ICARUS: Intelligent Classification And Retrieval of Unlabelled Scenes

Emilio Domenicucci[1] and **Francesco M. Donini**[1,2] and **Marco Schaerf**[1]
1: Dipartimento di Informatica e Sistemistica, Università di Roma "La Sapienza",
Via Salaria 113, I-00198 Roma, Italy
email: {domenic,donini,schaerf}@dis.uniroma1.it

2: Dipartimento di Elettrotecnica ed Elettronica, Politecnico di Bari,
Via Re David 200, I-70125 Bari, Italy
email: donini@poliba.it

Abstract

In this paper we present ICARUS: a tool for the creation, classification and retrieval of VRML scenes. ICARUS classifies each scene using a concept description language and uses its reasoning mechanism to retrieve information.

1 Introduction

In recent years multimedia databases are becoming more and more widespread. Image and video databases are now widely available and used. One of the main difficulties in setting an appropriate framework for storing complex objects in databases is the necessity of having an appropriate description and query language (see, e.g., [8]). Many other papers, (for example [5, 1, 7]) focus on the importance of having appropriate description and querying mechanisms to describe and retrieve multimedia information.

In this paper we present a prototype system (called ICARUS) that uses a logical language to describe the properties of three-dimensional scenes and, through this language, is able to classify the scenes and retrieve them using a rich query language. In particular, our system aims at creating, classifying and retrieving 3D scenes modeled using the Virtual Reality Modeling Language (VRML). We do not require the user to completely specify all properties of the scenes that are created. Our system is able to extract some info from the scene and to accurately classify it using knowledge of the domain. For this reason we call our scenes "unlabelled".

The goal of the project is to show that rich object description languages can be very useful in "intelligently" classifying and retrieving 3D scenes. At the current status of development, ICARUS only focuses on a simple domain, where we have a (non-fixed) set of characters, a number of relations among these characters, a set of scenarios, and a set of objects. Initial knowledge on the domain is stored into a knowledge base (KB) modeled using the concept description language `CLASSIC`. The KB describes the characters and objects that appear in the scenes, their properties and their relations with the other objects.

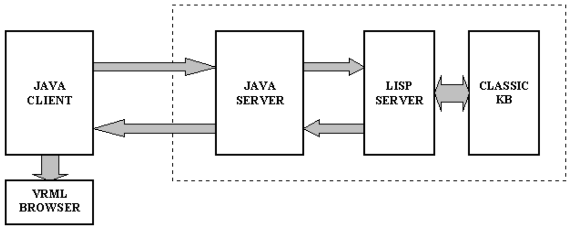

Figure 1: Architecture of ICARUS.

In a nutshell, the system capabilities are the following ones:

1. Create a new VRML scene using a friendly Web interface;
2. Create new characters and modify the existing ones;
3. Classify each new object introduced (scene, character or other object);
4. Query the KB for a desired scene and present it to the user.

The system is developed using Java for the client and the interface of the server and LISP for the interface to the `CLASSIC` KB.

We note that when systems based on much more expressive DL's (e.g., [3]) will be available, we will adopt them in place of `CLASSIC`, adding more knowledge-management capabilities to ICARUS.

2 ICARUS : Architecture

The ICARUS system uses a client/server architecture based on the idea that a computer specialized in the visualization of 3D scenes can display data stored and elaborated on a remote host connected via Internet. In our situation information to be shown are VRML scenes and stored data are `CLASSIC` descriptions of the scenes.

The architecture of the system is presented in Figure 1. We can subdivide the system in 5 software components:

1. an HTML BROWSER (Netscape Navigator or Internet Explorer) with a plug-in (Cosmoplayer 2.0+) which can display VRML scenes,
2. a JAVA CLIENT that opens a socket connection with the JAVA SERVER and implements the user interface,
3. a `CLASSIC` KNOWLEDGE BASE (KB) defining the structured concepts, their instances and allowing their creation and manipulation,
4. a LISP SERVER controlling the interactions of KB with the other components,
5. a JAVA SERVER placed between the JAVA CLIENT and the LISP SERVER that translates user queries and builds VRML scenes according to the information received from KB.

Initially the JAVA SERVER is running, connected to the LISP SERVER and waiting for a socket connection request. When the CLIENT program is launched, a new connection is established and data exchange can start.

The user interface is composed of two parts: the first is a window used to display VRML scenes, the second is a control panel used to choose the kind of interaction with KB. It is also present a text area where service messages are shown to help user controlling the transaction state.

The control panel allows the user to set queries in order to retrieve information stored in the KB.

A *"SET NEW QUERY"* button displays on the screen a group of panels used to select items, basic query elements. We can request KB to look for all scenes

- whose environment is a particular room (kitchen, sitting room ...) or a place in the garden (park bench, gazebo),
- with specific characters (Alice, Bob, Charlie, ...) or a certain number of them,
- with characters having certain characteristics (dressing in a particular way, sitting, wearing glasses, ...)
- in which there are some objects (tables, books, plants, lamps, paintings, ...)
- where characters satisfy some relations (all friends, all relatives, all colleagues, ...)
- that can be classified as "family meeting", "job meeting", "romantic scene", etc.

When research criteria are set, a *"SEND QUERY TO SERVER"* button sends them to JAVA SERVER that composes them in order to obtain a query in the `CLASSIC` syntax to be submitted to KB. The KB answer goes back to CLIENT as a list of scenes and the user chooses the one to display with a *"CONFIRM SELECTED SCENE"* button.

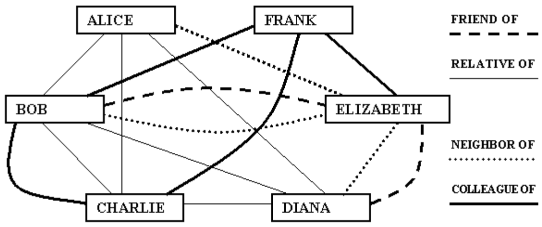

Figure 4: Relations among Characters.

Before displaying the selected scene the JAVA SERVER requests the KB all the *"fundamental elements of the scene"*, that are: the environment in which the scene happens, the characters involved and information about their position (standing or sitting). If the choice criteria cause the creation of an unsatisfiable query the LISP SERVER intercepts the warning message from `CLASSIC` KB and sends it to CLIENT.

At this point a *"DISPLAY SCENE"* button tells the JAVA SERVER to transform all collected information about the scene in a VRML description; to achieve this, the server asks the KB the new view point and all details about the fundamental elements of the scene (objects colors, characters coordinates, characters attitudes, ...). Finally, the VRML description is sent to the VRML BROWSER that displays it.

Using the interface, it is also possible to create new characters and new scenes that are added to the KB.

3 ICARUS: examples

Currently, ICARUS contains a KB describing scenes that take place in a mansion and involves a number of objects and characters. Diagrammatically, our KB can be presented as in Figure 2.

Notice that the VRML files are built using this hierarchy. In fact, we have prestored a set of VRML that define all the characters and objects present in the scenes. The simple objects are completely defined, while the complex ones are defined via a PROTO with parameters. In this case, the parameters are extracted at run-time from the `CLASSIC` KB.

The concept `scene` is further refined as in Figure 3. Notice that the relations expressed by `CLASSIC` are more complex than just the class-subclass relation. This hierarchy only represents a simplified view of the knowledge present in our KB. Furthermore, our KB uses an extended set of constructs, built using `TEST` concepts, that is not among the standard set of `CLASSIC` constructs.

All the characters present at the beginning of a session are: `Alice`, `Bob`, `Charlie`, `Frank`, `Diana` and `Elizabeth` and the relations among them are presented in Figure 4.

We now present some examples of use of our system. In our first example, let `C1`

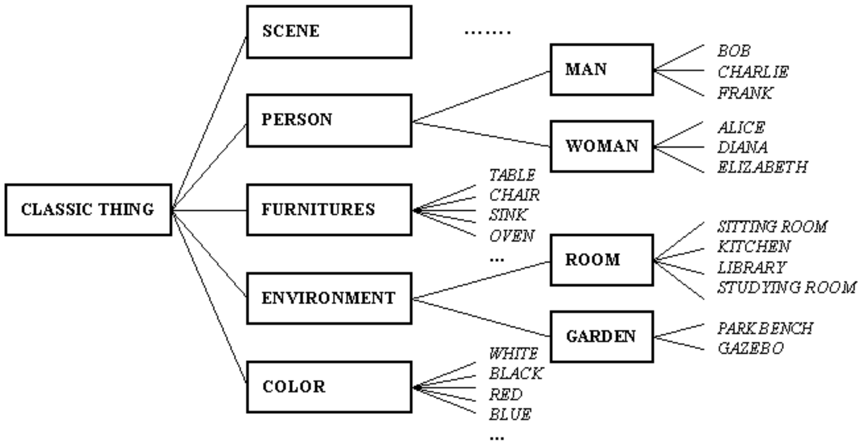

Figure 2: Hierarchy of concepts (within rectangles) and individuals.

be the `scene-family-meeting` concept, defined as (`AND scene-at-least-3-characters scene-all-relatives scene-sitting-room`), and `C2` be the `scene-Diana` concept. If we ask the KB to look for all scenes satisfying (among others) both concepts `C1` and `C2`, KB will answer that no scene (at present) satisfies the query. If user creates a new scene with characters: `Diana`, `Alice`, `Bob` (all relatives), environment: Sitting room, and asks the same query again, the new scene will now be returned.

Let `C3` now be `scene-romantic` concept, defined as (`AND scene-2-characters scene-all-friends scene-at-least-1-man scene-at-least-1-woman scene-park-bench scene-at-least-1-sitting`). When we send the KB a query having both concepts `C1` and `C3`, a warning message is returned because the query is *unsatisfiable*: `C1` and `C3` are disjoint concepts – an individual can't belong to both of them – because of the number of characters they require (`C1` is classified under the `scene-at-least-3-characters` concept and `C3` under the `scene-2-characters` concept). Thus the query is recognized to be *incoherent* and no attempt to retrieve any scene is done.

If instead of `scene-romantic` we define a new concept `C5` to be `scene-all-friends`, the query (`AND C1 C4`) becomes satisfiable and the scene in Figure 5 is returned. Note that if a scene is classified under the `scene-all-relatives` concept then it is also classified under the `scene-all-friends` concept. In fact, the KB contains a role hierarchy where the role `friend-of` is more general than `relative-of` and `colleague-of` roles.

Another situation causing a warning message from KB is when the user tries to create an inconsistent individual, for example a character with the following definition:

(`AND woman (FILLS hair-color Brown) (FILLS tshirt-color Red)` ... `(FILLS beard-color Black)`).

Inconsistency is due to the attempt of introducing an individual `i` belonging to the concept `woman`, and having a filler for the role `beard-color`. This causes an inconsistency because the concept `woman` contains in its definition `AT-MOST 0 beard-color`, while the system deduces that `i` also belongs to the concept `AT-LEAST 1 beard-color`.

4 Perspectives

In this paper we have presented a prototype system that helps in the creation, classification and retrieval of (simple) VRML scenes. In our system the emphasis is on the representation of the logical properties of scenes and their use for classification and retrieval.

Many other systems aim at classifying complex data (e.g. images) by the use of logical relations. For example, in [4] Corridoni et al. extract from an image a set of relations based on the color and these relations are then used for retrieval. At the present stage of development of our system, the base relations that a scene satisfies are entered by our scene editor. In the future we aim at being able to automatically extract some of the relations directly from the description of a scene in VRML format.

While we are not aware of any other system that uses a concept description language, such as `CLASSIC`, for the classification of 3D scenes, it is worth mentioning the system ALFRESCO [6] that uses YAK (another concept description language) for accessing information on 14th century Italian frescoes and monuments. Recently, in [2] a description logic to model and query video contents has been proposed; however, it is not clear which are the computational properties of the description logic proposed, nor a system implementation is presented.

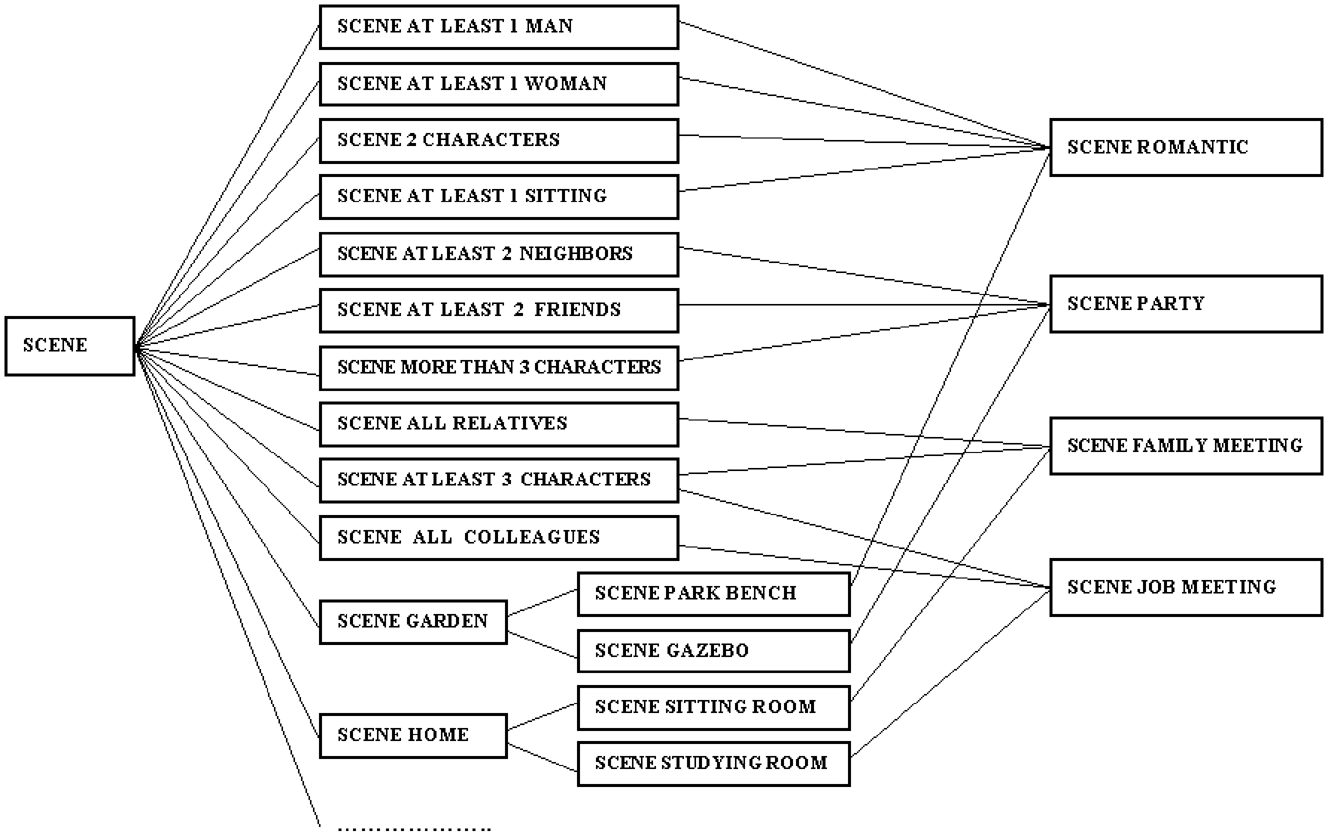

Figure 3: Hierarchy of scene concepts. Note the multiple inheritance.

There are several research issues that need to be addressed in the near future:

1. Classification of arbitrary VRML scenes. At the moment, our system can only classify the scenes that are built via our interface.
2. Description and classification of dynamic VRML scenes (containing events). An important aspect of VRML scenes is their dynamic behavior. Since this behavior is expressed via an event model, we are currently working on a logical language that can express the dynamic relations between objects.

References

[1] D. A. Adjeroh and K. C. Nwosu. Multimedia database management: Requirements and issues. *IEEE MultiMedia*, 4(3):24–33, 1997.

[2] Edoardo Ardizzone and Mohand-Said Hacid. A Semantic Modeling Approach for Video Retrieval by Content. In *Proc. of the IEEE Int. Conf. on Multimedia Computing and Systems*, June 1999. (To appear).

[3] Diego Calvanese, Giuseppe De Giacomo, and Maurizio Lenzerini. Structured objects: Modeling and resoning. In *Proc. of DOOD'95*, number 1013 in LNCS, pages 229–246, 1995.

[4] J. M. Corridoni, A. Del Bimbo, S. De Magistris, and E. Vicario. A visual language for color-based painting retrieval. In *Proc. of IEEE Symposium on Visual Languages*, pages 68–75, 1996.

[5] K. C. Nwosu, B. Thuraisingham, and P. B. Berra. Multimedia database systems: A new frontier. *IEEE MultiMedia*, 4(3):21–23, 1997.

[6] O. Stock et al. Alfresco enjoying the combination of natural language processing and hypermedia for information exploration. In Mark T. Maybury, editor, *Intelligent Multimedia Interfaces*, pages 197–224. 1993.

[7] P. Pazandak and J. Srivastava. Evaluating object DBMSs for multimedia. *IEEE MultiMedia*, 4(3):34–49, 1997.

[8] V. S. Subrahmanian. *Principles of Multimedia Database Systems.* Morgan Kaufmann, Los Altos, 1998.

Figure 5: Output of Query `C5`.