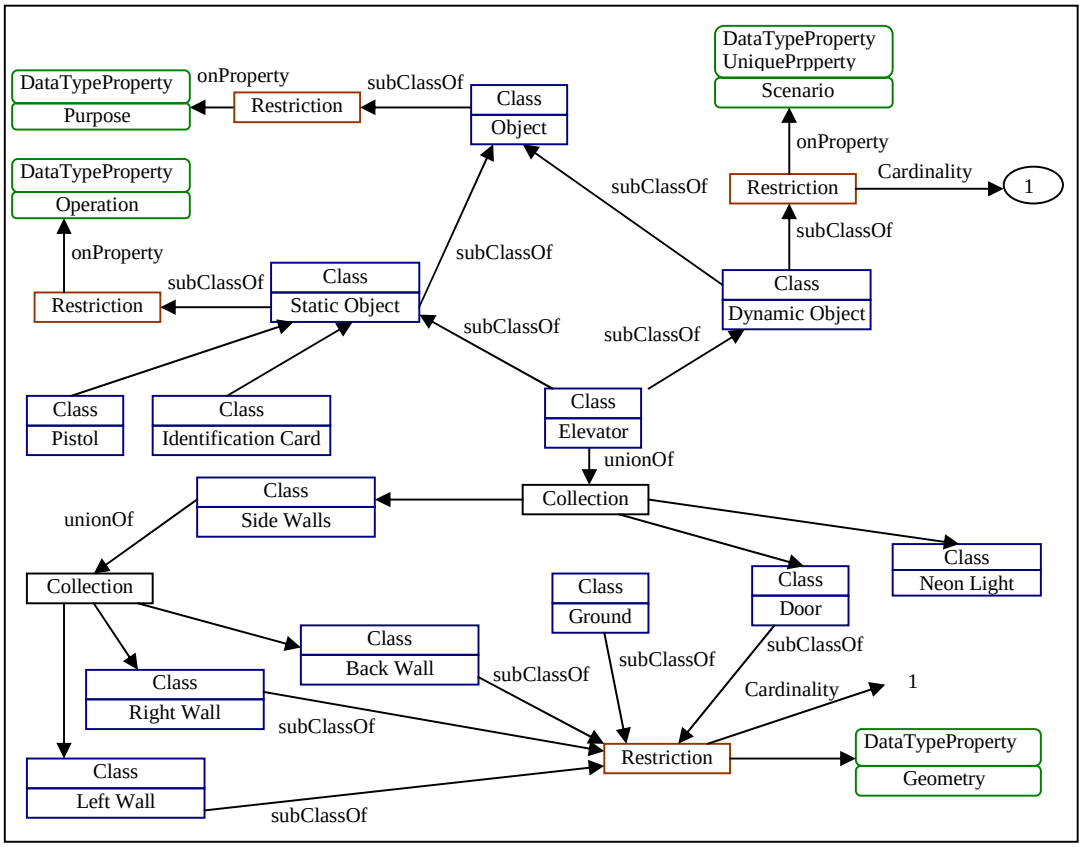

Figure 3. Part of the elevator ontology for anti-terrorist shooter training simulation

!walk	!crawl	!jog
!dead	!fire	!run
!stand	!speak	!finish

Constructing 3D Models

The visual database of 3D models is saved in MultiGen-Paradigm's OpenFlight (.flt) file format, which has become the standard file format for most realtime systems. The database hierarchy that the OpenFlight format uses has two main purposes: it organizes geometry into nodes that can easily be edited and moved, and it provides a tree structure that the runtime system can process. A node is the fundamental element or building block for constructing the database hierarchy. Beginning with the database (DB) node, a general database structure follows this order:

DB Node: Contains descriptive information about the entire database itself. Only one DB node can be at the top of the structure.

Master Groups: A master group node represents the entire model. All component hierarchies are ultimately attaches to this common point.

Significant Groups: A number of group nodes, each representing major components of visual database.

Objects: Objects can only contain face nodes (polygons), and are used to mark single objects with no moving parts.

Faces: Face nodes are easily recognized because they are always drawn in the color of their respective face as viewed. Faces can be attached to groups, objects, or other faces.

Vertex: Vertex attributes are characteristics of face attributes and have no hierarchal significance.

After 3D models have been constructed, semantic annotations might be added to nodes as comments in OpenFlight format, such as classes, design purposes, contents, and so on. When agents behave in the virtual spaces under the constraint of scenarios, they can interact with 3D models smartly. However, we are still on the way to build such an intelligent interactive multi-agent system.

4. Conclusions

We discussed the approaches in designing interactive multi-agent systems, including interaction between human users and agents which is described by *Q* scenarios, the interoperation among agents which is standardized by FIPA and Agentcities, as well as Web contents annotation studied by Semantic Web. To enable agents to act effectively and plausibly in 3D virtual spaces we introduce semantic annotation to objects in addition to geometries, such as design purposes, operations and contexts. An indoor anti-terrorist shooter training simulation system is designed, and example ontology annotated by DAML+OIL is shown.

References

1. Berners Lee T., Hendler J. and Lassila O.: The Semantic Web: A new form of Web content that is meaningful to computers will unleash a revolution of new possibilities. *The Scientific American*. 2001

2. Dale, J., Willmot, S., Burg, B.: Agentcities: Challenges and Deployment of Next-Generation Service Environments. *Pacific Rim International Workshop on Multi-Agents (PRIMA 2002)*, Tokyo, Japan, 2002

3. DAML+OIL Revised Language Specification, march 2001. <http://www.daml.org/2001/03/daml+oil-index/>

4. Doyle P.: Believability through Context: Using “knowledge in the world” to create intelligent characters. *AAMAS02*, pp342-349. Bologna, Italy, 2002.

5. Goldberg A.: IMPROV: A system for real-time animation of behavior-based interactive synthetic actors. *Lecture Notes in Computer Science*, 1195, 1997.

6. Martinez J.: "Overview of the MPEG-7 Standard (version 5.0)". *ISO/IEC JTC1/SC29/WG11 N4031*, Singapore, March 2001

7. Ishida, T.: Q: A scenario Description Language for Interactive Agents. *IEEE Computer*, Vol. 35, No 11, pp54-59, 2002

8. Ishida, T.: Digital City Kyoto: Social Information Infrastructure for Everyday Life. *Communications of the ACM (CACM)*, Vol. 45, No. 7, pp76-81, 2002

9. Zhiqiang, G., Tomoyuki, K., Akishige, Y., and Ishida, T.: Meta-Level Architecture for Executing Multi-agent Scenarios. *Lecture Notes in Artificial Intelligence*, 2413, Springer-Verlag, pp163-177, 2002

10. Zhiqiang, G., Arai S. and Ishida T.: Interoperability and Interaction Design, to appear in *Lecture Notes in Artificial Intelligence*, 2002