Creation of Virtual Environments Through Knowledge-Aid Declarative Modeling

Jaime ZARAGOZA[a], Félix RAMOS[a], Héctor Rafael OROZCO[a] and Veronique GAILDRAT[b]

[a]Centro de Investigación y de Estudios Avanzados del IPN
Unidad Guadalajara
Av. Científica 1145, Col. El Bajío, 45010 Zapópan, Jal., México
Email: jzaragoz, framos, horozco@gdl.cinvestav.mx
[b]Universitè Paul Sabatie
IRIT-CNRS UMR,
Toulouse Cedex 9, France
Email: gaildrat@irit.fr

Abstract. In this article is explored a way for Declarative Modeling of scenarios in 3D. The use of ontologies is proposed to aid the parsing of a declarative language. This strategy solves in part the complexity of semantic analysis, which is expensive computationally in resources and time. The result presented in this paper shows how this contribution is useful mainly when the proposed language constructions have a high complexity, such as those like natural language. This research is part of the GeDA-3D project, that intent to offer to non experts users a full tool to create complex virtual environments.

Keywords. Declarative Modeling, Virtual Environment, Declarative 3D editor, Animated Virtual Creatures, Virtual Agents.

Introduction

The creation of virtual environments can be achieved with several tools, from basic 3D modelers, like Flux Studio [1], to complete suites for created 3D worlds, such as 3D Max Studio [2]. Creating complex and detailed world with these tools requires that the user have some training, and only an experienced user can create worlds with complexity such as seen in video games or in movies [3].

For the inexpert user, to generate even the simplest objects can find really difficult, and in many cases be confounded by the complexity of the 3D modeling tools. Even

more, design an animation that looks natural is much more difficult, and most of times it is achieved through the use of dedicate motion capture mechanism.

If the user could just describe the scenarios and or scenes he or she is trying to created, and let the computer generated the environment, that is, the virtual scenario and the necessary elements for animate the scene, the creation of virtual environments could be at the reach of actual users of VR. The objective of the GeDA-3D project is to facilitate the creation, development and management of virtual environment by both expert and non-expert users. In this article is exposed how generate the virtual scenarios, by making use of the declarative modeling technique, supported by a knowledge base useful for semantic analysis. We adopt the definition from [4] for declarative modeling: "a technique that allows the description of a scenario to be designed in an intuitive manner, by only giving some expected properties of the scenario and letting the software modeler to find solutions, if any, which satisfies the restrictions". This process usually is divided in three phases [5]:

1. Description: Defines the interaction language.
2. Scene generation: The modeler generates one or more scenes that match what the user describes.
3. Insight: The user is presented with the generated models, from which can choose a solution.

The basis of the creation of the virtual scenario is centered in the declarative modeling method, and is aided by constraint satisfaction problem (CSP) techniques, which help us by solving any conflict between the geometry of the entities' models. This procedure is also aided by the knowledge base.

In this paper is presented a proposal for implementing a knowledge base useful in the declarative modeling process, including the resolution of any possible conflict, semantic or geometric, through the use of such base in an algorithm for solving CPS.

The structure of this article is: Section 1 presents the problem and motivations. Section 2 is devoted to present related works. Section 3 presents the GeDA.3D's architecture. Section 4 presents the proposal to declarative modeling using an ontology. Section 5 presents conclusions and future work.

1. Related Works

There are other works that deal with generation of virtual scenarios thought declarative modeling. Next, we present a brief description of the works that draw near our project.

WordEye. Coyne and Asproad present WordEye in [6], a system developed in the AT\&T laboratories. This system allows the user to generate a 3D scene from a description written in a natural language. The system can represent anything the user writes, including representing scenarios in a literal way, but the results are static and lack interaction with the user.

DEM^{2}ONS. Kwaiter et al describe in [7] the DEM^{2}ONS modeler, a 3D-scenario generator, which works with a high level of abstraction. It allows multimodal interface and make use of ORANOS, a constraint solver. This system allows user interaction but only presents static objects.

Multiformes. William et al present Multiformes in [8], a tool specially designed for sketching scenarios in 3D. Multiformes explore all the possible variations of an scenario, but works on an incremental method, where the user must specify the geometric objects in the scenario and the relationship between them.

Next section presents GeDA-3D's architecture, this article presents declarative's module of this architecture.

2. GeDA-3D Architecture

The architecture GeDA-3D is a platform for creating, designing and executing 3D dynamic virtual environments. GeDA-3D is useful to integrate and manage distributed applications. The platform offer facilities to manage the communication among software agents and mobility services used to share other services. GeDA-3D Agents Architecture allows the user to develop behaviors assigned to the agents, which participate in the environments generated by the virtual environments declarative editor [9, 10]. Such agent architecture contains the following main features:

- *Skeleton animation engine*: This skeleton engine allows animation of virtual creatures. The engine consists of a set of algorithms allowing the skeleton to animate autonomously its members producing more realistic animations.
- *Goals specification*: The agent is capable to receive a goals specification that contains a detailed definition about the way the goals of the agent must be reached. The global behavior of the agent tries to reach the defined goals and accomplish the whole specification of the agent.
- *Skills-Based Behaviors*: The agent is capable to add skills into their global behavior. Due to we are extending GeDA-3D architecture, the skills are shared in GeDA-3D. These skills are mobile services registered in GeDA-3D which could be added to agent behaviors.
- *Agent personality and emotion simulation*: Agent personality and emotion simulation make difference in behaviors of agents having the same set of skills and the same set of primitive actions. So, the agents are entities in constantly change affecting their emotions.

Figure 1 presents the architecture of the platform GeDA-3D. This platform has been grouped in different main modules: Virtual-Environments Editor (VEE), Rendering, GeDA-3D kernel, Agents Community (AC) and Context Descriptor (CD). The striped rectangle encloses the main components of GeDA-3D: scene-control and agents-control. The VEE includes the scene descriptor, interpreter, congruency analyzer, constraint solver and scene editor. The VEE provides an interface between the platform and the user, specifies the physical laws that governing an environment, and describes a virtual scene taking place in such environment. Rendering addresses all the issues related to 3D graphics, it allows the design of virtual objects and displaying of the scene. The AC is composed by the agents that are in charge of ruling virtual objects behavior.

The scene gives to an agent a detailed description about what we want an agent does instead of how we want it does. Furthermore, this scene might involve a set of goals to a single agent, and will not be necessary that these goals must to be reached in

a sequential way. Is not necessary to give a set of actions to perform by the avatar, only is necessary give a set of goals in a sequence of primitive actions before reaching them. So, we need agents able to add shared skills into their global behavior. Therefore, behaviors of agents are needed.

A user is enabled to construct a scene using a high level language similar to human language, the user is not meant to provide the sequence of actions that the avatar must perform, instead, the user must only specifies the goals that should be achieved by the virtual creature. That is, the agents have a behavior in charge of trying to reach the goals specified in the scene description. The reason of creating this kind of scenes is to appreciate a behaviors-based simulation. That is, the user specifies what agents must do, instead of how they have to do it. Therefore, two similar specifications might produce different simulations.

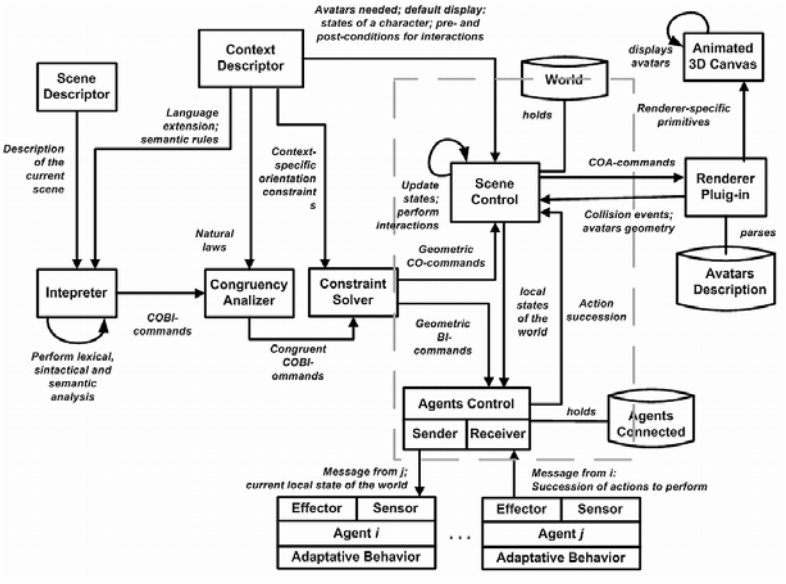

Figure 1. Architecture of the platform GeDA-3D

3. Proposal

The main concern is to assign the right meaning to each concept found in a description. Natural language is very ambiguous, so trying to parse it is a heavy duty, both in time and resources. To avoid such work charge and make easy the writing of the description, we defined a language completely oriented to describe scenarios. We call this language Virtual Environment Description Language.

3.1. Virtual Environment Description Language (VEDEL)

VEDEL is a declarative scripting language in terms described in [11,12]. This means that the user specifies what should be in the scenario, but no how it must be constructed and assumes the components required are defined somewhere else. The definition of VEDEL using the EBNF (Extended Backus Normal Form) is presented next.

Alphabet:

Σ = {[**A** − **Z** | **a** − **z**] , [**0** − **9**] **,** , **.**}
Grammar:
Description := <environment> <actor> <object>
<environment> := **[ENV]** <sentence> **[/ENV]**
<actor> := **[ACTOR]** <sentence>*** [ACTOR]**
<object> := **[OBJECT]** <sentence>*** [/OBJECT]**
<sentence> := <class>(<**,**> <property>) ***.** >
<class> := <entity> (<identifier>)\
<property> := <characteristic> | <position> | <emotion>
<characteristic> := <propid> <value>+
<position> := <posid> <identifier> | <number> <modifier>
<emotion> := <emoid> (<modifier>)
Vocabulary:
<class> := <word>
<entity> := <word>, <entity> ∈ Ontology's Classes
<identifier> := <word>
<propid> := <word>, <propid> ∈ Ontology's Classes
<posid> := <word>, <posid> ∈ Ontology's Classes
<emoid> := <word>, <emoid> ∈ Ontology's Classes
<value> := <word> | <number>
<modifier> := <word>
< identifier >:= ([A − Z | a − z] | [0 − 9])+
< word >:= [**A** − **Z** | **a** − **z**]+
< number >:= [**0** − **9**] + (**.** [**0** − **9**]+)

The language is oriented to guide the user through the construction of the description, separating the text into three blocks: Environment, Actors and Objects. Each block is composed of sentences, one for the environment section, and any number for the other blocks. The sentences are composed of comma-separated statements, the first being the entity's class and the actor's name, and the rest corresponding to the desired properties. Each sentence must end with a dot ".".

In the Environment section the user defines the context, the general setting of the scenario, the Actors section contains all the entities that can perform and be object of actions executed by other entity or the environment and perform emotions. Finally, an object is an entity that can't perform any action by its own, but can still be affected by entities or the environment.

3.2. Knowledge in Declarative Modeling

Once we have parsed the entry written in VEDEL, we need to establish the meaning for each concept stated by the user. To accomplish this task, we relay on a knowledge base. This knowledge base is implemented as an ontology.

This knowledge base stores all the relevant information for the concepts to be represented, from physical properties, such as size, weight or color, to internal properties, such as energy, emotions or stamina.

The parsed VEDEL entry is revised by an inference function, which makes use of the ontology to validate the entities and their properties, as well as validating the congruency in the scenario. This function also makes the necessary operations in the parsed entry for transform values from the string entry to arithmetic form.

The current work is to implement a Geometric Constraint Solver into the parsing solution. Thus the he user may indicate the position for any entity, or leave it to the system to arrange it. So, even when the size of the entity is considered when positioning any element in the scenario, there are still cases where some elements might present collisions. Those cases can present a variable difficulty to solve, from simply translating one of the entities to re-arrange the entire scenario. We define our CPS as follows:

- Variables. $\{Xi\}$, $i = 1 \ldots N$, where $X_i \in$ environment's entities.
- Domain. $D_i = \{\{x, y, z\}, \{\alpha, \beta, \gamma\}\}$, direction and orientation of the entity.
- Restrictions. Let $C = (Xi, Xj)$, $i \neq j$, be a collision function that returns *true* if there is a collision between entities and *false* otherwise.

The variables are composed as follows: a set $P = \{x, y, z\}$ for the spatial orientation of the entity, a set $O = \{\alpha, \beta, \gamma\}$ for orientation, and the set $G = (T,S)$ for the entity's geometry, where $S = \{x_i, y_i, z_i\} \in \Re$ is the set of the points that conform the entity, and $T = \{a_i, b_i, c_i\} \in S$, $a_i \neq b_i \neq c_i$ the triangles that form the entity.

The variables' domains are defined as follows: $D(P) \in \Re, [-\infty, \infty]$, $D(O) \in \Re, [0, 2\pi]$. G is static and does not change.

The restriction are limited to $\forall X_i \forall X_j \neg C(X_i, X_j)$ and keep the entities' order. If the user positioned an entity X_i left to another, X_j, the relationship $y_i < y_j$ for $y_i \in P(X_i) \wedge y_j \in P(X\_j)$ must be kept.

4. Results

Until now the parser for the VEDEL language is implemented and functionally working, there were established the necessary links with the GeDa-3D architecture in order to obtain a visual representation of the descriptions written in the language.

4.1. The VEDEL Parser

The parser was written in the Java language, for multiplatform capabilities and compatibility with the rest of the GeDA-3D architecture, using the JDK 1.5.0_07-b03 [13]. Our parser is basically a state machine: each section of the description corresponds to a state, as well as each sentences and each property. If the state generates an error output, the process is stopped and an error condition is arise. Each word is considered as a token and validated through the inference machine, with the exception of numbers, particular names and closing/opening constructions. If the parser machine successfully validates a term, all the data attached to that term is extracted from the ontology and stored in the environment's model. In the case of properties values, the model is used to validate them.

If the parsing process ends successfully, a conversion function is called, and using a MVC (Model View Controller) pattern, formatted in the desired output.

4.2. The GeDA-3D Prototypes

To obtain a visual output of the description written in VEDEL, we created a prototype of the GeDA-3D architecture. Such prototype consisted in a prototype kernel [14], a render working on the AVE (Animation of Virtual Creatures) project [15], and our parser, and it worked as follows: the kernel received the outputs generated by the parser (an output for the kernel and one in LIA-3D (Language of Interface for Animations in 3D, presented in [15], for the parser), generate the necessary agents, and the AVE output was sent to the render module, where the scenario was composed, rendered and presented to the final user. However, due to the limited number of available 3D models and since the Render module is still a work in process, we construct a new visual port in order to present our results, based on the X3D standard [16]. Next, we present, in figure 2 and 3 an example of a description written in VEDEL and some examples by this prototype.

```
[ENV] //environment section
   forest.
[/ENV]

[ACTOR] //actors section
   Man Adam, center.
[/ACTOR]

[OBJECT]//objects section
   CenterTable Main, left Adam, cristal clear blue.
   Chair one, front Main, facing north.
[/OBJECT]
```

Figure 2. Example of a description written in VEDEL

Figure 3. Examples obtained through the Parser Prototype.

5. Conclusions

The problem we contribute in this article is the creation of virtual scenarios using declarative modeling. This problem is important because designing virtual scenarios is difficult and consuming in time even for expert user of appropriated tools. In this article we contribute mainly in the definition of concepts and retrieving necessitated information to process the user request.

More specifically the way proposed use a knowledge base in the three phases constitutes the declarative modeling: *Description* to get semantic elements needed to verify a description, which is the input to the system. For the *Model Creation*, the modeler will use the knowledge database to obtain necessitated information to create

the model; to achieve this task the modeler has next two options: First, use a restriction satisfaction algorithm and second, use a geometric restriction solver. Finally, the user use the knowledge base in the *Insight* phase to obtain information of those requirements not satisfied, in case there was no solution.

The proposed solution has next two very important advantages: First, the solution obtained with in this way can be used in system able to evolve a scene as the GeDA-3D project described in this article. Second, include environmental information imply just increasing the knowledge database; this way, the declarative editor can be general.

Some of the results obtained include: propose a structured method for creating and editing description and tools necessitated to use this structure method, like lexical and semantic analyzers. The implementation of this prototype is using standards as *X3D* to visualize the generated scenario from the respective declaration. Those results are important because we got that using a knowledge database allows create a more general virtual editor. That is, include more types of scenarios need just increase with corresponding information the knowledge database.

Future works include study how useful can be the use of the knowledge database in the resolution of geometric conflicts.

References

[1] Keith Victor. Flux studio web3d authoring tool, 2007.

[2] Autodesk. 3ds max 9 tutorials. Online webpage, http://usa.autodesk.com/adsk/servlet/item?siteID=123112&id=8177537, 2007.

[3] J.S. Monzani, A. Caicedo, and D. Thalmann. Integrating behavioral animation techniques. In *EG 2001 Proceedings*, volume 20(3), pages 309–318. Blackwell Publishing, 2001.

[4] Demitri Plemenos, Georges Miaoulis, and Nikos Vassilas. Machine learning for a general purpose declarative scene modeller. In *International Conference GraphiCon'2002, Nizhny Novgorod (Russia), September 15-21*, 2002.

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

[6] Bob Coyne and Richard Sproat. Wordseye: An automatic text-to-scene conversion system. In *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 487–496. AT&T Labs Research, 2001.

[7] G. Kwaiter, V. Gaildrat, and 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, 1997.

[8] William Ruchaud and Dimitri Plemeno. Multiformes: A declarative modeller as a 3d scene sketching tool. In *ICCVG*, 2002.

[9] H. Piza, F. Zúniga, and F. Ramos. A platform to design and run dynamic virtual environment. I*nternationalConference on Cyber Worlds, Tokyo Japan*, pages 18–20, November 2004.

[10] F. Ramos, F. Zúniga, and H. Piza. A 3d-space platform for distributed applications management. *International Symposium and School on Advanced Distributed Systems 2002 Guadalajara, Jal., México.*, November 2002.

[11] Catherine E. Chronaki. *Parallelism in Declarative Languages*. PhD thesis, Rochester Institute of Technology, 1990.

[12] John K. Ousterhout. Scripting: Higher level programming for the 21st century. *IEEE Computer Magazine*, 31(3):23–30, March 1998.

[13] SUN Microsystems. The java se development kit (jdk). Online webpage, http://java.sun.com/javase/downloads/index.jsp, 2007. Last visited 02/01/2007.

[14] Alonso Gutierrez Aguirre. *Núcleo GeDA-3D*. PhD thesis, Centro de Investigació y de Estudio Avanzados del IPN, Unidad Guadalajara, 2007.

[15] Alma Verónica Martínez González. *Lenguaje para Animación de Creaturas Virtuales*. PhD thesis, Centro de Investigación y de Estudio Avanzados del IPN, Unidad Guadalajara, 2005.

[16] X3D. The virtual reality modeling language - international standard iso/iec. Online webpage, http://www.web3d.org/x3d/specifications/, 2007. Last visited 06/01/2007.