ISSN 1870-4069

# Creating an Ontology to Represent Qualitatively a Scene in a Virtual Reality Environment

Eduardo Eloy Loza Pacheco[1], Mayra Lorena Díaz Sosa[1],
Miguel Jesús Torres Ruiz[2], María Eugenia Canut Diaz-Velarde[1]

[1] Universidad Nacional Autónoma de México, Acatlan, Edo. Mex., Mexico
{eduardo.loza,mlds,marucanu}t@apolo.acatlan.unam.mx
[2] Instituto Politecnico Nacional, Centro de Investigación en Computación,
Mexico City, Mexico
mtorres@cic.ipn.mx

**Abstract.** Ontologies are widely used as a tool for representing knowledge in Artificial Intelligence more specifically in qualitatively knowledge representation and reasoning, for example, to represent concepts and their relationships. On the other hand, virtual reality has several applications in different fields: such as in medical systems, computer-aided design and education as a virtual learning environment. In both cases, qualitative representations are necessary to perform any qualitative reasoning task. In this paper, we see VRML and Java 3D. Which are formal languages are used to describe objects in 3D. We analyze the similarities between them to define an application ontology with the aim to represent virtual reality environments, independently of the programming language. Then a spatial ontology is defined to describe topological, directional and metric relation which can be used to describe the basic operations in a 3D scene to build a complex environment. Finally, these two ontologies are seen that can be constructed independently but integrated together whether needed.

**Keywords:** ontologies, knowledge-based systems, virtual reality, spatial relations.

## 1 Introduction

One of the main topics of Artificial Intelligence is the sharing of knowledge [1,2]. The processing of knowledge structures must be independent of the programming language. An ontology provides systems based on knowledge of interoperability through a language with which it is possible to define concepts, vocabulary and relationships that can be used to describe the domain of interest.

Ontologies are used in various fields, such as education [1], medicine, arts [4,5] and design [6,7], to name just a few examples. In general, it is sought with them to help the human expert to make better decisions and, in most cases, in real time [3] either through simulations, virtual reality, knowledge bases or a combination of these.

On the other hand, since the end of the 90s various technologies have given rise to different programming languages that have facilitated the production of graphic representations. Today there is a wide variety of useful languages to describe virtual

reality scenes: from markup languages such as VRML [8] and X3D [9], to technologies such as OpenGL, available in programming languages such as Java3D [10], Python and C ++.

In the midst of this boom some limitations have arisen. The great variety of graphic technologies and programming languages has led to a heavy dependence on software developments. That is, the code of a program must be almost completely rewritten each time the developer chooses or must migrate to a different language. To alleviate this difficulty, this work proposes three ontologies with the purpose of describing a virtual environment capable of reusing code oriented to perform transformation operations on objects: translation, rotation and scale.

## 2 Coding a Virtual Environment

Below is a comparative example of the code that defines the transformations of a cube in the VRML and Java3D languages. In both cases, the translation and rotation operations are described. As can be seen, the syntax of one and the other is totally different. In the first case, the operations are done on a "shape node" (technical name of the cube), whose size is defined through a vector with starting point at (0,0,0) and ending at (5 5 5). Figure 1 (lines 1-4) shows the translation of the cube on Z and its rotation in X and Y. Figure 2 shows the graphic output obtained.

```
1  #VRML V2.0 utf8
2  Transform{
3  translation 0 0 -20.5
4  rotation 1 1 0 .6
5  children[
6
7              Shape {
8                    appearance Appearance{
9                    material Material{ diffuseColor 0 1 1}
10
11                   }
12
13             geometry Box {
14                       size 5 5 5
15                 }
16              }
17
18
19         ]
20 }
```

**Fig. 1.** VRML Code to transform a cube.

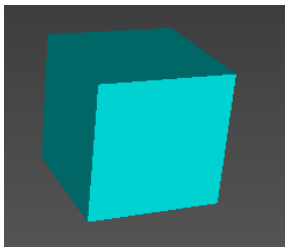

**Fig. 2.** VML graphic output.

In the second case, the Java3D, the code changes drastically[1]. For example, in the creation of the scene two objects called rotation and *segrotation* are required, in order to multiply the rotations. These objects are defined as Transform3D type and are necessary to perform the rotation at a given angle and around an axis, through the methods *rotX* and *rotY*. Finally, the rotations in X and Y are combined with the *mul* method. Figures 3 and 4 show the code and graphic representation developed with Java3D, respectively (See Figures 3 and 4).

```
1  public BranchGroup crearEscena(){
2
3      BranchGroup objRoot=new BranchGroup();
4
5      Transform3D rotacion=new Transform3D();
6      Transform3D segrotacion=new Transform3D();
7
8      rotacion.rotX(Math.PI/4.0d);
9      segrotacion.rotY(Math.PI/3.0d);
10     rotacion.mul(segrotacion);
11
12     TransformGroup objrotacion=new TransformGroup(rotacion);
13
14
15     objRoot.addChild(objrotacion);
16     objrotacion.addChild(new ColorCube(0.5));
17
18
19     objRoot.compile();
20
21     return objRoot;
22
23
24 }
```

**Fig. 3.** Java3D Code to transform a cube.

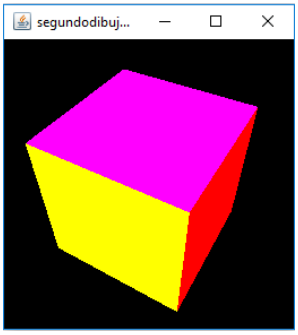


**Fig. 4.** Java3D graphic output.

As can be seen, the syntaxes and philosophies are very different from one another. However, the graphic outputs are similar (figures 2 and 4). Also, the semantic elements that modify the objects are the same, that is, the transformations, the material and the geometry.

[1] Only the fragment of code concerning the transformations that determine the position of the object in the virtual environment (scene), as well as its methods are discussed. The main method as well as the ones that defines the framework and the constructor of the class are omitted.

## 3 Designing the Graphic Knowledge Representation

In general terms, objects represented in space can be described from their properties, based on their hierarchy and behavior. Next we will show the description of 3D scenes through semantic trees for VML and Java3D in the context of the transformations described.

VRML is a markup language –like HTML– whose main purpose is to build a 3D scene. Figure 5 shows the semantic tree corresponding to a graphic representation constructed from the root. For example, the "transform" node has the transformation actions: translation, rotation and scale, while the "shape" node has three attributes that define it: a geometry (in this case it is a cube), an appearance and a type of material.

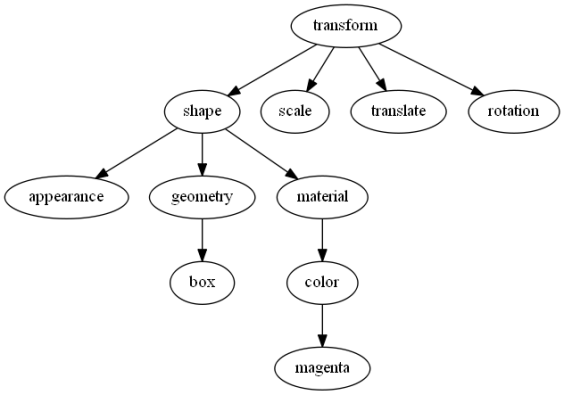


**Fig. 5.** Graphic representation of a scene produced with VRML.

In Java 3D, however, there is a general structure for the design of a virtual environment (Figure 6). This graph can be used as a form to share knowledge among programmers, facilitating the modification of the structure and the realization of changes or for the recycling of design elements.

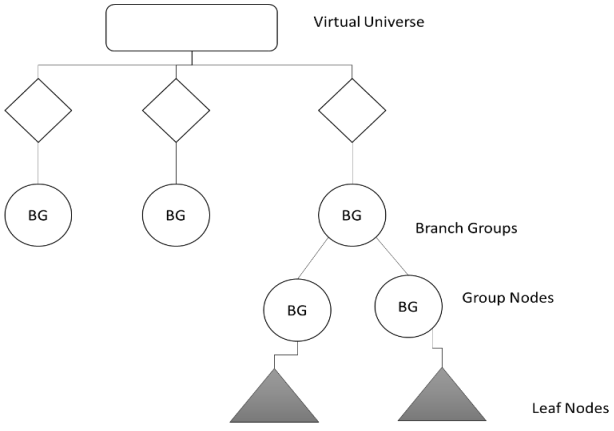


**Fig. 6.** Graphic representation of a scene produced with Java3D.

## 4 Definition of Ontologies

The proposed ontologies are independent of each other to ensure high cohesion and low coupling between them, similar to the way classes are modeled in object-oriented software engineering [11]. First, an ontology "Object property" is defined, similar to the representation observed for VML as we saw in Figure 5. This structure can be populated with different objects such as instances or leaves (see Figure 7). This ontology could be used to describe complex objects, for example, a robotic arm.

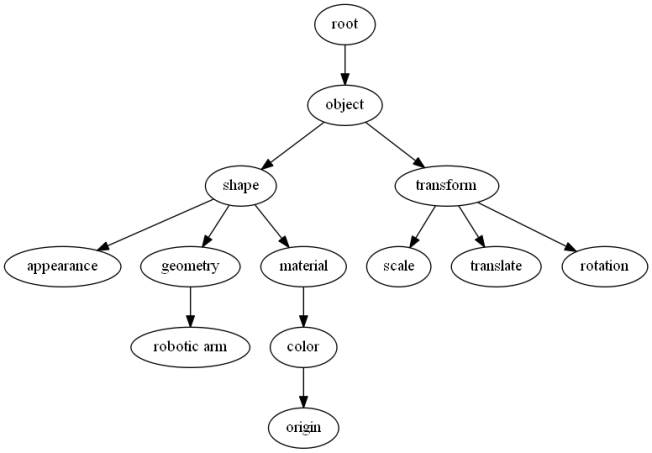


**Fig. 7.** Ontology of properties.

Once the ontology of properties has been defined, the topological ontology must be established. Figure 8 shows a scene with three cubes, which can be described from their relationships, as in [12], from an ontology in which - beyond the geometry, appearance or transformation of objects - its topological relationships are highlighted (Figure 9). In this example, it is not necessary to describe the type of object. However, this would be useful if it were required to map the ontology with another knowledge structure.

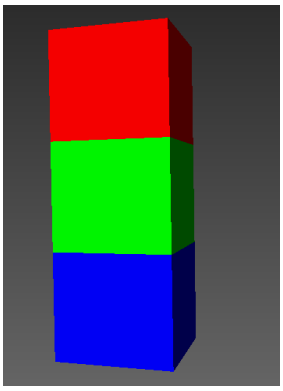

**Fig. 8.** Three cubes in a 3D environment.

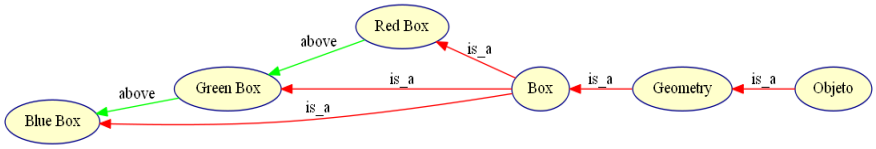


**Fig. 9.** Knowledge representation of a virtual environment.

By combining both ontologies it is possible to create complex objects based on geometric primitives, materials and transformations, as well as the spatial relationships between them. This would facilitate a recursive construction scheme, object by object, with the definitions of the previous structures.

## 5 Conclusion

As we exposed, ontologies can be proposed for different purposes, among them, to define transformations and properties. But is clear that this will require the construction of a specific middleware for each programming language so that each ontology is connected to its graphical implementation, as well as a user interface to build the virtual environment. The advantage of this strategy is that knowledge can be adapted, reused, and updated for future work. Likewise, the segmentation of knowledge bases improves the execution processes in graphics loading and avoids unexpected overheads. The proposed work does not consider a virtual environment in motion so it can be extended using multiple states of the ontology.

**Acknowledgements.** The authors would like to thank to CONACyT and DGAPA-UNAM (PAPIME PE104718 and PAPIME PE 304717) for the funds to support this work.

## References

1. Gruber, T.: Ontolingua: a mechanism to support portable ontologies (1992)
2. Martini, R., Librelotto, G., Henriques, P.: Formal description and automatic generation of learning spaces based on ontologies. In: Procedia Computer Science, 96, pp. 235–244 (2016)
3. Nuñez, D., Borsato, M.: An ontology-based model for prognostics and health management of machines. Journal of Industrial Information Integration, 6, pp. 33–46 (2017)
4. Béhé, F., Galland, S., Gaud, N., Nicolle, C., Koukam, A.: An ontology-based metamodel for multiagent-based simulations. Simulation Modelling Practice and Theory, 40, pp. 64–85 (2014)
5. Cavazza, M. et al.: Intelligent virtual environments for virtual reality art. Computers & Graphics 29(6), pp. 852–861 (2005)
6. Mahdjoub, M., Monticolo, D., Gomes, S., Sagot, C.: A collaborative design for usability approach supported by virtual reality and a multi-agent system embedded in a PLM environment. Computer-Aided Design 42(5), pp. 402–413 (2010)

7. Chen, T.: Knowledge sharing in virtual enterprises via an ontology-based access control approach. Computers in Industry 59(5), pp. 502–519 (2008)
8. Carey, R., Bell, G.: The annotated VRML 2.0 reference manual (Vol. 15). Reading, MA: Addison-Wesley (1997)
9. Web3d.org. What is X3D? | Web3D Consortium. [online] Available at: http://www.web3d.org/x3d/ (2018)
10. Pratdepadua, J.: Programación en 3d con java 3d. Ra-Ma (2003)
11. Pressman, R.: Software engineering: a practitioner's approach. Palgrave Macmillan (2005)
12. Guarino, N., Giareta, P.: Ontologies and Knowledge bases (1995)