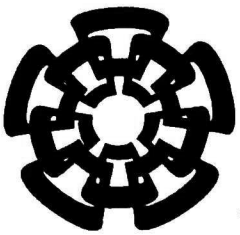
XX(131349.1)



CINVESTAV

Centro de Investigación y de Estudios Avanzados del I.P.N.
Unidad Guadalajara

Representación y Explotación del Conocimiento para la Fase de Descripción en Modelado Declarativo de Ambientes Virtuales

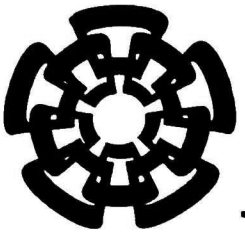
Tesis que presenta:
Jaime Alberto Zaragoza Rios

para obtener el grado de:
Maestro en ciencias

en la especialidad de:
Ingeniería Eléctrica

Director de Tesis:
Dr. Félix Francisco Ramos Corchado

Guadalajara, Jalisco, Septiembre de 2006.



CINVESTAV

Centro de Investigación y de Estudios Avanzados del I.P.N.
Unidad Guadalajara

Representation and Exploitation of Knowledge for the Phase of Description in Declarative Modeling of Virtual Environments

A thesis presented by:
Jaime Alberto Zaragoza Rios

to obtain the degree of:
Master in Science

in the subject of:
Electrical Engineering

Thesis Advisors:
Dr. Félix Francisco Ramos Corchado

Guadalajara, Jalisco, September 2006.

Representación y Explotación del Conocimiento para la Fase de Descripción en Modelado Declarativo de Ambientes Virtuales

Tesis de Maestria en Ciencias
Ingeniería Eléctrica

Por:
Jaime Alberto Zaragoza Rios
Ingeniero en Sistemas Computacionales
Instituto Tecnológico de Jiquilpan 1998-2003

Becario de CONACTY, expediente no. 190965

Director de Tesis:
Dr. Félix Francisco Ramos Corchado

CINVESTAV del IPN Unidad Guadalajara, Septiembre de 2006.

Representation and Exploitation of Knowledge for the Phase of Description in Declarative Modeling of Virtual Environments

Master of Science Thesis
In Electrical Engineering

By:
Jaime Alberto Zaragoza Rios
Engineer in Computer Science
Instituto Tecnológico de Jiquilpan 1998-2003

Scholarship granted by CONACYT, No. 190965

Thesis Advisors:
Dr. Félix Francisco Ramos Corchado

CINVESTAV del IPN Unidad Guadalajara, September, 2006.

Representación y Explotación del Conocimiento para la Fase de Descripción en el Modelado Declarativo de Ambientes Virtuales.

En esta tesis se presenta un estudio del uso del conocimiento en el proceso de generar ambientes virtuales de manera declarativa. Se encontró que hasta el momento el uso del conocimiento no ha sido utilizado en el proceso declarativo a pesar de que este juega un papel primario en este proceso. Una explicación posible es que el trabajo hasta hoy realizado se el usuario juega un papel importante en este proceso. Esto a pesar de ser natural puede ser fuente de errores, si el usuario no es experto o aun en este caso no hace las consideraciones pertinentes.

Este trabajo se presenta de la siguiente manera:

1. Se hace una revisión de cómo se realiza el modelado declarativo y el rol que tiene el conocimiento en este proceso.
2. De la revisión se hace la definición de un lenguaje de tipo pseudo natural que permite al usuario escribir oraciones vacías de ambigüedad.
3. Se estudia y decide la implantación de una base de conocimientos. En dicha implantación se hace hincapié en la estructura del conocimiento ya que esta es importante para el proceso total de generación declarativa de ambientes.
4. Finalmente se hacen las implantaciones pertinentes y se obtienen resultados útiles para concluir sobre la pertinencia de la presente investigación.

Use of Knowledge for Helping the Phase of Description for Declarative Modelling of Virtual Environments.

This thesis focuses on the process of generating environments in declarative manner. The proposal is original and makes use of knowledge to aid the performance of the description phase.

Some of the problems we deal in this work are:

1. How to write a declaration in such a way to reduce the analysis complexity. To solve this problem is proposed a like natural declarative language. Rules of this language allow reduce ambiguity of the descriptions;
2. Define how the knowledge is useful for describing a scenario;
3. Define a convenient structure for knowledge in order to have an efficient access to the necessitated information during the process of analysis of sentences and also during the creation phase.

Defined problems were solved and an implantation of the proposed solutions is presented. Finally conclusions regarding this work are described together with some directions for continuing this work.

Index

Chapter 1. Introduction

Abstract

In our first chapter, we exposed the reasons that lead us to develop a method for using ontology in the description phase of declarative modeling, the description of the problems we found and the solutions proposed for solving them.

1.1. Objectives.

Describe the problems this research will intent to solve, explain the solutions we proposed, highlighting the activities developed for reaching our goal, and the results we expect at the end of the research. Finally, we present the organization of the thesis.

1.2. Problem's Description.

Our focus on this research is the use of knowledge to aid in the generation of virtual environments. To generate such environments, we dispose of an array of tools and techniques. Some of those techniques require expert knowledge in 3D modeling, computer programming, or specialized hardware. Such tools present a challenge to casual users. The process for master such tools and techniques is long and complicated. The user must deal with mathematic and logic operations and the quality of the outcome vary from user to user, since some artistic abilities are necessary.

Such environments or virtual scenes are focused on simulating almost anything possible, from real settings to fantastic situations. Some of those scenes have a complex nature, so is necessary to create and model everything so it can work like we want. That requires specialized knowledge on modeling, both for the 3D representations, as for how each entity would evolve.

The use of such representations is wide, especially on the entertainment industry, where it has gained popularity on videogames and the film industry [MONZANI01]. Other applications for such tools are simulations (such as crime zones [LODHA99]), the construction industry [LARIVE], or even as a virtual story telling [GÖBEL03].

Our goal with the present research is to develop a friendly, easy to use tool for the final, non expert user (Figure 1). Thus, we focused the research on declarative modeling.

Definition 1. Declarative Modeling. is very powerful technique allowing to describe the scene to be designed in an intuitive manner, by only giving some expected properties of the scene and letting the modeler find solutions, if any, satisfying these properties [PLEMENOS02].

The declarative modeling process usually composes of three modules [LE ROUX03]:

- *Description phase*. Defines the interaction language.
- *Scene generation phase*. The modeler generates one or more scenes that match what the user describe.
- *Insight phase*. The user is presented with the models. The user then can choose a solution.

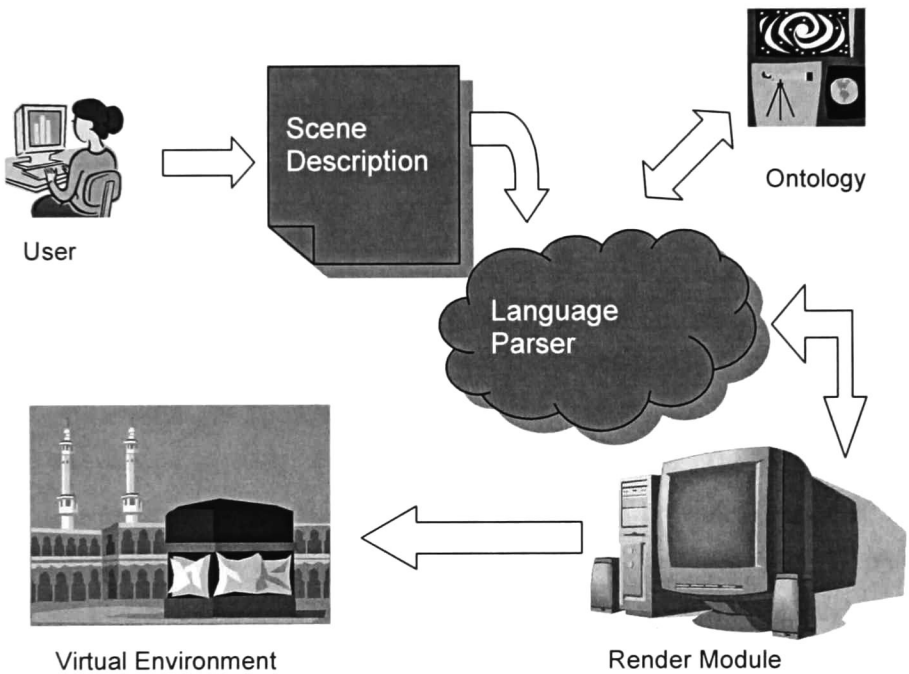


Figure 1. Project Overview.

The focus of this research is the description phase. We pretend to give to the user a tool that allows the creation of virtual scenes, that can evolve in accordance with the user expressed. Those tools could be a full Integrated Development Environment (IDE), a language specifically designed to allow the description of virtual scenes, or a parser that allows the analysis of a description written on natural language.

Each of those options presents different complexities. The IDE would need not only a parser for the description, but also a rendering machine and some type of knowledge base. For the language, we need to define both the grammatical and syntactic rules, and only the parser and the knowledge base are needed, but we relay on some rendering machine already constructed. Finally, the parser for a natural–

language written description would require only the parser and knowledge base, but the parser would be a lot more complex.

Since the tool should be easy to use, with a small learning step curve, we decided to take the language approach. If the language is close to the natural language, with few key words, with phrase construction resembling every–day–language, any user can learn it fast. The rendering machine would be provided by the GeDA-3D [RAMOS02] architecture, which also hosts the tools for evolving the scene. In addition to the definition of the language, a parser will be constructed so the descriptions based on it can be processed, and a knowledge base will be generated so the parser can be process the terms in the description and generated the appropriated output.

1.3. Goals.

Our fist goal is to define the lexical and semantic rules for the language. Such rules must take into consideration that the parser will use a knowledge base that defines the semantic meaning of the word being processed, and the whole meaning of the sentence being analyzed. The second goal is the creation of a parser that analyses the descriptions written in language previously defined, and then validated such description, indicating to the user for possible errors, and generating a previously defined output. Our final goal is to created a link between the parser and the GeDA-3D architecture, so the scene can be displayed and evolve.

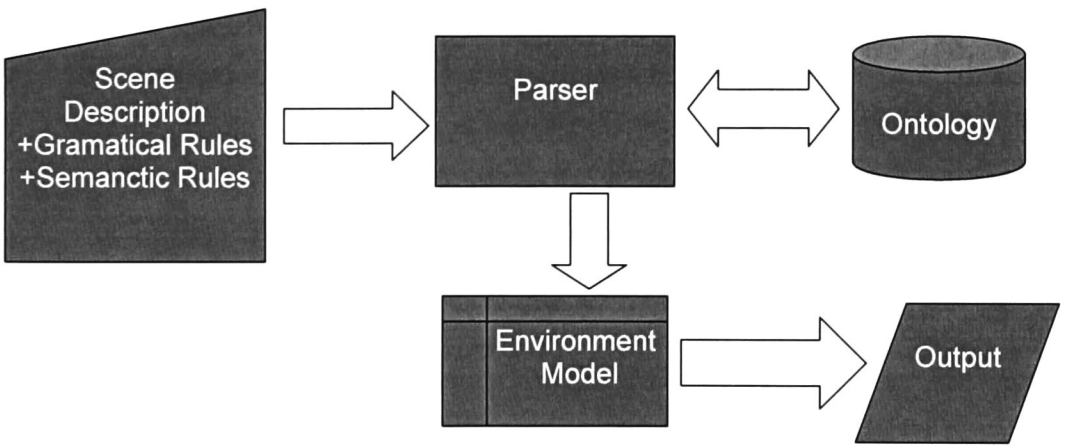


Figure 2. Basic Architecture

1.4. Solution Proposed.

To achieve the goal exposed before, we proposed the following considerations:

- The definition of the language would no consider a semantic approach, since every term must be revised within the knowledge base. This will allow expanding or restricting the language, by adding or subtracting elements from the knowledge base. Also, it allows the separation and identification of homonymous words.
- Also, the language needs to be both easy to learn and fast to write. It should not limit what the user can express. This can be accomplished by creating an ordered structure, rather than relay on the way a person normal would describe and scene.
- The parser should revolve around using the knowledge base for semantic purposes. The lexical and syntactic rules defined on the language can be hard-coded.
- An ontology is the election of the knowledge base. Such decision is analyzed and explained on the next chapters.
- The output will be generated for the rendering, agent control and core modules of the GeDA-3D architecture. Those outputs are bound to be defined by the needs of those modules, and the communications between those and the parser will be controlled by the architecture's core.

1.5. Thesis Organization.

This thesis organization is presented in the following lines:

- Chapter 2, State of the Art, describes previous and actual works on declarative modeling, as well as some literature on the subject.
- Chapter 3, Declarative Modeling, provides and scope on the concept and methods for declarative modeling.
- Chapter 4, Proposed Solution, states in detail our approach, the research conducted to validate such approach, and the selection of methods and data structures.

Chapter 5, Virtual Environment Definition Language, presents a declarative–oriented language, which enables non-expert modeler and casual user to generate virtual scenarios. Language specification and formal definition are presented in this chapter.

Chapter 6, Conclusions and Future work, offers the compilations of results acquired during the research, as well as proposed task for future approaching on the research subject.

Chapter 7 enumerates the references consulted for developing this research.

Chapter 2. State of the Art

Abstract

In this chapter we expose some concepts about virtual environments and natural language parsing. We also review some projects that match some of the concepts of the GeDA-3D project in the text-to-scene module.

2.1. Objective.

To provide some related concepts, and to present some previous projects and investigations that deal with the generation of virtual scenes (such as classrooms, offices, plane simulators, ecosystems, roads, among others) from descriptions and sentences in a natural language.

2.2. Introduction.

World and artificial environment modeling using 3D computer graphics is a difficult and time consuming process. Learning the complex software packages includes understanding menus, tools, parameters, files and others, which can be very complex to average users, with little or no knowledge of 3D modeling [COYNE01].

Creating a virtual environment by means of scene description would allow inexpert users to create such environments quickly, without the necessity to go through manuals and complex menus, or the underlying geometric representation [RUCHAUD02].

The creation of a 3D-animated scene as a result of a high-level description demands the development of a platform capable to: translate the description, compute a suitable solution for every intention with the help of intelligent algorithms, manage the interaction of the characters, validate the actions according to a set of natural laws, and render continuously a 3D-scenario [RAMOS02].

In order to make agile the generation of the 3D environment, the platform will be linked to a knowledge base, an anthology, which will include not only the details and descriptions of objects, but also the way they can related with each other, the positions they can adopt, the actions they can execute, movement or positioning restrictions, among others.

2.3. Virtual Environments.

One of the best descriptions for virtual reality we can give is: "Virtual Reality is a way for humans to visualize, manipulate and interact with computers and extremely complex data" [AUKSTAKALNISM92].

The visualization part refers to the computer generating visual, auditory or other sensual outputs to the user of a world within the computer. This world may be a CAD model, a scientific simulation, or a view into a database. The user can interact with the world and directly manipulate objects within the world. Some worlds are animated by other processes, perhaps physical simulations, or simple animation scripts [ISDALE98].

Some of the virtual reality applications are:

- Education
- Business
- Architecture
- Science
- Medicine
- Robotics
- Army
- Handicap tools
- Fly simulators
- Art
- Entertainment
- Sports and fitness

2.4. Avatar.

Typically, an avatar is defined as a representation of an actual person within a virtual environment, by means of software agents or controlled by a live participant. In this work, an avatar stands for any

virtual entity that can perform certain activity, or that can be modified by other avatar, objects, or the environment. Obviously avatars as expressed in our previous works have in its core intelligent algorithms or are controlled by a live participant. Thus for us, an avatar is any virtual entity that can perform certain activity, or that can be modified by other avatar, objects, or the environment.

2.5. Object.

An object will be considered, by the means of this work, as an avatar with no capabilities, which means they can not perform any activity, but can be modified by others avatar, or other objects. Thus for example a door, a wall, a table are objects. They can be modified by all type of avatars or by other objects. Imagine a ball that hits a lamp the state of this last was modified by another object.

2.6. Natural language parsing.

Most of the recent work in empirical natural language processing has involved statistical training techniques for probabilistic models such as HMMs (Hidden Markov Models) and PCFGs (Probabilistic Context-Free Grammars). These methods attach probabilities to the transitions of a finite-state machine or the productions of a formal grammar and estimate these probabilistic parameters based on training data. If the training set is pre-annotated with the structure being learned, learning consists simply of counting various observed events in the training data. If the corpus is not annotated, an estimation-maximization strategy could be used (for example, the forward-backward algorithm for Markov models and the inside-outside algorithm for PCFGs) [BRILL97].

Understanding natural language is a complex task and involves many levels of processing and a variety of subtasks. We can divide the field of natural language processing as follows:

Speech recognition. Concerns mapping a continuous speech signal into a sequence of recognized words. The problem is difficult because of the wide variation in the exact pronunciation of words spoken by different speakers in different contexts. Other problems include homonyms (for example, pair, pear, pare), other forms of acoustic ambiguity (for example, youth in Asia and euthanasia), and the slurring of words (for example, didja) that happens in continuous speech [BRILL97].

Semantic analysis. Involves mapping a sentence to some sort of meaning representation, for example, a logical expression. We have two important subtasks of semantic analysis: (1) word-sense disambiguation and (2) semantic parsing. Word-sense disambiguation roughly means deciding which of the possible meanings for a word is correct in a particular context. Part of semantic parsing involves producing a case-role analysis, in which the semantic roles of the entities referred to in a sentence, such as agent and instrument, are identified [BRILL97].

Discourse Analysis and Information Extraction. Discourse analysis involves determining how larger intersentential context influences the interpretation of a sentence. Information extraction is the task of locating specific pieces of data from a natural language document.

Syntactic analysis involves determining the grammatical structure of a sentence, that is, how the words are grouped into constituents such as noun phrases and verb phrases; is the process of assigning a "phrase marker" to a sentence. This is useful in determining the meaning of a sentence. But there are cases where a sentence can get different meanings. For most grammars (certainly for the ones statistical parsers typically deal with), we can not assign at least a semi-plausible meaning to all of the possible parses [CHARNIAK97].

Therefore, we required that the analyzer chooses the right one between all the possible solutions of a sentence. For this matter, we have some statistic methods.

2.6.1. Part-of-speech Tagging.

The disambiguation as inseparable from parsing is well illustrated by the first natural-language processing task to receive a thoroughgoing statistical treatment: part-of-speech tagging. A tagger assigns to each word in a sentence the part of speech that it assumes corresponds in the sentence. Each word is assigned with any of its probable parts of speech in order or frequency. A tagging algorithm then looks for the maximize sum of probabilities for a set of assigned tags to a sentence [CHARNIAK97].

This method has been proven to be useful in the EyesWorld project. The algorithm search for an appropriated tag for each word, so we propose use an ontology to make the algorithm faster, since every word in the ontology can include not only which tag should be used, but also the in what context that particular tag is right.

2.6.2. Statistical Parsing.

We started our discussion of statistical taggers by assuming we had a corpus of hand-tagged text. For our statistical parsing work we assume that we have a corpus of hand-parsed text. In this article we concentrate on two such measures labeled precision and recall, Precision is the number of correct constituents found by the parser summed over all the sentences in the test set divided by the total number of non terminal constituents the parser postulated. Recall is the number correct divided by the number found in the tree-bank version. We consider the parts of speech to be the terminal symbols in this counting and thus ignore them. Otherwise we would be conflating parsing accuracy with part-of-speech tagging accuracy. A constituent is considered correct if it starts in the right place, ends in the right place, and is labeled with the correct non-terminal [CHARNIAK97].

Statistical parsers work by assigning probabilities to possible parses of a sentence, locating the most probable parse, and then presenting that parse as the answer. Thus to construct a statistical parser one must figure out how to (a) find possible parses, (b) assign probabilities to them and (c) pull out the most probable one.

2.6.3. Lexicalized Parsing.

Gathering statistics on individual words immediately brings some sparse data problems. Some words we will have never seen before, and even if we restrict ourselves to those we have already seen, if we try to collect statistics on very detailed combinations of words, the odds of seeing the combination in our training data become increasingly remote.

To minimize the combinations to be considered, a key idea is that each constituent has a "head", its most important lexical item. For example, the head of a noun phrase is the main noun, which is

typically the rightmost. Lexicalized statistical parsers collect, to a first approximation, two kinds of statistics: One relates the head of a phrase to the rule used to expand the phrase, and the other relates the head of a sub-phrase to the head of a sub-phrase [CHARNIAK97].

2.7. Constraint Satisfaction Problems.

Once we have validated the elements in the description, their properties and capabilities for that particular environment, we need to establish the position of each element within the scene. We need to take into account the position and direction of every element, and the spatial properties that each object has.

Constraints are the language of design and as such, can be an extremely valuable tool when placed in the hands of a designer in the proper form. Constraints can provide a concise, declarative language for specifying geometry and the relationships between objects in graphic applications.

A constrain describes a declarative relationship that must be maintained among objects. These constraints are solved by automatically assigning the task of maintaining them to the constraints solver. This allows the user to concentrate on the relationships that need to be maintained, rather than on the techniques used to maintain them [KWAITER97].

However, the increase in the user's requirements adds special challenges to the research in constraints based systems: User may want to quickly explore and improve the subject of his design by adding or removing constraints. Thus, the constraints solver must re-satisfy the system dynamically as the user interactions occur. Moreover, the user quickly discards a system that slows him down or delays his work. So, the constraints solver must always give a fast answer even in critical situations such as over-constrained or under-constrained problems.

By using an ontology, we can resolve some of the constrains, by find which positions, orientations, and actions can an object adopt, as well as the possible areas they can occupy, or that can be occupied within the object.

2.8. Other works on declarative modeling.

There have being some works on declarative modeling. Some use direct text to scene conversion, others use specialized interfaces to allow the creation of the scene, and some relay on the complete construction of the scene by using simplified elements to created complex structures.

In the next paragraphs, we review some of this tools, with the purpose of highlight the differences between such works and our project.

2.8.1. WordsEye: An Automatic Text-to-Scene Conversion System.

Bob Coyne and Richard Asproad presents a system developed in the AT&T laboratories, WordsEye, which allows the user to generate a 3D scene, from a description in a natural language, like "The bird is in the bird cage. The bird cage is on the chair". The text is initially tagged and parsed using a part-of speech-tagger and a statistical parser. The output of this process is a parse tree that represents the structure of the sentence.

Next, a depictor (a low-level graphical specification) is assigned to each semantic element. Those depictors are modified to match the poses and actions depicted in the text, by the means of inverse kinematics. Next, the implicit and conflicting constrains of depictors are solved. Each assigned depictor is then applied, while maintaining constraints, to incrementally build up the scene, the background environment, ground plane, and lights are added, and .the camera is positioned. Then, the scene is rendered [COYNE01].

If the case that the text includes any abstraction or description that does not contain physics proprieties and relations, then some techniques are used, such as: textualization, emblematization, characterization, literalizacion or personification.

This system already makes the text-to-scenes conversion, using statistical methods and constrains solvers, and also makes use of different techniques to allow certain expressions to be depicted. However, it only presents static scenes. For the means or our project, we need to establish a dynamical

environment, populated by both avatars and objects, which can perform both autonomous and user based actions. Also, the depictors are taken from a database, whereas we need to access ontology.

2.9.2. DEM²ONS: A High Level Declarative Modeler for 3D Graphics Applications.

DEM²ONS has been designed by Ghassan Kwaiter, Véronique Gaildrat, and René Caubet to offer the user possibilities to easily build 3D scenes in a natural way and high level of abstraction. It is composed of two main parts: Multimodal interface and 3D scene modeler. [KWAITER02]

The DEM²ONS multimodal interface allows the user to communicate with the system, using simultaneously multiple combined ways provided by several Input Device modules (dataglove, speech recognition system, spaceball, mouse...). Syntactic analysis and Dedicated Interfaces modules analyze and control the low-level events in order to convert them into normalized events.

For the 3d scene modeler, DEM²ONS uses ORANOS, a constrain solver designed with several features that let them expand the range of declarative modeling applications, such as generality, avoid breaking, and dynamic constrain solving.

Then, the objects are modeled and rendered by the Inventor Toolkit, which provides the Graphical User Interface, as well as the menus and widgets.

This system already allows the user to interact with the objects in the scenes, as well as resolving any constrain problem, but only accepts static objects, and there is no support for dynamic objects, our avatars, an there is no indication of any interaction whit any knowledge base, two of the main aspects for our project.

2.8.3. Multiformes: A Declarative Modeler as a 3d Scene Sketching Tool.

William Ruchaud and Dimitri Plemenos present us Multiformes, a general-purpose declarative modeler, especially developed for 3D scene sketching. As with every declarative modeler, the work on a scene with MultiFormes is handled essentially through its description (the way the designer

introduces all the characteristics of the geometric elements of a scene and the relationships between them) [RUCHAUD02].

The most important characteristic of MultiFormes is its ability to automatically explore all the possible variations of a scene. Unlike most existing sketching systems, MultiFormes does not enforce only one interpretation of each imprecise property. Starting from the same description, the designer may obtain several variations of the same scene. This may lead him to finally choose a variation of a scene he may have not thought by himself. This exploration process relies on a constraint solver.

MultiFormes is a declarative modeler. The description of a scene includes of two sets: the set of the geometric objects present in that scene and the set of the relationships existing between the geometric objects. In order to allow progressive scene refinement, MultiFormes uses a hierarchical approach of scene modeling. Following this approach, a scene can be incrementally described at different levels of detail. Thanks to its geometric constraint solver, MultiFormes is able to explore several variations of a sketch satisfying the same description.

The geometric constraint solver of MultiFormes is the heart of the system. Inspiration has been essentially taken in Constraint Satisfaction Problem (CSP) resolving techniques,

This system takes an incremental approaching for the declarative modeling. It uses the CPS for constrain solving, and a hierarchically decomposed scene creation. This is not useful for our project, since we are constructing the whole scene with pre-defined objects, so the user doesn't have to explicitly describe the each of the objects.

After reviewing some works, we found that none of them where designed to created fully interactive, self-evolutionary environments. While the main focus of this works is the creation of designing tools, we aim to create a tools that not only let the user design a scene, but that can interact with a bigger architecture in the generation of virtual environments, where the entities that dwell on such environment can act and evolve accordingly. Additionally, the integration of an ontology to the designing of the tools expands the possibilities.

2.9. What is and why to use an Ontology.

One of our main problems is the semantic representation for the entities inside the scene. How can you describe a human? Or how you explain what "run" is? How can be linked "human" with "run"? Since all this concepts need to be stored, we need some kind of database. But such database must have the capabilities to establish relationships between terms, and the concept of these bonds. So we decided to turn to the ontology concept.

An ontology is an explicit specification of a conceptualization. A conceptualization is an abstract, simplified view of the world that we wish to represent for some purpose. For knowledge-based systems, what "exists" is exactly that which can be represented. When the knowledge of a domain is represented in a declarative formalism, the set of objects that can be represented is called the universe of discourse. This set of objects, and the describable relationships among them, are reflected in the representational vocabulary with which a knowledge-based program represents knowledge [GRUBER93].

Basic ontology of the real world is an account of the basic types of things that exist in the world, classified according to their mode of existence. By a mode of existence, we mean the way in which something has come into reality and the manner in which it currently exists [BREY03].

Knowledge-based systems and services are expensive to build, test, and maintain. A software engineering methodology based on formal specifications of shared resources, reusable components, and standard services is needed. We believe that specifications of shared vocabulary can play an important role in such a methodology.

Several technical problems stand in the way of shared, reusable knowledge-based software. Like conventional applications, knowledge-based systems are based on heterogeneous hardware platforms, programming languages, and network protocols. However, knowledge-based systems pose special requirements for interoperability. Such systems operate on and communicate using statements in a formal knowledge representation. They ask queries and give answers. They take "background knowledge" as an input. And as agents in a distributed AI environment, they negotiate and exchange knowledge. For such knowledge-level communication, we need conventions at three levels:

representation language format, agent communication protocol, and specification of the content of shared knowledge [GRUBER93].

Since our project is dealing with the recreation of a scenario using only a natural language description, is necessary to have the means to make a rightful analysis of the sentences, as well as maintain coherency in the scene. An example of this will be a sentence like "The whale is in the living room" By logic, a whale ca not be in a living room, is too big and is a sea mammal. This can be derived from the ontology and the sentence will be marked as invalid. But, a sentence like "The toy whale is in the living room" will be accepted, since a toy shape like a whale effectively can be in a living room. Again, this all can be derived from the ontology.

The ontology will be given the semantic sense to the natural language parser, since it can reason from the ontology. In the "whale" example, the ontology will reason that a whale is a sea animal, a mammal, and that it is several meters long and really heavy to be in a living room. But a toy can be in a living room, only that it can be a toy shaped like a whale. In this case, the sentence is correct.

Thus, we will be using the ontology the exploit knowledge and assure the coherency of the sentences in the descriptions used for recreating a scene in the GeDA-3D virtual environment.

2.9.1. Ontology Creation Tools.

There are several options for creating an ontology, as well as different standards and languages. Here we explore some of them.

Ontolingua [FARQUHAR96]. Developed at KSL of Stanford University. The system consists of a server and a representation language. The server provides a repository of ontologies, allowing ontology creation and modification. The ontologies in the repository can be merged or included in a new ontology. The interaction with the server is achieved by using a standard web browser to connect to the server. This server was designed to allow ontology developing cooperation, easy creation of a new ontology by the possibility to include (parts of) existing ontologies from a repository and the possibility to include primitives from a frame-ontology. Ontologies stored at the server can also be converted to a different format for use in other applications. This allows the users to use the Ontolingua server to

create an ontology, export this ontology and then for instance use it for a CLIPS-based application. It is also possible to import ontology-definitions from a number of languages into the Ontolingua language. The Ontolingua server can also be accessed by other programs that know how to use the ontologies stored in the Ontolingua language [GRUBER93].

WebOnto [DOMINGUE98]. Is, as the name suggests, fully accessible via the Internet. Was developed by the Knowledge Media Institute of the Open University and designed to support the collaborative browsing, creation and editing of ontologies. In particular, WebOnto was designed to provide a direct manipulation interface displaying ontological expressions using a rich medium. WebOnto was aimed to be easy-to-use, yet have facilities for scaling up to large ontologies. Finally, WebOnto was designed to complement the ontology discussion tool Tadzebao. WebOnto is a mainly graphically orientated tool for constructing ontologies. The language used to model the ontologies in WebOnto is OCML. OCML, which stands for Operational Conceptual Modeling Language, was originally developed in the context of the VITAL project to provide operational modeling capabilities for the VITAL workbench [MOTTA97]. The tool has a number of useful features, like saving diagrams of structures, viewing the relations, classes, rules etc. separately. Other features include for example, working cooperatively on ontologies, by the use of drawing, and using the broadcast and receive functions.

Protégé [FERGERSON98]. A multi-platform program. It is meant for building ontologies of domain models, and has been designed by Stanford's Medical Informatics Section. Protégé has been developed to assist software developers in creating and maintaining explicit domain models, and in incorporating those models directly into program code. The Protégé methodology allows system builders to construct software systems from modular components, including reusable frameworks for assembling domain models and reusable domain-independent problem-solving methods that implement procedural strategies for solving tasks [ERIKSSON95]. The program consists of three major parts, first there is the 'Ontology-editor' which let you make your own ontology about a domain by just expanding a hierarchical structure, and including abstract or concrete classes and slots. Based on the ontology built, Protégé is able to generate a knowledge acquisition tool for entering the instances of the ontology. The KA-tool can be fine-tuned to the needs of a user by using the Layout-editor. The last part of the program is the 'Layout-Interpreter' which reads the output of the 'Layout-Editor' and shows the user an input-screen with a few buttons. These buttons can be used to make the instances for the classes and sub-classes. The whole tool is graphical which is very usable for a naive user.

2.10. Conclusions.

The declarative modeling is an investigation field that can lead to useful, simply to learn, user-oriented tools. Several of the works in the field lead to static models [COYNE01], semi-interactive environments [KWAITER97], or used incremental modeling [RUCHAUD02].

None of this works deal with totally interactive, agent based environments, or the exploitation of knowledge bases, being the first an integral part of the GeDA-3D project, and the later, an approach that is posed as a solution for the natural language parsing and constrain problem solving. Since we're trying to establish a natural-like language for the modeling stage, none of the previous works helps us.

This leads us to generate a language that is both natural and restricted. The restriction is necessary, because we're trying to avoid any conflicting structure. We then can assure the module will effectible make the conversion, with no considerable computational cost.

Chapter 3. Proposed Solution

Abstract

In this chapter we present the approach we take to solve the challenges we found as we advance in the research, the solutions we proposed, the justification for using such solutions, and the process to generate applications compliant with them.

3.1. Introduction.

In this chapter we present our own defined language: VEDEL, or Virtual Environment Definition Language, which is a high-level language created to define virtual scene and environments. We also specify VEDEL using EBNF, and we explain the framework created to establish to flow from the high-level description to the output through the GeDA-3D platform. We also specify the ontology use and organization, as well as the specification of the parser and the interaction it will be presenting with the ontology.

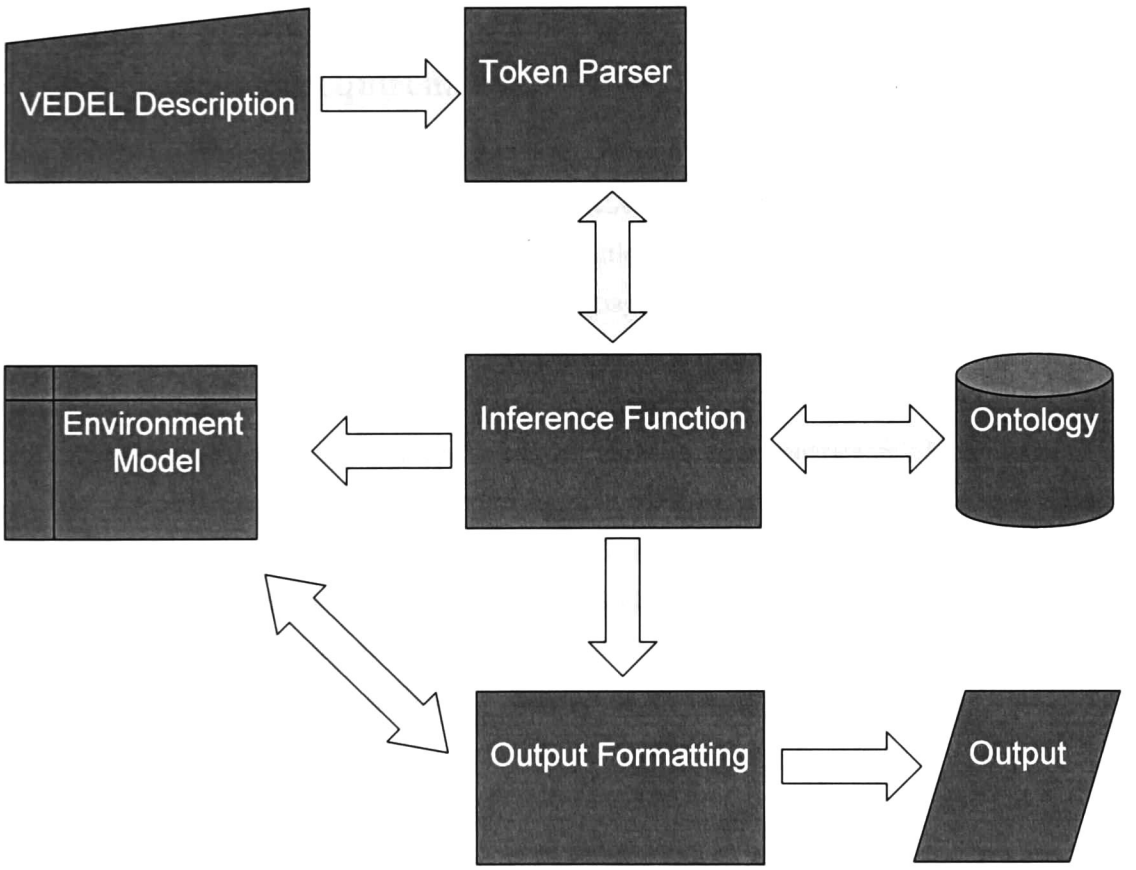


Figure 3. Our proposed architecture.

VEDEL was developed taking into account the need to establish an ordered method to describe a scene. When a person takes the task of describing an scene, he or she may tend to concentrate on something that attract their attention, like some animal, a construction or a person on the scene. Such

predisposition to concentrate on one element can generate several different descriptions from the same setting.

Analyzing the human natural speech is a complex task, both in computational resources and time. Since our aim is to create virtual scenarios from natural-like descriptions, the cost must be reduced, so we can provide real-time parsing and output generation. To carry out this objective, we define the three main elements in a scenario, environment, actors and objects, and dispose of redundant elements, such as articles, organize the way the description must be conducted, and simplify the definition of actions, positions and properties.

3.2. Language requirements.

In the State of the Art section, we exposed some types of applications our project can embrace. Since any of the applications require specific syntactic and semantic elements, there is the need to specify which elements will be supported by the language, considering the ambiguities inherent to the natural language (such as double meanings, grammatical constructions or semantic ambiguities).

The language needs to be expressive enough to let the user generate any kind of virtual environment, but also it has to be a context free grammar, in order to avoid ambiguities. Among such ambiguities, the harder to solve are those related with the composition of the sentence. A sentence like "a dog is under the table" has the same meaning than "under the table, there is a dog", but each has a different method to solve. We need to define a language that doesn't have this kind of context-based problems, so the user can only uses one form of construction, in this case, the easier for us.

The addition of the ontology to the syntactic and semantic parser will ease some of the work, since the individual elements that need to be treated for meaning will be found and categorized by the ontology itself, allowing expanding the capacity of the parser by simply expanding the contents in the ontology. However, some of the language elements need to be in the definition of the language, such as prepositions, articles or adverbs, due to the connecting nature those have. Since many nouns and verbs have context-based meaning, we will let those to be solved by using the ontology.

3.3. Scope.

Since natural language parsing is a complicated task if we don't make restrictions on what can be expressed and how it can be constructed, is evident the need for delimiting the expressiveness and constructing elements.

The natural language has an ambiguous, non standard constructing form. The same idea can be expressed in several different ways. Some of the parsers that have taken the full natural language parsing use statistical methods, in which case the computational cost can be high, both resource and time wise.

The scope of the language will be, then, to allow the user to express the idea that it has for the scene, delimiting the forms it be can done, but not what can be expressed.

3.4. Restrictions.

Since we're delimiting the scope and reach the language can have, then we have to set the restrictions necessary for defining an easy to understand and read, low computational cost parsing language. Thus, the language must allow one way to construct sentences, as well as one form to structure the whole description.

The natural language has an ambiguous, non standard constructing form. The same idea can be expressed in several different ways. Some of the parsers that have taken the full natural language parsing use statistical methods, in which case the computational cost can be high, both resource and time wise.

We're confident that any user can express anything even with a restricted language effortless, even if the user needs to learn some new syntactic and semantic rules. This also allows us to design a fast and reliable parser, since is a lot easier that a person learn to express something in slightly different way, than teach the computer to understand the natural language humans use everyday. Many of the parser

used statistical approaches or stochastical approaches, what can make the parsing slow, or not totally reliable. Others make use of databanks or knowledge bases, but still the parsing can take long time since some of the constructions possible with a human natural language can be difficult to understand, due to the language is naturally context-based.

3.5. Introduction to VEDEL.

VEDEL is a high-level language oriented towards the easy creation of virtual scenarios, which then can evolve by their own, based on the particular characteristics of each entity within the environment. A description written in VEDEL is divided in three main sections, with each section been composed of sentences. A sentence is a comma separated list of the properties we want to set to an avatar, and begins with the entity name and optionally an identifier for that specific avatar. The later part of the sentence is composed of the properties that particular avatar we want it to show, such as color, size, position. The list must present in the first place the property name, and then the value we want to assign to it. Additionally, there are certain properties that have no values to assign, so the single reference to those is valid. For actions or emotions the actor can perform or be subjected to, no values are required. Positioning of avatars can be executed in two different ways: by expressing a relative point or by expressing distance. In both cases, the first element must be the spatial reference (such as north, south, left, above, far) or the distance expressed in numeric and later the reference point (a valid element within the environment). The preset distant unit is meters, and any distance can be expressed using real digits. Also, the user can simply express a spatial point (such as center, north, south, northeast), and the element will be positioned in such point.

There are three sections in VEDEL: Environment Section, Actor Section and Object Section. These sections are explained later in deep.

We must have in mind that VEDEL is a declarative language. Thus, in a declarative language the instructions indicate how things are going to be made, instead of what is going to be made [CHRONAKI90]. The language also has a scripting part, that comes from the concept that a script language doesn't care about the components they are calling, they assume they already exist [OUSTERHOUT98], in our case, a rendering component and the agent core, both from the GeDA-3D

system. Since we will be indicating how the scenes must be created, it is important that the parser have something that indicates which syntactic and semantic features a token has.

Next, we present a description written in VEDEL:

```
[ENV]
Desert, day, temperature 39, no wind.
[/ENV]

[ACTOR]
Camel Albert, big, center
Spider Spider1, front Albert, small, grey.
Snake, left Spider1, green, angry.
[/ACTOR]

[OBJECT]
Rock R1, behind Snake1.
Cactus C01, left R1.
[/OBJECT]
```

Figure 4. An example of the VEDEL language

3.5.2 Basic Structure

Since VEDEL is a high-level language, is difficult to establish a conventional type to classify it. The notional conventions that will be use for presenting the syntax are:

<*Pattern* >	optional, this section can be excluded on the description.
(*Pattern* ...)	iteration, the contents of this section can be repeated, with no finite restriction.
`Wooden Table`	terminal syntax or example. A terminal value corresponds to an entry to the ontology, a value for a property, or an object identifier.

The basic structure is presented next:

[ENV]
Environment keyword <environment settings>
[/ENV]

[ACTOR]
Actor1 Class <Actor1 Identifier> <Actor1 properties>
(<Actor2 Class <Actor2 Identifier> <actor2 properties>>...)
[/ACTOR]

[OBJECT]
Object1 Class <Object1 Identifier> <Object1 properties>
(<Object2 Class <Object2 Identifier> <Object2 properties>>...)
[/OBJECT]

In the next sections we present the detailed description of these description sections and subsections.

3.5.3. The ENV section.

In this section, the user can describe the environment the scene will be placed. It is composed by two subsections, an environment keyword, and the environment settings section.

For the environment keyword, the user would express with one word the general setting for the scene, such as "room", "desert", "sea", "forest", "school" or any other noun that corresponds to a place.

The environment setting section is a comma separated list, which can include any number of elements, describing specific elements of the environment, such as day/night setting, temperature, weather or specific setting such as window for a room, wall materials or density for a forest. The specific description for those specialized setting can be furthered expanded on the objects sections.

3.5.4. The ACTOR section.

This section is composed by the descriptions of every avatar (actor) the scene will include. The basic layout for the section is a list separated by a carriage return.

On each element of the list, the fist entry should correspond to the general description for the avatar, in other words, a single sentence depicting only the generals for the avatar, such as "man", "girl", or "dog". Next, the user can express specific characteristics for that actor, such as size, clothing, hair color, etc. Finally, the user can dictate what actions will be performing that actor at the beginning of the scene, such as walking, running, swimming, indicating specifics for that action. No more than one action is allowed for each actor.

3.5.5. The OBJECT section.

The final section corresponds to the description of every object in the environment. These objects can be environment related (such as the windows of a room, or the trees of a forest) or independent (such as boxes, chair, cars or buildings)..

This section is, like the ACTOR section, a carriage return separeted list of sentences, in which each element of the list is sub-divided in two: the general description for the object, with a single, simple sentence (such as "box", "table" or "river") and the specifics for each object, including which actions can be performed over them or are subject to.

3.6. VEDEL formal definition.

In this section, we present a formal definition of the VEDEL language using the EBNF (Extended Backus Normal Form), being our language context-free [ALANEN04].

Description Structure.

Description := <environment> <actor> <object>

Section Structure

<environment> := "**[ENV]**" <esentence>"**[/ENV]**"

<actor> := "**[ACTOR]**" <asentence> *"**[/ACTOR]**"

<object> := "**[OBJECT]**" <osentence> * "**[/OBJECT]**"

Sentence Structure

<esentence> := <cenvironment> (<separtor> <eproperty>) * <eos>

<asentence> := <cavatar> (<separator> <aproperty>) * <eos>

<osentece> := <cobject> (<separador> <oproperty>) * <eos>

Class Structure

<cenvironment> := <environmentid>

<cavatar> := <avatarid> (<identifier>)

<cobject> := <objectid> (<identifier>)

Property Structure

<eproperty> := <characteristic>

<aproperty> := <characteristic> | <action> |<position> | <emotion>

<oproperty> := <characteristic> | <position>

<characteristic> := <propid> <value>

<action> := <actionid> (<modifier>)

<position> := <posid> <identifier> | <number> <identifier>

<emotion> := <emoid> (<modifier>)

Vocabulary

<environmentid> := <word>

<avatarid> := <word>

<objectid> := <word>

<propid> := <word>

<actionid> := <word>

<posid> := <word>

<emoid> := <word>

<value> := <word> | <number>

<modifier> := <word>

<identifier> := [letter | digit] +

<word> := letter +

<number> := digit + (“.” digit +)

Alphabet

letter := [“A” – “Z”] | [“a” – “z”]

digit := [“0” – “9”]

<separator> := “,”

<eol> := “.”

3.6.1. Identifiers.

While doing the parsing of the description, we can find several different identifiers. The parser has then to deal with them using the ontology, which will indicate the next step on the analysis process. The identifiers the parser can find are:

a) <environmentid> : The environment is defined by a single word, a noun that represent a real, or fantastic location for the scene.

b) <avatarid> : Avatars are virtual entities. Those entities can perform actions, react to the environment or other avatars and have emotions.

c) <objectid> : An object is a virtual entity that can't perform any action. However, the environment and other entities can change their states.

d) <propid> : A propriety is a characteristic that an avatar holds. This can be a visual characteristic, or an internal characteristic.

e) <actionid> : Actions are performed only by the avatars, and the avatar can only perform them if that actions is allowed on the environment or the context previously defined.
f) <posid> : The position an avatar or an object can be expressed in a relative way, or as a explicit statement.
g) <emoid> : Emotions, which only avatars can represent. Such states can affect the way an avatar performs its actions.

Other identifiers are context-related, and bear only the values the previous identifier will present. Such identifiers are:

a) <identifier> : An alphanumeric name given to a particular avatar. Helps in the process of creating the description by allowing the user to refer to another avatar by a proper name. Those identifiers must be unique for each avatar.
b) <value> : Alphabetic or numeric values that a property can accept. Those values are also confronted with the ontology for validation.
c) <modifier> : A word that express an non-specific amount. Those words hold a percent or numeric value in the ontology, which indicates the level of intensity an action is performed or an emotion is exhibit.

3.7. Semantic Association.

Due to the enormous amount of data that a word can provide for the construction of an environment, is necessary to allocate as much as possible on an organized, easy-to-use source. For such source, we have selected the ontology.

Since the ontology can have in an entry more than just a semantic key, we can also assign other attributes, such as associations with other entries or specialized data (genders, physical or emotional characteristics, capabilities, among others), which can allow us to expand the capacity of the parser and ease the processing task by giving valuable information such as legal values, data types, valid structure organization or simply to aid in the generation of the output. Also, the kernel of the GeDA-3D system requires some specific details for the agents, such as innate characteristics for an avatar (behaviors or actions the avatar or object can be subjected to), details that can be provided by the ontology.

Each ontology entry will be assigned to a specific class, which helps during the parsing process by adding a control mechanism and validation data (such as verifying that an object is really an object, or that an action is such and not a law). Since not only individual words would be stored in the ontology, but also laws, collisions and other application-specific data, the organization of the ontology is critical. Even when a word could have several meanings, the class under that word is localized helps the parsing process to determine which others elements should appear next, or what element must have derived from. The entry should have a mirror entry on the render database, since that module will be presenting the visual output. If there is no entry on the parser, the specified element can't be represented.

The ontology is basically organized in four classes: Environment, Actors, Objects and Keywords. The Environment class holds all of the possible scenarios the render can present. Actors and Objects correspond each to the possible entities that can be presented. And Keywords keeps all the possible modifiers and values that need a special entry.

3.7.1. Semantic Validation.

Once the parser has located any identifier within the description, the validation of such identifier must be located within the ontology. Such procedure can resumed as the following steps:

a) Find the entry for such identifier. In the case such identifier is an avatar's proper name; we look for it in the avatar table to avoid duplicity. If the identifier is a value, we search for the property within the avatar and validate such value. If the identifier is recognized as an avatar class or property, we search through the ontology for the corresponding ontology class or property.
b) Once the identifier class is found in the ontology, we verify the class, so if it corresponds to the section we are analyzing, or the property is valid for the avatar we are validating, we can continue, otherwise, an error condition is launched.
c) For the avatar class, if the validation is successful, the properties for that avatar are extracted, and then a new entry is added to the avatar list, with the property values set to the default stored in the ontology.

3.8. VEDEL Framework.

The objective of this section is to present the formal description of the analysis process that the VEDEL parser performs over the description. The output is adapted for the current GeDA-3D prototype, and consists of three sections: An AVE-formatted description of the scene, the context description, and the relation of agents that must be created to allow the evolution of the scene.

Let $E = \{S, A, O, l\}$ be the non-empty set that creates the virtual environment, where S corresponds to the environment, A is the actors set, O holds the object set and l the set of laws that rule over the environment. Both A and O can be empty, but not l.
We define $S = \{SE, w\}$, where SE is the non-empty set composed of every element that composes the environment, thus $SE = \{e \mid e \subseteq \{\{\text{ontology classes}\} \cup \{\text{ontology properties}\}\}$, w is the set of laws that rule only over the environment, as defined on the ontology.

Let $e = \{m, EP, v\}$ be the set that defines the environment, then e must satisfy the following:

1) $m \in \text{EC} \mid \text{EC} = \{\text{Environment Classes}\} \Rightarrow \text{EC} \subseteq \{\text{Ontology Classes}\}$.
2) $EP = \{p_0 \ldots p_n \mid p_i \in \{\text{Ontology Properties}\}$ and $p_i \in \{\text{properties defined for } m\}\}$.
3) $v = \{ EP \times val \mid val = \{\text{any valid data value}\}\}$

For w, we define the set of rules that governs over the environment as:
$w = \{lw_o \ldots lw_n \mid lw \in \{\text{Law Classes}\} \Rightarrow \{\text{Law Classes}\} \subseteq \{\text{Ontology Classes}\}\}$

Let A be the set of actors for the defined scene. Then $\text{A} = \{AT_i \mid \text{i} = 0 \ldots n\}$, where $AT = \{a.\ AP, AC, AE, P, va, vc, ve\}$ must satisfy:

1) $a \in \text{AC} \mid \text{AC} = \{\text{Actor Classes}\} \Rightarrow \text{AC} \subseteq \{\text{Ontology Classes}\}$.
2) $AP = \{p_0 \ldots p_n \mid p_i \in \{\text{Ontology Properties}\}$ and $p_i \in \{\text{properties defined for } a\}\}$.
3) $AC = \{\text{w}_0 \ldots w_n \mid \text{w}_i \in \{\text{Action Classes}\} \wedge w_i \in \{\text{actions assigned to } a\}\}$.
4) $AE = \{\text{s}_0 \ldots s_n \mid \text{s}_i \in \{\text{Emotions Classes}\} \wedge s_i \in \{\text{emotions assigned to } a\}\}$.
5) $P = \{xa, ya, za \mid na = \{\text{Z}\}\}$
6) $va = \{ AP \times val \mid val = \{\text{any valid data value}\}\}$

7) $vc = \{ AC \times val \mid val = \{\text{any valid data value}\}\}$

8) $ve = \{ AE \times val \mid val = \{\text{any valid data value}\}\}$

For P, defined above, each na value can be assigned in three different ways:

a) $P = \text{PA}(A_1) + \text{PS}(r)$, where

$\text{PA}(Ai) = \{xa, ya, za\}$ for a set Ai, where $Ai \neq A$.

$\text{PS}(r) = \{x, y, z \mid x, y, z = \{\text{Z}\}\}$, where each element of the set receives its value from r, which satisfies:

- $r \in \{\text{Keyword classes}\} \Rightarrow \{\text{Keyword classes}\} \in \{\text{Ontology classes}\}$

  $\{x, y, z\} \in r$.

b) $P = \text{PA}(A_1) + \text{PS}(z)$, where

$\text{PA}(Ai) = \{xa, ya, za\}$ for a set Ai, where $Ai \neq A$.

$\text{PS}(z) = \{x, y, z \mid x, y, z = \{\text{Z}\}\}$, where each element of the set receives its value from z, which satisfies:

- $z \in \{\text{Keyword classes}\} \Rightarrow \{\text{Keyword classes}\} \in \{\text{Ontology classes}\}$

  $\{x, y, z\} \in r \Rightarrow \{x, y, z\} * \{u_1, u_2, u_3\}$, where u_i is a valid numeric value expressed by the user.

c) $P = \text{PS}(z)$, where

$\text{PS}(z) = \{x, y, z \mid x, y, z = \{\text{Z}\}\}$, where each element of the set receives its value from z, which satisfies:

- $z \in \{\text{Keyword classes}\} \Rightarrow \{\text{Keyword classes}\} \in \{\text{Ontology classes}\}$

  $\{x, y, z\} \in r$

For the set O, set of objects defined scene in the scene, we have $O = \{OT_i \mid i = 0 \ldots n\}$, where $OT = \{o, OP, P, vo\}$ must satisfy:

1) $o \in \text{OC} \mid \text{OC} = \{\text{Object Classes}\} \Rightarrow \text{OC} \subseteq \{\text{Ontology Classes}\}$.

2) $OP = \{p_0 \ldots p_n \mid p_i \in \{\text{Ontology Properties}\} \text{ and } p_i \in \{\text{properties defined for } o\}\}$.

3) $P = \{xa, ya, za \mid na = \{\text{Z}\}\}$

4) $vo = \{ OP \times val \mid val = \{\text{any valid data value}\}\}$

For P, defined above, each na value can be assigned in three different ways:

a) P = PA(A_1) +PS(r), where

PA(Ai) = {xa, ya, za} for a set Ai, where $Ai \neq A$.

PS(r) = {x, y, z | x, y, z = {Z}}, where each element of the set receives its value from r, which satisfies:

$r \in$ {Keyword classes} $\Rightarrow$ {Keyword classes} $\in$ {Ontology classes}

$\{x, y, z\} \in r$.

b) P = PA(A_1) +PS(z), where

PA(Ai) = {xa, ya, za} for a set Ai, where $Ai \neq A$.

PS(z) = {x, y, z | x, y, z = {Z}}, where each element of the set receives its value from z, which satisfies:

- z $\in$ {Keyword classes} $\Rightarrow$ {Keyword classes} $\in$ {Ontology classes}
- $\{x, y, z\} \in r \wedge \{x, y, z\} * \{u_1, u_2, u_3\}$, where u_i is a valid numeric value expressed by the user.

c) P = PS(z), where

PS(z) = {x, y, z | x, y, z = {Z}}, where each element of the set receives its value from z, which satisfies:

z $\in$ {Keyword classes} $\Rightarrow$ {Keyword classes} $\in$ {Ontology classes}

$\{x, y, z\} \in r$

Finally, the set $l = \{l_0 \ldots l_n\}$, where $l_n \in$ {Law Classes} $\Rightarrow$ {Law Classes} $\in$ {Ontology Classes}.

3.9. Ontology Organization.

3.9.1 Why choosing an ontology?

There are some options to define, index and store data. Our first need was the quick access to that data, but we also needed a way to organize that data, recreating any relationship two entities could bear. Also, we need to express a number of properties, from physical to physic. Also, we need the ability to relate two entities expressing what kind, and the name, of relation link those entities.

Another need was the capability of link different data banks, where some of the data could be on the same machine, or in different machines, either on a LAN, WAN o even over the WWW. This also comes with the need to adopt a standard for designing and storing the data, as well as the distribution of the data.

All this requirements could be covered by using a database, but the execution of some would be too complicated and make the data organization and mining complex and difficult to maintain. For that reason, we decided to use a knowledge base.

A knowledge base is committed to some conceptualization, explicitly or implicitly. A body of formally represented knowledge is based on a conceptualization: the objects, concepts, and other entities that are assumed to exist in some area of interest and the relationships that hold among them [GENESERETH87]. A conceptualization is an abstract, simplified view of the world that we wish to represent for some purpose.

One of such knowledge base is the ontology, which not only allows the organization of the data in a hierarchical structure, it also allows the expression of relationships between terms, as well as the assignment of specific data values for each term stored in the ontology. Adding to these characteristics the ability to access the data with no platform-dependent needs, and the capacity to integrate new data from another ontology, our choice was evident.

The ontology creation tools choose was Protégé [FERGERSEN02]. It was selected because of its graphical, easy to use interface, which allows a quicker creation of ontology projects, even for the inexperience user, the total compatibility with the OWL standard [DEBORAH04], used to generate W3C compliant ontology which can be accessed through the WWW, the multi-platform execution, since the platform is coded on Java, and the easy-to-import and completely documented API.

3.9.2. Classes, properties and datatypes defined for VEDEL.

To allow an easy organization of the terms and speed up the search for terms and the corresponding validation through the parsing progress, we establish a model for the ontology. Such model comes from the separation we define in the language, plus additional classes that contain laws and control words.

Properties don't require any control structure since many of them are valid for any of the classes, and specific information for each class can be stored individually under the same property. The organization is presented next:

a) *Environment class*: Contains all the environments that will be accepted in the description. Each subclass must be a single word that describes a specific scenario, with cases like "desert", "sea" or even "tropicalForest" permitted, and must contain the allowed actions in the form of restrictions, through the "AllowedAction" property.
b) *Actors class*: All the actors must be placed in this class. The actor must be described as a single word, with the cases identical to the environment naming. The allowed actions and emotions must be specify in the form of restrictions, using the "CanPerform" and "CanExpress" properties.
c) *Object*: Any object that can be represented is parsed through this class. The naming convention for these subclasses is the same as the Environment and Actors classes.
d) *Actions*: The valid actions the under laying architecture can execute. Naming is restricted to single, non-composed words. They also must contain, in the form of restrictions, in which environments such action is allowed, through the "AllowedOn" property.
e) *Emotions*: The emotions the support architecture can handle. Naming can only be accepted as single words.
f) *Keywords*: Pronouns and specific directions.
g) *Laws*: Any law that must be considered inside the environment.

Also, each sub-class must contain an "<subclass_name>_default" individual, where all the property values can be stored.

1. For the environment, properties, properties values and laws must be specify on such individual.
2. Actors' individuals must contain the allowed properties, properties values, and the default action.
3. In the objects' case, the only attributes must be properties, and properties values.
4. Actions, Emotions and Laws only contain the properties they can accept.
5. Keywords only hold the necessary data to modify values from properties, actions, emotions and laws.

The only properties that can be added are datatype properties, and each property added to an entity must also be place on the "attributes" property list. Each property must contain at least two values, the first indicating the data type for that property (1 for strings, 2 for numeric values) and then either the expression "VAL = <value>" or the expressions "DEF = <value>", "MIN = <value>" and "MAX = <value>".

Chapter 4. Obtained Results

Abstract

The present chapter presents the relation of results product of our investigation. We present two different prototypes for the GeDA-3D project, as well as a parser created to allow the user to create a virtual environment, which is presented on the GeDA-3D render, and evolved thought its core.

4.1. Introduction.

In this final chapter we offer a recount of the actions we take to prove useful the use of ontologies in the description phase of declarative modeling. Such actions cover from obtaining information about natural-language parsing methods to understanding the Protégé API for ontology access and data access.

Some work was conducted towards the GeDA-3D project, specifically the resulting output and the data collection for the ontology. This resulted on two prototypes, one aimed to present the property analysis and emotional agents' evolution, and the other to present a 3D environment constructed through a description.

4.2. The VEDEL parser.

In order to prove the efficiency on using an ontology as a medium for analyzing and validating a description written on the VEDEL specification, we construct a parser. Such parser was written on Java, for multiplatform capability, using the Protégé API for access to the ontology. The parser was written and built using the last JDK available, 1.5.0_07-b03.

Basically, our parser is a state machine: each section of our description corresponds to a specific state. By extracting words from the description, which we call Tokens, we can establish the current state and determinate which state comes next.

Basically, the validation for each word comes from a Inference Function: such function was designated to extract information from the ontology. When a new type of avatar is found, all the information that avatar can hold is extracted from the ontology and store on a data structure. By doing this, we can then access and modify specific properties for each individual, accessing the ontology only to acquire new information, like the specific operation or values to place an entity on the center of the scene, or to

validate a specific value, such as a color, considering that perhaps not all color are available for an avatar.

Once all of the description has been parsed and validated, we can then construct any output we desire. This can include specific data encoded in the ontology, which can still be accessed, such as xml code or properties that only affect one type of avatar and cannot be modified (paths to 3D object or sound files).

4.3. The GEDA-3D prototypes.

In order to value the progress obtained during two different stages of the development of the parser and the language definition, we designed outputs for two different versions of the GeDA-3D core prototypes.

The first was called "The battle of the frogs", and was basically a prototype that demonstrates the performance of the emotional agents used to evolve a scene. The prototype presented a 2D graphic field where two opposing "frogs" battle one against the other by throwing "balls" The emotional state of the frogs influences the shooting rate and the aggressiveness in the behavior of the frogs. A scare frog would try to avoid the incoming attacks ratter than attack, and an angry frog would prefer stand against the incoming attack so it can return fire the more often possible.

The ontology took class entries for the "frog" class, as well for the "bullet", being the only two entities on the environment, each containing the necessary data to deliver to the core the necessary context data, such as bullets the frogs could have, the energy they bore at the beginning, and the corresponding laws, actions and environment settings and data. The output then was formatted according to the XML sheet provided for the core.

The result was the generation of different matches, and the possibility to increase the number of possibilities for the frogs and scenarios by adding new data to the ontology. From new environments to new actors, the new additions can be done by simply creating their corresponding class to the ontology,

giving the user the opportunity to modify values such as the default bullet number, the starting emotional state and even change the actions allowed inside any giving scenario.

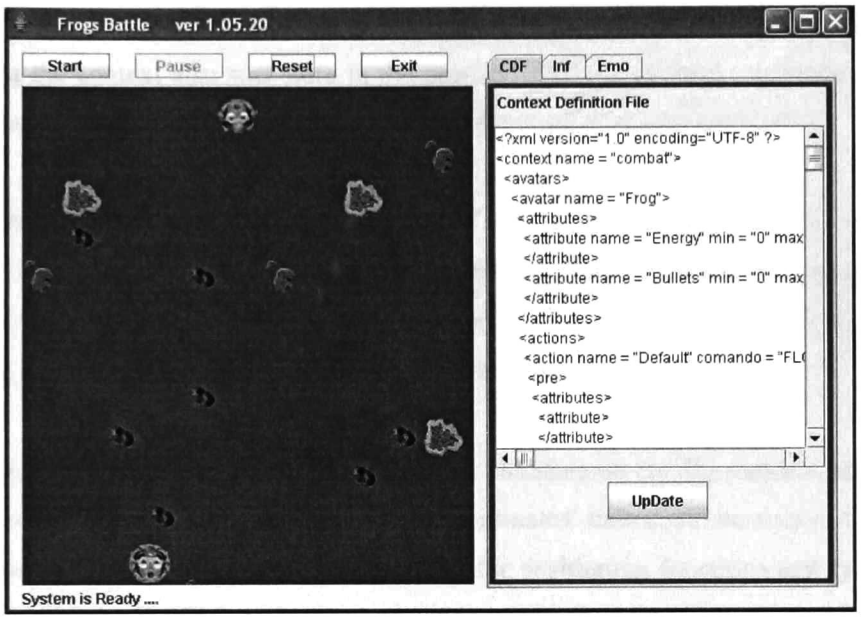


Figure 5. *Battle of the frogs*. Starting Settings.

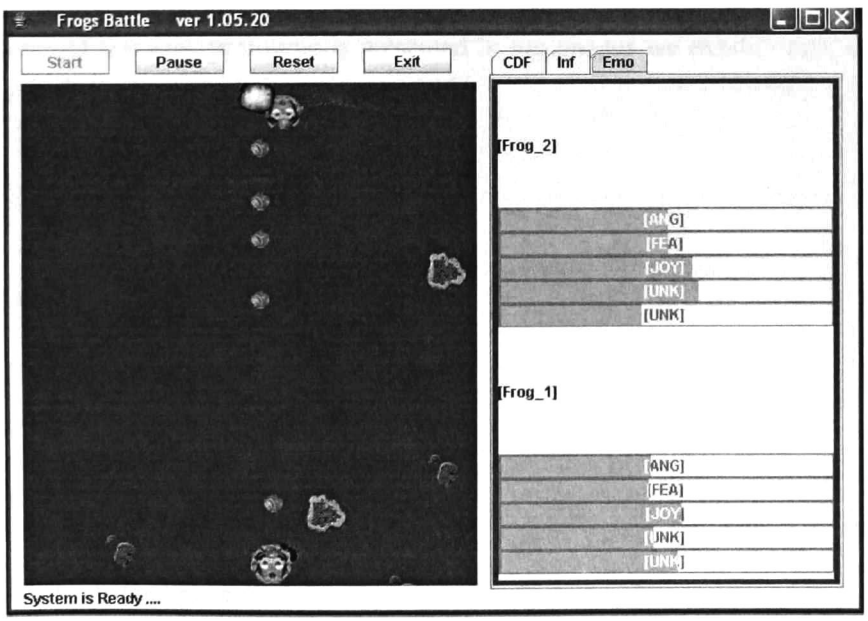


Figure 6. *Battle of the frogs*. Scene Development (with emotions properties displayed).

By using VEDEL, we can express the initial state of the match. The frogs have the "bullets" and "energy" properties, and there are two types. Each type has a different initial emotional state. The ontology not only holds the classes for "frog", "bullets" and "energy", but also contains the valid actions for the "frog" actor" as well as the laws there are subdue to. Finally, the data necessary for describing the context also was store in the ontology. All the process was transparent for the user, as only the basic description creates all the elements for conducting the simulation.

The second prototype uses the render module of the GeDA-3D project, called AVE, to present a 3D scene, on which the core could perform a determined act, such as the predator-pray setting. What we do basically for this second prototype was to improve the parsing process to optimize the code, by removing replicate instructions and making it more readable.

Also, there where added functions to position the elements on the 3D virtual space. This also required adding new properties to the ontology, so the coordinates' data could be stored, as well as new classes for "keywords". Such keywords where selected for positioning functions and are adjectives (such as right, left), adverbs (behind) or cardinal points (north, east). Each of those new additions bear specific indications for placing the entities in the desired place.

The outcome of this new prototype is presented in the images we exhibit next, each accompany with the source description.

Example 1.

Description:

```
[ENV]
Cuarto.
[/ENV]

[ACTOR]
 HombreVerde Antonio, center.
 HombreExpresiones Bruno, left Antonio.
[/ACTOR]
```

Figure 7. Description 1. Front view.

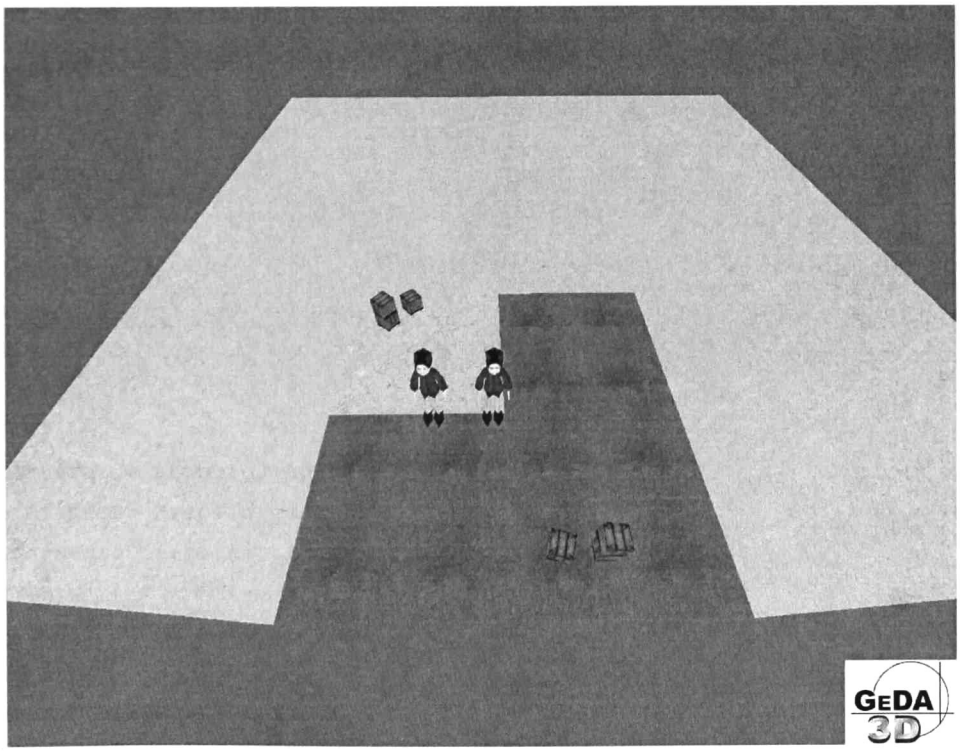


Figure 8. Description 1. Up-down view.

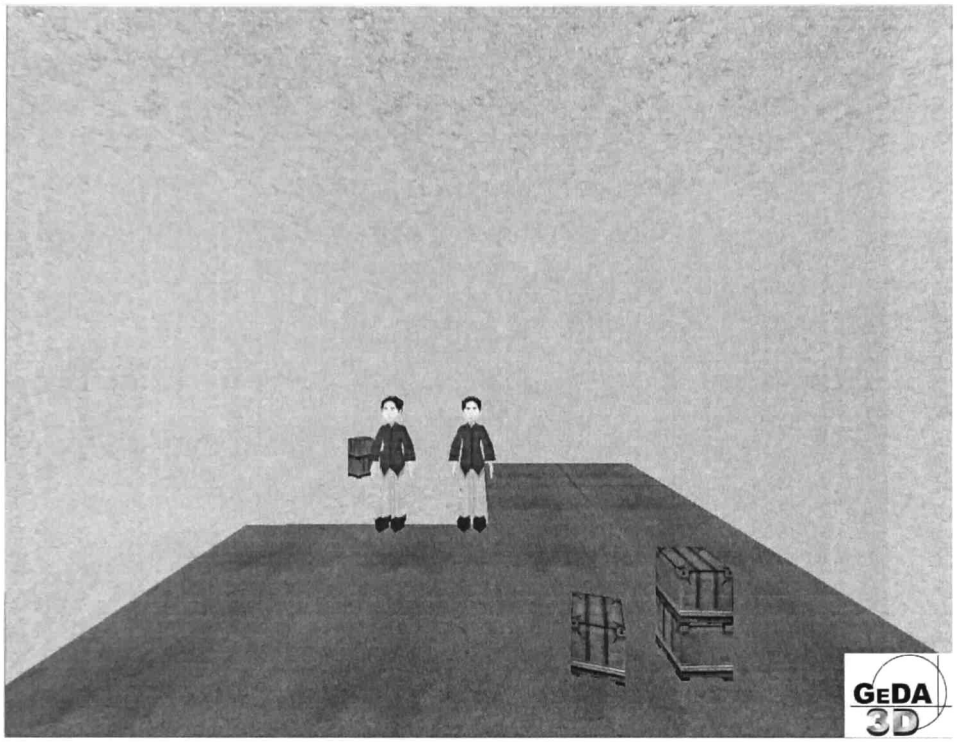


Figure 9. Description 1. Wide view.

Example 2.

Description.

```
[ENV]
cuarto.
[/ENV]

[ACTOR]
  HombreVerde Antonio, north.
  HombreNegro Braulio, south.
  MujerNegro Carolina, front Braulio.
[/ACTOR]

[OBJECT]
  Perro Donas, left Antonio.
  Mariposa Esperaza, behind Donas.
[/OBJECT]
```

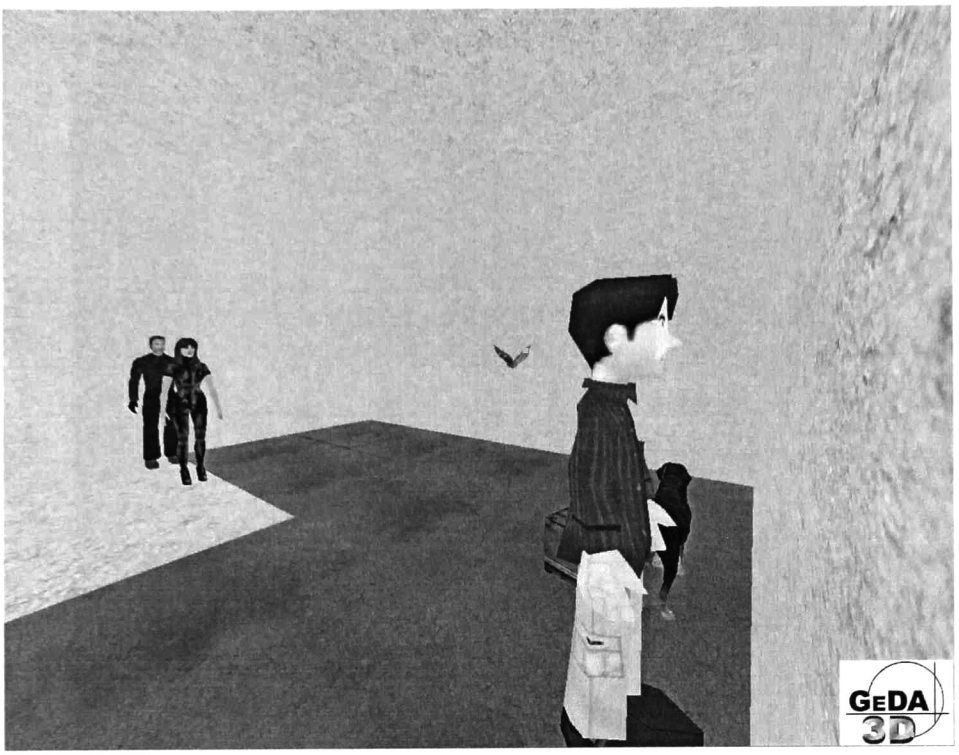


Figure 10. Description 2. Close View.

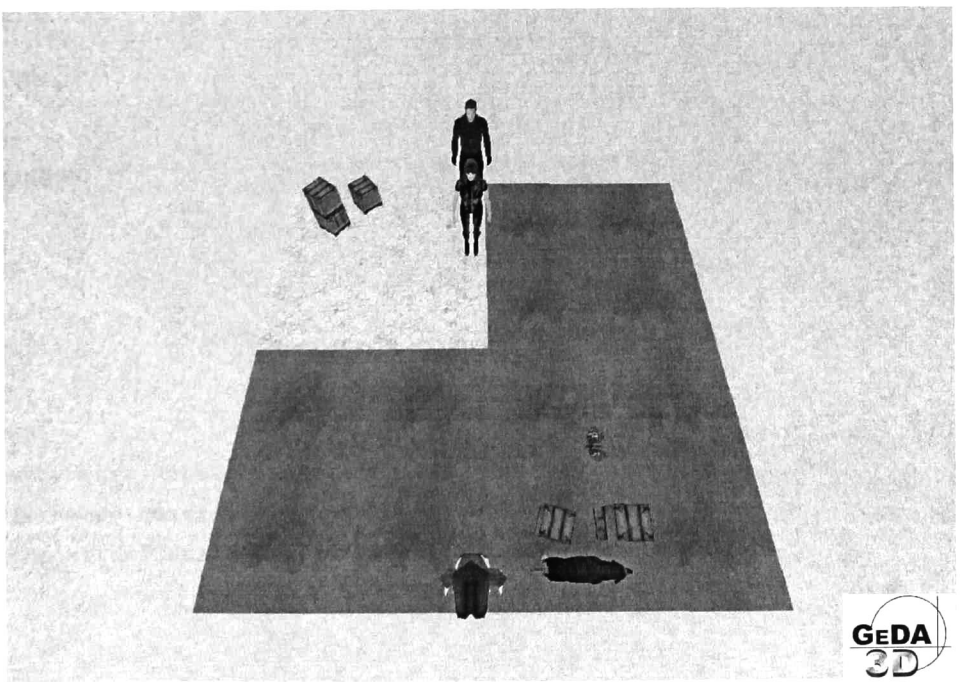


Figure 11. Description 3. Up-Down View.

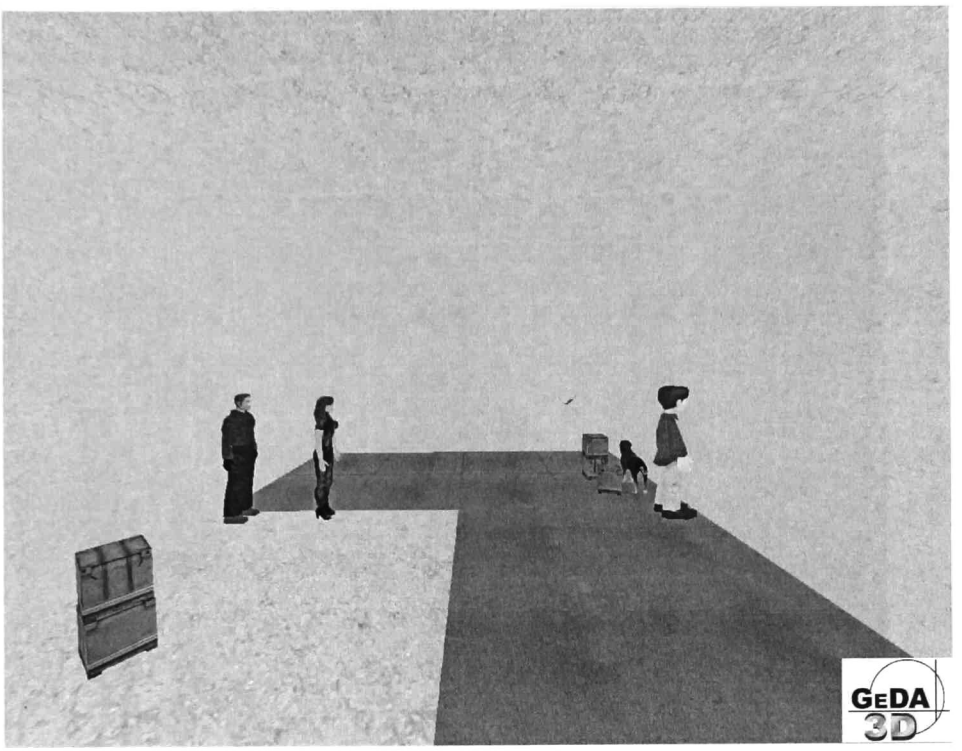


Figure 12. Description 3. Wide View.

Example 3.

Description.

```
[ENV]
  cuarto.
[/ENV]

[ACTOR]
  HombreNegro Alberto, center.
  MujerNegro Beatriz, right Alberto.
  HombreVerde Cecilio, left Alberto.
[/ACTOR]

[OBJECT]
  Chapulin Insecto1, south.
 [/OBJECT]
```

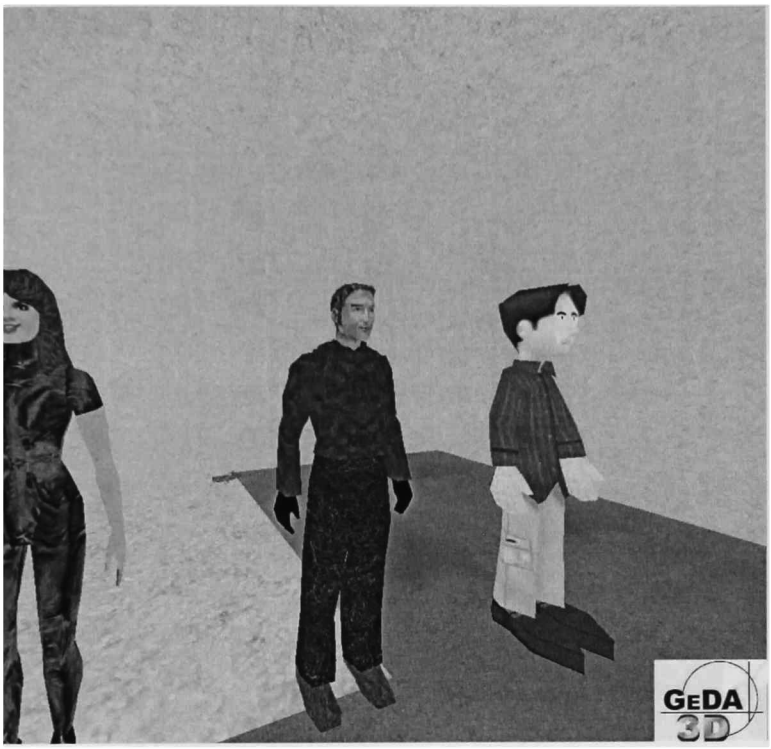


Figure 13. Description 3. Close View.

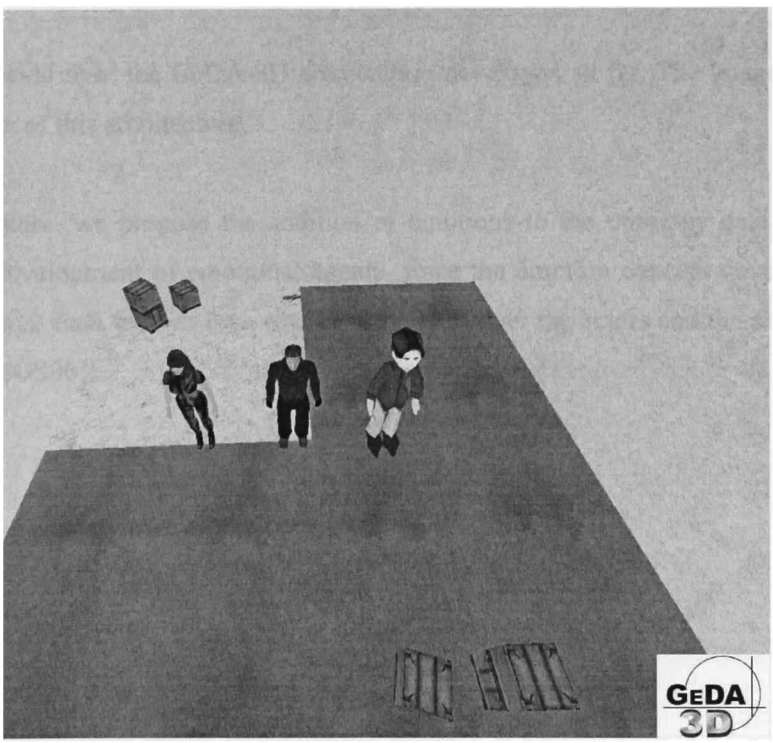


Figure 14. Description 3. Up-down View.

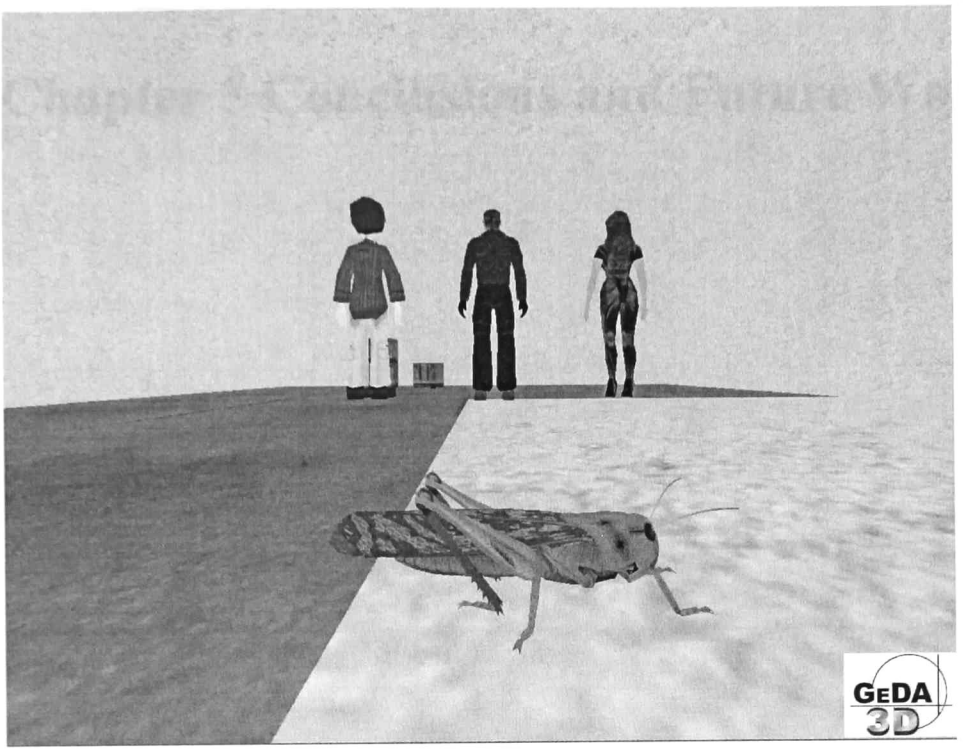


Figure 15. Description 3. Detail view.

All tests were conducted over the GeDA-3D architecture developed so far. The images correspond to the 3D render module of this architecture.

As an additional feature, we propose the addition of emotions to the ontology data. This feature is critical to help the development of emotional agents, since the emotion concept covers a really wide array of different terms, each with its own causes and effects over the actors and the actions they could be performing [RAMOS06].

Chapter 5 Conclusions and Future Work

Abstract

In our last chapter we present our final thoughts and goals derived from the research and the development of tools oriented to prove our premise. Also, future approaches to the subject of this thesis are exposed, as well as goals to be reached on future investigations.

5. 1. Introduction.

After going thought different stages of our research, we finally reach a point where we can prove our main premise: that the possibilities for the use of ontologies in the description phase of declarative modeling offers a method to resolve expression meanings, establish entities relations and resolve properties validations.

Also, we discover new applications for the mentioned knowledge bases, in the field of virtual environment description and evolution, as well as new goals to reach for VEDEL, the proposed language and its parser.

5.2. Conclusions.

The work conducted over the subject of this thesis, exposed in the previous chapter, leads to a number of conclusions about the viability of using knowledge bases such as the ontology. Such conclusions are presented next:

1. Given the challenge of analyzing the human natural speech, we found out that not only a procedure to make such analysis, and then construct a data structure that a computer could understand and use, but also the data repository needed for such task could be really complex and resource consuming.

   This led us to design a way to express descriptive sentence on a structured way. To cover such design, we defined the VEDEL language, which is oriented no only toward reduce complexity and resource consuming, but also to allow inexperience users to develop virtual scenarios, giving them a near-natural, but structured, method to express descriptions.

2. Once we defined our language, a new challenge awaited: How to store the meaning for a specific entity, such a person, an animal, an object, or even for the whole scenario or world where the scene could take place?

We have already invested some time researching on the ontology use, so we decided to use such knowledge structure to store all the necessary and pertinent data for the entities that would dwell on the environment, as well as the data to generate such scene.

3. When we started defining our knowledge database, we found out that the amount of data a single entity can hold is enough for our goals. From basic data like the name, to specific data to help to create the context for the core of the architecture, so the scene can evolve, there was not restriction on what or what not can be stored and/or expressed.

   Also, we where able to locate and access such data with relative ease, only formatting the search to accommodate the conventions defined to ingress data into the ontology.

4. Upon starting designing our parser, one of the critical points was the election of the programming language. Due to cross-platform needs, and the compatibility with the rest of the GeDA-3D architecture, we select to conduct all of the coding in Java. Also, the Protégé platform includes the necessary APIs to access, modify, and query ontologies.

   Once the parser was finalized, the result where satisfactory. All of the data was completely accessible, and the parsing of any non-include element could be handled. New elements included in the ontology didn't affect the performance of the parser, and allow expanding the possibilities by making entity adding easy.

5. After many tests, we can assure that employing knowledge databases can benefit not only the description phase for declarative modeling, but all of the process. This benefit comes in the form of data structuring, quick access, disambiguation and semantic problems solving.

   Other great advantage comes in the form of transparent access to data necessary to create the scene. This information could come in the form of explicit data for other modules, entity relationships, laws specifications or any other necessary information, such as file paths, mathematical or logical operations or scripts.

6. Finally, the integration of the parser module with the rest of the GeDA-3D architecture was successful. The results presented in this thesis were the product of the interaction between the parser, part of the Virtual Editor, with the Core and Render modules.

Finally, we can resume that the outcome of our investigation cover the next points:

1. The inclusion of an ontology in the description process of declarative modeling is valuable: we can store all kind of related information, so the process can be controlled and monitored.

2. Since all kind of data can be store, the user can express a wide array of configuration for any conceivable scenario.

3. The 3D representation can then be reduced to whatever the render module can present, or be expanded if the render can generate any kind of representation.

4. The simulation gets benefits in the form of transparent generation of the context for the environment and the agents that control the creature that dwell in it.

5. Finally, the language can be as natural-like as the ontology permits. Either we can stored molecular data (such as man, hair, run, north, above), or use especial annotations (such as HappyMan, RunFast, BehindOf).

5.3. Future Work.

Now we expose the goals we pretend reach in future works on this thesis. Even when we covered most of the aspects defined at the beginning of the investigation, new goals appeared and new requirements where evident, so we list them in the next paragraphs.

First, there is the need for a constraint solution module. Such module is necessary to prevent the generation of scenes where two entities be placed in the same area, interjecting each other. This, of course, can't be allowed, and every situation of this type must be solved.

The resolution of such problems isn't easy. Requires specialized methods to optimize such task, and it is necessary the existence of data that allow the arrangement of the various structures that can exist in a scene. For that matter, the definition and design of such module have to be taken in a very sensible way.

Other aspect we need to cover is the possibility of generate any kind of output, covering any need the user or module receiving the result for the analysis and constraint resolution may have. This also adds the possibility for the user to define specialized data for the output, and the corresponding treatment for this in the output generation process.

Such need can be covered by generating and output formatting module, which can be feed with an output sheet, which can include indications for extracting specialized output data from the ontology. This data isn't used in the process, and therefore isn't present on the model constructed by the parser, but still can be accessed and processed using such model.

The final aspect we need to cover is the ontology's content. So far, only a few elements are available both in the ontology and in the render object database. This also limits the possible scenarios to only one possibility. Then, is necessary to enlarge both, by adding new elements that can be presented by the parser, as well as the actions the entities can perform.

Such actions are now simplistic: walking, turning, running. But the recreation of more complex routines is a task the render designer must complete. This leads us to help by generating new elements and providing to the designer. This, of course, is an extra task, and cannot occupy all of the research time.

References

[ALANEN04] M. Alanen, I. Porres. *A Relation Between Context-Free Grammars and Meta Object Facility Metamodels.* TUCS Turku Center for Computer Science. ISBN: 952-12-1337-X. March, 2004.

[AUKSTAKALNISM92] Steve Aukstakalnism, David Blatner, *Silicon Mirage, The Art and Science of Virtual Reality*, Peachpit Press, 1992

[BREY03] Phillip Breay, *The social ontology of virtual environments - Criticisms and Reconstructions*, The American Journal of Economics and Sociology, January, 2003.

[BRILL97] Eric Brill, Raymond J. Mooney, *An overview of empirical natural language processing Natural Language Processing,* AI Magazine, Winter 1997.

[CHARNIAK97] Eugene Charniak, *Statistical Techniques for Natural LanguageParsing,* Department of Computer Science Brown University, 1997

[CHRONAKI90] Catherine E. Chronaki, *Parallelism in Declarative Languages.* Master Thesis. Rochester Institute of Technology. 1990

[COYNE01] Bob Coyne, Richard Sproat, *WordsEye: An Automatic Text-to-Scene Conversion System,* AT&T Labs Research, 2001

[CUDDIHY00] E. Cuddihy, D. Walters. *Embodied interaction in social virtual environments. Proceedings of the third international conference on Collaborative virtual environments.* San Francisco, CA, USA, 2000, pp 181 – 188

[DEBORAH04] Deborah L. McGuinness, Frank van Harmelen, *OWL Web Ontology Language Overview W3C Recommendation*, 10 February 2004, http://www.w3.org/TR/2004/REC-owl-features-20040210/,

[DOMINGUE98] J. Domingue, *Tadzebao and WebOnto: Discussing, Browsing, and Editing Ontologies on the Web.* In Proceedings of the Eleventh Workshop on Knowledge Acquisition, Modeling and Management, KAW'98, Banff,

Canada.

[ERIKSSON95] Eriksson, H., Shahar, Y., Tu, S.W., Puerta, A.R., and Musen, M.A, *Task modeling with reusable problem-solving methods*, Artificial Intelligence 79, pp. 293-326, 1995

[FARQUHAR96] A. Farquhar, R. Fikes, and J. Rice, *The Ontolingua server: A tool for collaborative ontology construction*, Technical report, Stanford KSL 96-26, 1996.

[FERGERSON98] W. E. Grosso, H. Eriksson, R. W. Fergerson, J. H. Gennari, S. W. Tu, & M. A. Musen, *Knowledge Modeling at the Millennium (The Design and Evolution of Protege-2000)*. 1998.

[FERGERSON02] John H. Gennari, Mark A. Musen, Ray W. Fergerson, William E. Grosso, Monica Crubézy, Henrik Eriksson, Natalya F. Noy, and Samson W. Tu, *The Evolution of Protégé: An Environment for Knowledge-Based Systems* Development, University of Washington, Stanford University, Linköping University. 2002

[GENESERTEH87] Genesereth & Nilsson, *Logical Foundations of Artificial Intelligence*, Morgan Kaufmann Publishers, Los Altos, CA, 1987.

[GÖBEL03] Stefan Göbel, Oliver Schneider, Ido Iurgel, Axel Feix, Christian Knöpfle, Alexander Rettig, *Virtual Human: Storytelling & Computer Graphics for a Virtual Human Platform,* CeBIT 2004.

[GRUBER93] Thomas R. Gruber, *A Translation Approach to Portable Ontology Specifications*, Knowledge System Laboratory, Stanford University, 1992

[ISDALE98] Jerry Isadle, *What is Virtual Reality?, A Web-Based Introduction*, http://vr.isdale.com/WhatIsVR/frames/WhatIsVR4.1.html

[KWAITER97] G. Kwaiter, V. Gaildrat, R. Caubet. *DEM²ONS: A High Level Declarative Modeler for 3D Graphics Applications.* In Proceedings of the International Conference on Imaging Science Systems and Technology, CISST'97, pages 149-154, Las Vegas, June 30-July 3, 1997.

[LARIVE] Mathieu Larive, Yann Dupuy, V´eronique Gaildrat, *Automatic Generation of Urban Zones.*

[LE ROUX03] Le Roux, O, *Constraint Satisfaction Techniques for the Generation Phase in Declarative Modeling*, PhD thesis, Universite Paul Sabatier, 2003.

[LODHA99] Lodha, Suresh K. and Arvind Verma,. *Animations of Crime Maps Using Virtual Reality Modeling Language*. Western Criminology Review 1 (2). Available: http://wcr.sonoma.edu/v1n2/lodha.html. 1999.

[MONZANI01] J.S. Monzani, A. Caicedo and D. Thalmann. *Integrating behavioral animation techniques*. In Proceedings of EUROGRAPHICS'2001, Computer Graphics Forum, 20(3) 309-318, 2001.

[OUSTERHOUT98] John K. Ousterhout. *Scripting: Higher Level Programming for the 21st Century.* IEEE Computer Magazine, March 1998.

[PLEMENOS02] Demitri Plemenos, Georges Miaoulis, Nikos Vassilas, *Machine Learning for a General Purpose Declarative Scene Modeller*, International Conference GraphiCon'2002, Nizhny Novgorod (Russia), September 15-21, 2002.

[RAMOS02] Félix Ramos, Fabiel Zúñiga, Hugo I. Piza. *A 3D-Space Platform for Distributed Applications Management.* International Symposium and School on Advanced Distributed Systems 2002. Guadalajara, Jal., México. November 2002. ISBN 970-27-0358-1

[RAMOS06] Félix Ramos, Alonso Aguirre, Jaime Zaragoza, Luis Razo, *Use of Ontologies and semi natural language to the context definition for cybernetics worlds with emotional agents*, Congreso de Electrónica, Robótica y Mecánica Automotriz, 2006, Cuernavaca, Morelos. September 2006.

[RUCHAUD02] William RUCHAUD, Dimitri PLEMENO, *MULTIFORMES: A Declarative Modeller As A 3d Scene Sketching Tool*, MSI Laboratory, University of Limoges, France, 2002.

CENTRO DE INVESTIGACIÓN Y DE ESTUDIOS AVANZADOS DEL I.P.N.
UNIDAD GUADALAJARA

El Jurado designado por la Unidad Guadalajara del Centro de Investigación y de Estudios Avanzados del Instituto Politécnico Nacional aprobó la tesis

Representation and Exploitation of Knowledge for the Phase of Description in Declarative Modeling of Virtual Environments

del (la) C.

Jaime Alberto ZARAGOZA RÍOS

el día 29 de Septiembre de 2006.

Dr. Juan Manuel Ramírez Arredondo
Investigador CINVESTAV 3B
CINVESTAV Unidad Guadalajara

Dr. Luis Ernesto López Mellado
Investigador CINVESTAV 3A
CINVESTAV Unidad Guadalajara

Dr. Félix Francisco Ramos Corchado
Investigador CINVESTAV 2B
CINVESTAV Unidad Guadalajara