Coupling Ontologies with Graphics Content for Knowledge Driven Visualization

Evangelos Kalogerakis[1] Stavros Christodoulakis[1] Nektarios Moumoutzis[1]

TUC/MUSIC - Technical University Of Crete [2]

ABSTRACT

A great challenge in information visualization today is to provide models and software that effectively integrate the graphics content of scenes with domain-specific knowledge so that the users can effectively query, interpret, personalize and manipulate the visualized information [1]. Moreover, it is important that the intelligent visualization applications are interoperable in the semantic web environment and thus, require that the models and software supporting them integrate state-of-the-art international standards for knowledge representation, graphics and multimedia. In this paper, we present a model, a methodology and a software framework for the semantic web (Intelligent 3D Visualization Platform – I3DVP) for the development of interoperable intelligent visualization applications that support the coupling of graphics and virtual reality scenes with domain knowledge of different domains. The graphics content and the semantics of the scenes are married into a consistent and cohesive ontological model while at the same time knowledge-based techniques for the querying, manipulation, and semantic personalization of the scenes are introduced. We also provide methods for knowledge driven information visualization and visualization-aided decision making based on inference by reasoning.

CR Categories and Subject Descriptors: H.5.1 [Multimedia Information Systems]: Artificial, augmented, and virtual realities; I.3.6 [Computer Graphics - Methodology and Techniques]: Languages - Standards

Additional Keywords: ontologies, web graphics, semantic driven visualization, intelligent virtual environments, domain knowledge

1 INTRODUCTION

In this paper, we present an interoperable framework for the integration of virtual reality scenes with semantic information and methodologies and tools for the exploitation of this rich framework for many highly desirable functionalities like semantic querying, interaction, personalization and construction of scenes with inference. The semantic enrichment of scenes can play an extremely important role in enabling the viewers to query, understand and interact with the usually complex and incomprehensible visualized information, in simple, intuitive and user-friendly ways and allowing them to identify 3D objects or sets of them based on their graphical and semantic properties and relationships with other objects in the scene at a time. Interactive queries, such as "what is this object which I clicked upon with my mouse?", "what is the functionality, the behavior, the role of the component represented by this object?" or "show all the components of the same type or of the same hierarchical level of abstraction and hide all the others from the scene" should be answered with the appropriate visual and textual response. Such queries could allow users to explore the external and inner parts of the models and understand their behavioral and functional patterns, which usually carry mappings with processes and events of the real world. The mere visual display of complex and large amounts of information is not sufficient itself to answer such queries especially when viewers do not possess sophisticated knowledge of the domain related to the visualized information.

Designers are also interested in generating, updating or deleting graphics content from the scenes based on the semantics of the visualized information. For example, user commands, like "apply a half-transparent material to all the objects representing the concept X (e.g. a window)" or "apply a transformation to all the objects representing the concept Y" or even "create a 3D object for every instance of concept Z accordingly", could significantly facilitate or even automate the manipulation of the graphics content according to the specific domain knowledge coupled with the scenes. These manipulation commands can also be personalized with the help of user profiles. A simple example of personalization of the scenes would be: "apply a specific background stored in the user's profile to the scene" or "apply this material taken by the same user's profile to every object representing the concept W". User profiles combined with domain knowledge could automatically adapt the content of the scenes to designers' preferences.

The incorporation of semantics into 3D models and scenes can greatly enhance the retrieval capabilities of search engines. Lately, there has been significant research in this field [2], where users enter keywords or 2D/3D sketches to find a target 3D object. A search engine implementing semantic similarity methods [3] will improve both the precision and the recall of the retrieval of 3D models as was shown in many information retrieval multimedia and natural language interface environments [10][11]. As the semantics of the scene are closely bound to lexical data for natural language parsing and interpretation, they could also be used for language-based interactive manipulation of the scene [15].

The approach of this paper is based on semantic web technologies. Ontologies provide us the theoretic and axiomatic basis to underlying knowledge bases. The scenes are explicitly treated as semantic network style knowledge bases where any operations on them are modeled with a system of logic in a formal and uniform manner. Based on this formalization, we describe the supporting framework to perform reasoning by inference on the content and the semantics of the scenes in order to perform knowledge driven visualization and visualization-aided decision making.

The paper is organized as follows: section 2 presents related work and contributions. Section 3 presents our approach for the development of upper-level graphics ontologies. Section 4 presents our ontological model that defines general types of mappings between the content of the scenes and domain-specific ontologies so that we natively achieve their systematic semantic enrichment. Section 5 refers to the implementation supporting our model and methodology. Section 6 describes knowledge-based techniques for the construction, manipulation, personalization, querying of the

[1] email: {vkalog, stavros, nektar}@ced.tuc.gr

[2] mail address: Technical University of Crete Campus, 73100 Kounoupidiana, Crete, Greece

IEEE Virtual Reality 2006
March 25 - 29, Alexandria, Virginia, USA
1-4244-0224-7/06/$20.00 ©2006 IEEE

scenes combining their integrated semantics. Finally, section 7 concludes and presents future work plans.

2 Related Work and Contributions

Although several researchers advocate the attachment of keywords to graphics scenes in order to represent non-visual semantics, very few approaches exist in the literature that try to systematically integrate complex semantics expressed in domain ontologies with 3D and virtual reality scenes. Often these approaches focus on supporting a specific application and they do not provide a generic standard-based platform for developing intelligent interoperable applications in the web.

The MUG platform [4] is a collaborative 3D environment for authoring design semantics supporting knowledge-based computer design in CAD applications. It uses the DAML language for expressing behavioral and functional properties of design components using domain ontologies. The designers create conceptual designs for new products with a set of primitive shapes or pre-existing geometries and they annotate the designed objects with their behavioral semantics in the specific design. The approach is very useful for supporting the first stages of CAD design and documentation, but it specifically aims at the design objective without providing a generic platform with mechanisms to support diverse applications.

A storage, archival, and sketch-based query and retrieval system for 3D objects has been developed in [5]. The project focused on anthropology applications and developed an XML schema to organize the semantics related to anthropology applications. It also introduced segmentation and feature extraction algorithms for extracting the specific semantic information for indexing and querying purposes. This work is also mostly application specific.

Recent researchers try to integrate 3D graphics with semantic information to create generic platforms supporting intelligent virtual reality and simulation environments. In [6], knowledge related to rooms, groups, roles etc is encapsulated. In [7], intelligent agents and autonomous avatars, which represent the users, are incorporated into intelligent virtual environments. In [8], an external file format for representing AI, graphics, physics and other simulation related content is introduced. A very generic and expandable model for this line of research is presented in [9] and [16] where a Functionally Extendable Semantic Network (FESN) has been introduced to capture semantic knowledge as well as physical simulation specific knowledge in a central knowledge base supporting the design of intelligent virtual environments. In addition, the problem of data synchronization between the different graphics and physics databases has also been addressed in [17].

For the development of platforms supporting interoperable semantic web applications for 3D and virtual realty, the use of standards as the basis of the platform development is very important. In [18] and [19], a model to represent interactive digital objects with multiple visual representations and functionalities based on MPEG-7/21 standards is presented. A generic methodology on how to integrate domain knowledge expressed in ontologies with MPEG-7 content descriptions is shown in [10].

The research presented in this paper also aims at the development of software frameworks (models and platforms) for the development of interoperable virtual reality applications. We focus on the general problem of associating 3D graphics and virtual reality scenes with complex semantics knowledge expressed in domain ontologies in a systematic, standard-based, software engineering approach so that firstly, many diverse intelligent 3D and virtual reality applications can be built with less effort on top of this framework and secondly, the framework is based on standards including graphics (like X3D/VRML), multimedia (like MPEG-7/21) and ontology representation (like OWL) so that the applications for scene design and management are interoperable in the semantic web environment. More specifically, in our methodology, we first raise the description of the graphics content up to the ontologies layer of the Semantic Web, albeit we ontologically describe the content of the scenes by using the W3C Ontology Web Language - OWL (http://www.w3.org/2004/OWL). OWL has become the standard mark-up language to represent an area of knowledge with its semantics allowing intelligent applications (agents) to perform automated reasoning on the web. In order to achieve the ontological description of the scenes, we capture all the necessary primitives concerning not only the design of shapes in graphics applications but also all the necessary concepts which are commonly used to build virtual reality scenes (such as transformations, animation, navigation etc) without replacing existing graphics standards and languages or the content of low level graphics databases. The result is the construction of OWL ontologies playing the role of abstract visualization libraries that can be accessed and imported by graphics designers to describe their scenes and communicate their content across the web. We also provide an OWL model for integrating domain knowledge about the scenes and their objects using domain ontologies where their graphical and conceptual meaning clearly remains distinct. Then, we show how this uniform environment can be used to provide a comprehensive set of mechanisms for knowledge-based querying, set-oriented retrieval, interaction, browsing, manipulation (e.g. insert, delete, update objects or sets of objects), semantic personalization and automatic construction of the scenes with inference. Our model and methodology are generic; they fully integrate standards for 3D graphics and semantics and are applicable to diverse knowledge domains of virtual reality applications. The I3DVP platform we have developed incorporates all this functionality.

3 Upper Level Graphics Ontologies

The key objective of our approach is to couple the 3D content of the scenes with their domain specific semantic descriptions into a cohesive ontological model in a uniform manner. The first step to achieve this goal is to create an ontological description of the scene capturing the well-accepted primitives and concepts that are commonly used to build 3D models and scenes by the graphics and virtual reality communities. These upper level ontologies play the role of abstract visualization libraries being modeling-oriented rather than presentation-oriented, i.e. they focus on the essential nature of 3D models and their animation ignoring the low-level details of creating and presenting graphics in them [12]. More intuitively, they target on "how to describe geometry objects with their transformations and their animation in the scene rather than what their geometry is". The graphics ontologies also capture common user interaction and scene navigation primitives. The graphics ontologies are expressed with the latest W3C standard Ontology Web Language of the Semantic Web, which provides plethora of description logics that can be exploited in order to provide a complete and enriched representation of graphics and virtual reality concepts and primitives. In this section, we present our first approaches to build upper level ontologies – abstract visualization libraries for graphics and virtual reality communities.

Our first OWL-based 3D graphics ontology, called OntologyX3D, is merely based on the well-accepted VRML and its considered successor X3D standard [13]. In order to build the ontology, we have mapped the X3D node elements representing graphics and virtual reality concepts into OWL classes. The OWL classes provide an abstraction mechanism for grouping the graphics resources, defined within the X3D schema, for geometry and appearance of objects, scene navigation, lights, environmental

Proceedings of the IEEE Virtual Reality Conference (VR'06)
1087-8270/06 $20.00 © 2006 IEEE

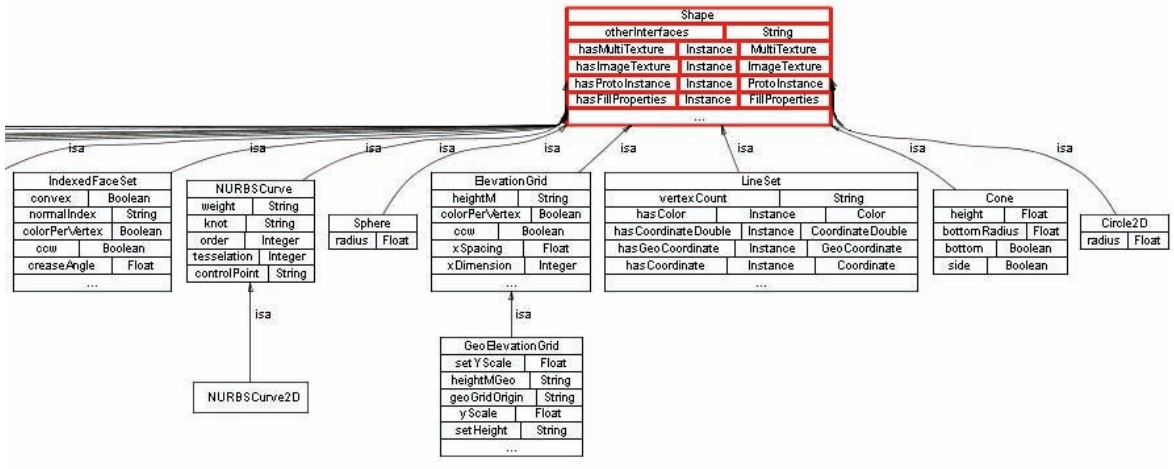

Figure 1: Graphical representation of part of the Shape in OntologyX3D

effects, sound, interpolators (for linear animation), sensors, events, humanoid animation and geography. Then we used the OWL description logics that enable the definition of class hierarchies and object properties relating the OWL classes in order to create a complete and valid ontological description of scene graphs like those defined with VRML and X3D. We present a straightforward example illustrating a small part of the ontology concerning the concept of Shape in figure 1.

Designers can import the defined OWL classes and generate instances of them, called OWL individuals, that represent facts about the whole content of the scenes they intend to create. The users can choose to describe e.g. the existence of a Shape in the scene without specifying its exact graphical representation (by instantiating only the superclass) (level 1, most general description) but, if they wish, they can also specify the exact type of geometry to be imposed on the existence of this object (level 2 description). Then, optionally, they can also define multiple different external datasets from heterogeneous resources to be coupled with the attributes of the geometry of the object (level 3 description) or finally, they can explicitly define these attributes (level 4, most detailed and constrained description). The references to external heterogeneous resources can represent vertices and indices of polygonal objects, texture coordinates, control vertices, weights and knot vectors of NURBS curves and surfaces etc. This last extension becomes extremely important as we would like our model to be independent from specific low level graphical representations, avoid to communicate large datasets across the web and load all or parts of them on-demand in browsers and visualization engines when needed. The references to external datasets are encapsulated into special OWL classes that declaratively describe their location (local or remote files, external XML node attributes, columns in tables of a relational database or finally web services), their input parameters (which can be data types properties of other existing individuals) and finally the coupled data type property of the related individual e.g. representing the coordinates of the vertices and the indexes of the polygons of a mesh.

The set of the OWL individuals with the corresponding data type and object properties eventually describe the scene graph with all the possible combinations of the positions of nodes in it. We note that the individuals can also be generated with knowledge-based methods that we will present in section 6. Figure 2 illustrates a characteristic example. The OWL files are triple-based graphs with nodes representing classes, individuals or simple data types (e.g. integers, strings, double-precision numbers etc) and with edges representing object and data type properties. We use OWL graphs to give multi-level descriptions of scene graphs while at the

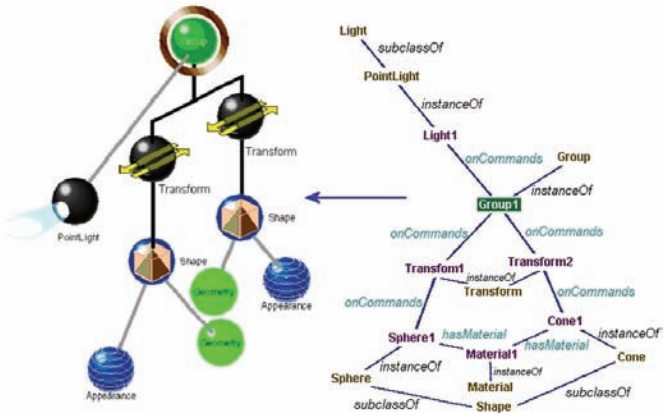

Figure 2: A triple-based OWL graph describing a scene graph

same time we can directly map the involved classes and individuals with their properties to other classes and individuals of other domain specific ontologies in order to achieve the native semantic enrichment of the scenes (as we will show in section 4).

The description of a scene with OWL offers several other important advantages. First of all, we are able to express semantic interrelationships between nodes that cannot directly be expressed in the strict hierarchical model of VRML/X3D. For example, we can describe the notion of a shared object belonging to multiple groups (where the USE attribute defined in VRML and X3D is not applicable or results in the generation of independent objects [14]). In general, the ontological description of scenes allows us to describe meaningful and valid semantic many-to-many relations in scenes where children nodes are allowed to share more than one parent nodes in the scene graph in order to define shared appearances (like Material1 in figure 2), shared objects in grouping nodes, shared transformations, colors, coordinates, normals, shared joins and segments (for humanoid animation) and finally shared semantic metadata. Moreover, we can enrich the above relations with OWL description logics; e.g. the object property "onCommands", defining a parent-child relationship between nodes in the scene graph, is a transitive property. This results in the automated inference that if command (node) A (a geometry object, a transformation etc) is semantically child of B and B is semantically child of C then A is considered as child of C. We also define the inverse property of "onCommands" which is the "parentCommands" (and is transitive, too). A practical example of application for this type of inference is that if a user query requests to find all

the objects in a group or all the distinct transformations applied to a shape independently of the other intermediate nodes in the group, an OWL reasoner will have no problem in directly finding the right answer (thus, there is no need to implement any complex algorithms to traverse the nodes in the OWL graph).

We have also decided to extend the graphics ontologies beyond the scope of OntologyX3D in order to support the description of advanced modeling and animation concepts defined in the libraries of modern graphics tools such as Alias' Maya that could be used by more demanding designers. These additional ontologies support the description of the following object-centered operations a) create surfaces from curves e.g. revolve, extrude, loft curves etc, b) create new surfaces from existing ones e.g. align, attach, intersect, stitch, trim, boolean operations on surfaces etc, c) manipulate surfaces by altering their components e.g. move their control vertices, insert/delete their knots etc, d) create new curves from existing ones e.g. align, detach, close curves etc, e) manipulate curves by changing their control vertices, knot vectors, weights etc, f) create polygonal objects from existing ones by applying merging, separating, boolean operations etc, g) manipulate polygonal objects by altering their components e.g. move the polygon vertices or edges, merge, delete, append polygonal facets h) more camera functions such as dolly, roll, track etc, i) some advanced animation concepts such as animation with driven keys, blending transitions between different animations, non-linear animation etc and finally, j) special 2D and 3D surface and volume shaders and textures.

An example of how designers could describe a 3D model using the modeling primitives defined in the extensions of the graphics ontologies is depicted in figure 3. The figure shows an OWL graph that describes a loft operation on five curves that make up a 3D object:

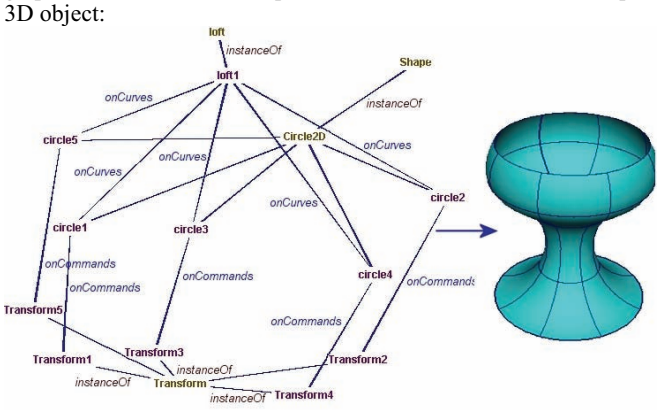

Figure 3: An OWL graph describing a scene with a 3D cup

Finally, we stress that OWL provides us with internal mechanisms to ensure the validity of the content of the scenes. This is achieved not only by the class and property definitions of the ontologies but also by value restrictions (that put constraints on the range of data and object type properties when applied to particular class descriptions) and cardinality constraints (that put constraints on the number of values a property can take e.g. an instance of loft class should take at least two curves as input – a min cardinality of two is applied to the onCurves property of the loft class).

4 Incorporating Domain Knowledge into the Scenes

The ontological description of the 3D content in scenes allows their uniform native (OWL-based) semantic enrichment with domain knowledge by creating mappings between the graphics objects created using the graphics ontologies and any other specific domain ontologies shared by communities in the World Wide Web. These mappings ensure that the graphical representation and the conceptual meaning of the objects remain distinct. The ontological mappings are easy and simple to define as their nature is close to the human perspective of describing objects and processes of the real world. For example, when we perceive an object, we cognitively map it with a concept or another object we know well. Alternatively, if we are not sure about the nature of the perceived object, we map it with a similar concept (or concepts) that partially describe it or with other objects that look like it. Alternatively, we can say that the object is part of larger known concept hierarchy (taxonomy). The "nature" of these mappings should carefully be examined to acquire model behavioral and functional meaning.

We have defined fifteen primitive types of semantic mappings between concepts, roles and individuals of the graphics ontologies and other domain ontologies in OWL. These primitive types can be used by the designers for the efficient integration of semantic knowledge with the 3D content. Most of them correspond to relations defined in the "SemanticBaseSemanticBaseRelation" description scheme of the MPEG-7 standard, which describes semantic relations between semantic entities [21]. We have provided OWL representations of these semantic relations and extended them to specifically support the mappings between the graphics ontologies we provide and the ones of the knowledge domain. The following definitions are the general semantic relationships supported in our model:

1. "represents" mappings: an individual from the graphics ontologies represents a specific domain concept e.g. a NURBS curve represents the outer membrane of a cell.
2. "equivalence" mappings: a graphics class, an individual or a property is mapped to a concept, role or individual of the specific domain ontology e.g. every NURBS curve represents a membrane or a polygon mesh (instance of the IndexedFaceSet) represents a component in a specific cell.
3. "similarity" mappings: a graphics class, an individual or a property is similar (but not equivalent) to a specific domain concept, role or individual. The similarity has also a weight with values in the range (0, 1).
4. "disjointness" mappings: a graphics class, an individual or a property is incompatible with a specific domain concept, role or individual e.g. a specific polygon mesh is definitely not a cell component.
5. "is part of" mappings: a graphics individual or a class is part of a specific domain concept or individual e.g. an interpolator (animation) of a polygon mesh is part of a cell biological function.
6. "has parts" mappings: is the inverse mapping of the above one e.g. a polygon mesh has parts of a cell nucleus.
7. "oppositeTo" mappings: a graphics individual or a class has a function represents the opposite in meaning to a specific domain concept.
8. "refines" mappings: a graphics individual or a class adds detail to the meaning of a specific domain concept.
9. "isRefinedBy" mappings: is the inverse mapping of "refines".
10. "generalizationOf" mappings: a graphics individual or a class is a kind of generalization in the meaning of a specific domain concept.
11. "specializationOf" mappings: a graphics individual or a class is a kind of specialization in the meaning of a specific domain concept (inverse mapping of "generalizationOf").
12. "hasFunction" mappings: a graphics individual or a class has a function represented by a specific domain concept.
13. "hasBehaviour" mappings: a graphics individual or a class has a behavior represented by a specific domain concept.
14. "domain relationship" mappings: a property of a graphics class also belongs to the domain of a specific domain concept e.g. the diffuseColor of the material class also belongs to the domain of the class melanin (representing its color characteristics).

Proceedings of the IEEE Virtual Reality Conference (VR'06)
1087-8270/06 $20.00 © 2006 IEEE

15. “coupling attribute” mappings between data type properties of individuals e.g. the scaleX of a transformation applied to an individual (representing a cell) is related to the size property of the cell with an equation. These mappings can also be defined between data type properties of the graphics ontologies e.g. Transformation1.rotationX = 2 * Transformation2.translationY.

The above mappings relate classes, properties and individuals of ontologies, which may be very different in what they exactly describe. Thus, our model supports generality; domain knowledge expressed with ontologies in different domains and applications can uniformly be represented and integrated with 3D knowledge representations. In figure 4, we present a simple example of mappings between a scene and concepts from the Gene Ontology (www.geneontology.org), a biological domain ontology about biological processes, cellular components and molecular functions.

It is clear, that it is impossible to express arbitrary mappings directly with OWL object properties relating the involved classes. We overcome these difficulties using an intermediate ontology that defines the classes that represent each of the above mappings. We introduce fifteen OWL classes that have object properties relating them to the classes, the properties and the instances of the involved ontologies. For example, we introduce the object property “fromClass” having as range the built-in class “rdf:Class” (meaning that the property values can be one of the ids of the classes from the graphics ontologies). In the same manner, we introduce the “toClass” property referencing the ids of the classes from the other domain specific ontologies. In the last type of mappings, we describe the equation relating the involved properties with a MathML expression which is an W3C standard XML-based language describing mathematical notations (www.w3.org/Math/).

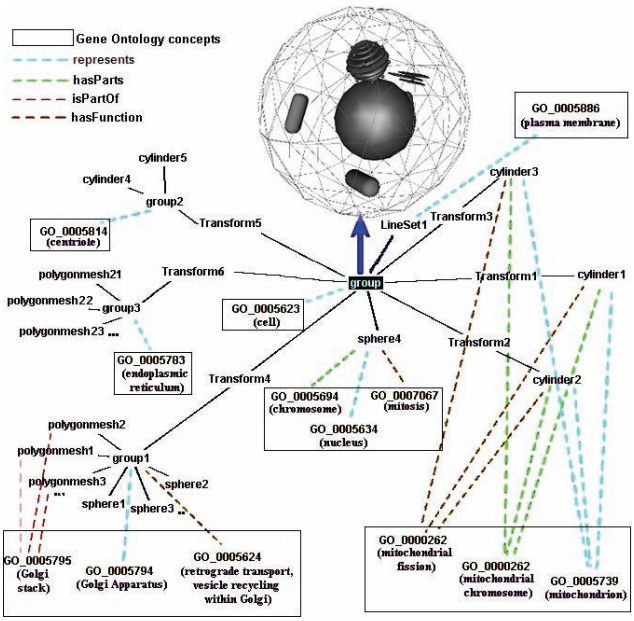

Figure 4: Example of incorporation of domain knowledge (Gene Ontology) into a scene showing the components of a cell

5 THE I3DVP PLATFORM

The I3DVP platform proves the viability of our approach to combine the knowledge-based and visualization technologies we describe in this paper. It is based on a distributed client-server architecture, which is depicted in figure 5. The tools we used to implement the architecture were Java (Java Development Kit 1.5 version), the Protégé API (version 3.1) and the J-Algernon inference engine (version 5.0.1). For the purposes of the I3DVP platform, we have developed a special file format (called I3DVPproto format) which can be used to define the individuals from the graphics ontologies and those of specific domain ontologies to be created and the semantic mappings between them. We also permit the definition of multiple sets of individuals (called visualization sets) that belong to more than one scene simultaneously (e.g. we can create two scenes – one with animation of an object and another without its animation). We can also define references to external datasets to be imported by the visualization engine. The format of these I3DVPproto files can be considered as a visualization & semantic language, which describes the generation of 3D scenes with their specific domain semantics.

The I3DVP platform offers another possibility for users to dynamically describe the scenes with a high level programming language such as Java based on the graphics ontologies. When the

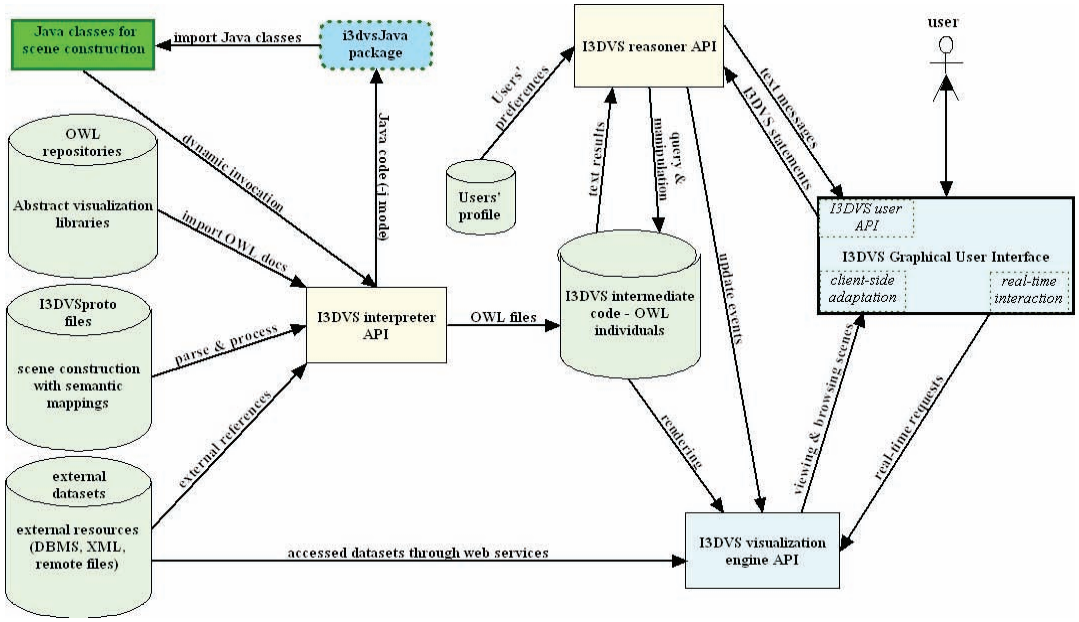

Figure 5: Architecture of the I3DVP system

I3DVP interpreter is executed in "ontologyToJava mode", it produces Java code packed in a package called I3DVPJava, which can be imported by the designers. In figure 6 we present an example of Java programming to describe a scene based on the graphics ontologies. The I3DVP interpreter API parses and reads the I3DVPproto files, imports the graphics and domain specific ontologies and dynamically invokes the user defined Java classes (if specified) to produce the OWL individuals describing the content of the scenes. Finally, the interpreter processes any references to external resources so that they can properly be processed by the visualization engines. The produced OWL files (I3DVP intermediate code) are accessed by the client visualization engines that are responsible for rendering the scene with OpenGL. The visualization engine accesses the middleware services that import any large datasets from external resources if specified. The users interact with the scene through the I3DVP GUI interface and send their manipulation and querying requests to the I3DVP reasoner in the server. The server, with the help of an inference engine (Algernon) makes all the necessary changes to the OWL individuals of the scenes and sends update events to the visualization engine.

```
private Shape CreateF16Nose() {
    IndexedFaceSet i = new IndexedFaceSet("F16nose");
    i.p1_coordIndex = "24,0,3,-1, 4,0,24,-1, ...";
    i.p1_creaseAngle = (float) 0.5;
    i.p1_normalIndex = "16,0,3,-1, 4,0,16,-1, ...";
    Coordinate c = new Coordinate("F16Nosecoordinates");
    i.p1_hasCoordinate = c;
    c.p1_point = "-0.32 0.36 -4.91, -0.38 0.43 -4.21 ...";
    Normal n = new Normal("F16Normal");
    i.p1_hasNormal = n;
    n.p1_vector = "-0.68 0.714 -0.166, ...";
    Material m = new Material();
    m.p1_diffuseColor = ".25 .25 .25";
    i.p1_hasMaterial = m;
    return i;
} ...
```

Figure 6: constructing a scene with the I3DVPJava package

The user visualization engine is also capable to create the hierarchical X3D scene graph automatically so that designers do not need to create it independently. This is achieved in two passes. During the first pass, for every OWL individual, we appropriately create new scene graph nodes. For example, for every individual representing a fact about a geometry object, we create a Shape node and a geometry object node under it. We also use a hash table to store the id (unique name) of the OWL individual and an id (a pointer) for the corresponding parent node we created. External datasets, defined with individuals of the OWL classes responsible for referencing external resources of low level graphical representations (e.g. in different graphics databases), are also imported in this step accordingly. During the second pass, for every object property in the OWL graph, we create the corresponding parent-child relationships between the involved nodes (generated during the first pass) in the scene graph. The hash table is used to efficiently find the parent nodes of the scene graph in constant time corresponding to the individuals referenced by object properties e.g. for every object property of type "hasMaterial" between a geometry object and a material, we put the appearance node for the material under the corresponding shape node. Of course, when we detect that an OWL individual is referenced by more than one object properties (indicating e.g. a shared material, transformation etc), we appropriately generate copies of the corresponding nodes in the scene graph ensuring its integrity and validity.

6 Knowledge Driven Visualization – Performing Reasoning on the Content of the Scenes

In this section we present implementation details of our platform, as well as the use of the system for knowledge driven 3D visualization functions in advanced applications taking advantage of knowledge-based technologies. The ontological description of scenes offers us a strategic advantage beyond their semantic enrichment. Ontologies provide us with the model theoretic and axiomatic basis underlying network structured knowledge bases (KB) where reasoning by inference can be performed on their content. This is possible as we have the reasonably compact syntax of OWL with well defined semantics for knowledge representation giving us sufficient expressive power to represent knowledge. Making inference can be very useful for a variety of reasons: a) create new content or update the scenes based on their existing content or their incorporated domain knowledge, b) query the scene combining both their content and their domain knowledge and synchronize possible redundant data between them c) personalize the scenes by formalizing users' preferences about their content and d) create the scene entirely based on the semantic instances of a specific domain (semantic driven visualization).

OWL does not provide reasoning capabilities by itself. What we additionally need is a KB Management System (KBMS) with an inference engine and a formal language to express definite (Horn) clauses, which are very useful and common in all knowledge systems. The intended meaning of a rule can be read as follows: whenever the conditions specified in the antecedent hold, then the conditions specified in the consequent must also hold. The Semantic Web community gradually understands the importance of defining such clauses in a formal way within its framework so that they are syntactically and semantically accessible and executable in the same way by all software agents. The latest and most well-known attempt to achieve this is the Semantic Web Rule Language (SWRL - http://www.daml.org/2003/11/swrl/) that extends the OWL syntax to describe such definite clauses. To support the functionality of knowledge bases, we used the Stanford's Protégé Java API (http://protege.stanford.edu/) for their management together with the Algernon-J (http://algernon-j.sourceforge.net/) rule based inference engine that works with frame-based KBMS (such as Protégé) and supports both backward (if-needed) and forward chaining (if added) rules.

Algernon embodies the Access-Limited Logic (ALL) language, which formalizes the access limitations inherent in a network structured knowledge base. An access limited logic retrieves all assertions reachable by following an available access path; the complexity of inference is thus independent of the size of the knowledge base and depends only on its local connectivity polynomially [20]. The construction and manipulation of the scenes is achieved with a series of TELL statements through the Algernon API into the Protégé core for inserting new facts about the content of the scenes and their querying operations on them are implemented with ASK statements for answering queries. The statements are processed on a client-server architecture where clients send the querying, construction and manipulation statements, the server receives them and manages the OWL graph of the scenes with their semantic mappings accordingly. We stress that the knowledge-based techniques presented in this section can easily be integrated with appropriate e.g. natural user interfaces so that end-users do not need to write ALL statements or any line of code. For example, in order to describe the simple model of a planet (figure 7), three TELL statements need to be executed by the server – we express the statements with a logic programming-style syntax, which permits the mathematical formulation of these statements and is used by most knowledge representation languages like ALL:

TELL: instance(Sphere, planet1) ^ name(planet1, "Planet")
TELL: instance(Torus, rings1) ^ name(rings1, "Rings") ^
scale(rings1, "2 0.1 2") ^ radius(rings1, "0.4")
TELL: instance(Group, group1) ^ name(group1, "PlanetSystem")^
onCommands(group1, planet1) ^ onCommands(group1, rings1)

Proceedings of the IEEE Virtual Reality Conference (VR'06)
1087-8270/06 $20.00 © 2006 IEEE

Figure 7: A 3D model of a planet

As large scenes usually need to be created collaboratively by distant users, it is essential for designers to apply rules in the form of a trigger "Condition=>Update" in order to synchronize and facilitate their design ensuring its validity. We present a simple example of a forward chaining rule to demonstrate their usefulness and application – we stress again that these rules can easily be defined with a user interface (users do not need to write any code).

TELL: defrule add-to-planetary-system (
(∀ g)(instance(Group, g)=> onCommands(PlanetarySystem, g)))

which means that every group in the scene (instance of the Group class) should belong to a group named PlanetarySystem. This rule acts as a trigger – every time that a user creates an instance of Group, it is inferred by the OWL reasoner that it is automatically added to this group.

The developed model and mechanisms can also be used to support semantic personalization of scenes. Traditional 3D languages or platforms do not provide any mechanisms to support this; every scene is the same for all designers and users. Our approach for the introduction of semantic personalization [10] involves the creation of contextualized ontologies that synchronize the user and designer preferences with the content of the scenes. These contexts contain facts about new individuals to be inserted to the scene when accessed (representing facts about preferred objects such as backgrounds, materials, preferred navigation types etc) and rules which not only define the application of these facts into the scene but also describe transformations of existing objects in the scene (e.g. scale all objects in the scene by 2x). More specifically, user profiles have been implemented in "personal" OWL files that import the graphics ontologies so that users define their preferred individuals to be inserted into the scene (e.g. a preferred background) and rules that define the "personalized" manipulation of the existing content of the accessed scenes. For example, a user defines a rule saying that for every scene he accesses, a specific material from his user profile should be applied on every shape in the scene with no assigned material.

TELL: defrule add-personal-material (
(∀ s)(instance(Shape, s) => hasMaterial(s,PreferredMaterial1)))

which when executed by the server, it will automatically generate a material for every shape with no assigned material (the rule will unsuccessfully applied to shapes that already have been assigned with an existing material). We stress that, as these changes should not be applied for all users, the server keeps a copy of the personalized scene especially for this user.

The uniform integration of domain semantics with the 3D content opens up many more capabilities for 3D intelligent environments. Supposing that we have already specific domain semantic information in our disposal, we would like to have the chance to automatically generate new content or synchronize the existing one based on these semantics. For example, we may have an ontology about chemical molecules and we may need to visualize all instances of this ontology describing the atoms with their bonds. The solution we offer is to define the appropriate logical clauses, playing the role of "semantic information visualization rules" that perform the task of converting the semantic information into 3D objects, transformations and animation. These "visualization rules", when executed, will produce e.g. a sphere for each atom (with the necessary semantic mappings between them), lines representing bonds between the spheres, different colors for each type of atom and finally the necessary transformations (e.g. translations) for the produced objects (place the spheres of the atoms based on the crystallographic information of each molecule):

TELL: defrule construct-atom-spheres (
(∀ a) (instance(Atom, a) ^ name(a, n) =>
instance(Sphere, s) ^ name(s, n) ^
instance(Equivalence, e) ^
fromInstance(e, a) ^ toInstance(e, s)))

The above statement means that for every instance of atom, we generate a sphere into the scene and create a semantic mapping of type "equivalence" between them.

TELL: defrule translate-atom-spheres (
(∀ e) instance(Equivalence, e) ^ fromInstance(e, a)
^ toInstance(e, s) ^ crystallographicCoords(a, c) =>
instance(Transform, t) ^
translation(t, c) ^ onCommands(t, s))

The above statement means that we get the crystallographic coordinates of every atom and we generate a transformation to every corresponding sphere. If we need to convert the crystallographic coordinates, we may execute additional rules or call external functions to perform any complex mathematical operation. The Algernon inference engine allows us to call external (static or non-static) functions in Java or define Lisp expressions that are incorporated into the statements. We alternatively allow designers to write fragments of Java code (figure 6) in order to create the individuals of the scene and then send TELL and ASK statements together with this code. In the same way, we can define rules to visualize the bonds between the atoms and assign the necessary materials for each type of atom. In each step of the execution of the rules, new content is generated and then the existing content together with the defined semantic mappings are used to describe new content again until the scene is complete. The semantic driven visualization we have applied to visualize chemical molecules is depicted in figure 8. We note that the above sets of rules may not be unique. The users may want to define their own (personalized) rules.

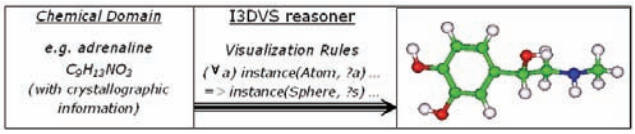

Figure 8: semantic driven visualization of chemical molecules

We can use the same approach to define rules that allow visualization-aided decision making i.e. based on the graphics content of the scenes and their applied semantics, we can make inference to automatically take decisions (e.g. a medical diagnosis based on the nature of the materials assigned to the objects representing organs, tissues etc.). Based on these decisions, we can further manipulate the content (e.g. assign an interpolator to an object in order to produce an animation based on one of its attributes – if a tissue is damaged, then create a rotational animation on it to observe it by all angles). In general, we consider that the integration of rules executed by inference engines on the ontological description of the scenes with their semantic mappings offers vast possibilities to be explored in many different fields of applications.

Until now, we have presented how TELL statements are used for the manipulation and personalization of the scenes. Querying is performed by sending ASK statements to the server. These statements may only refer to the content of the scenes or involve their semantics as well. Our approach incorporates the description logics of OWL into the scenes and enriches them with native semantic mappings with domain knowledge so that we not only simplify but also enhance the querying operations. The queries are applied on the high level semantic representation of the scenes making it easy to answer queries like "find all shapes with their materials in a group" or "find the position of an object" independent of the intermediate transformations of the groups that it belongs to. A

straightforward example is the following:

ASK: instance(Shape, s) ^ onCommands(PlanetarySystem, s) ^
onCommands(t, s) ^ instance(Transformation, t)

which asks to find all shapes belonging to the group "Planetary-System" with all their transformations. The onCommands property is transitive so that an OWL reasoner will directly find all the planets and satellites of the system without needing to write complex queries to traverse the group hierarchy. Then, we can also define a "coupling attribute" mapping between e.g. the position attribute of a physics simulation ontology and the multiplication of the returned transformation matrices to face synchronization issues. In general, the results of queries can be used to update the value of redundant attributes of instances in different ontologies, by defining "coupling attribute" mappings between them and the results of the queries together with a specified mathematical relation to automatically achieve the necessary synchronization.

In order to also take into account the semantic enrichment of the scenes for the queries, we simply involve the corresponding semantic mappings. To answer queries such as "what is this component which I clicked upon with my mouse?" or "what is the functionality of this component?", we search for "represents" or "equivalence" mappings and "hasFunction" mappings for the object, if they exist. We can also answer interactive queries like "focus on the component X of the scene" – actually, we create a viewpoint (camera) based on its translation and center of the object. Such interactive queries involve the execution of TELL statements that add information (cameras, navigation type) in the scene.

7 Conclusions and Future Work

In this paper, we presented a generalized model and methodology to incorporate domain knowledge into 3D scenes in web based environments. Our methodology raises the description of graphics and virtual reality content to the ontologies layer of the semantic web offering new interesting techniques to manipulate and query the scenes. Their enrichment with domain knowledge is accomplished by utilizing generalized easy-to-understand ontological mapping primitives between the graphics ontologies we have developed and any other domain-specific ontologies. We also presented the implementation of a platform based on our model and methodology. Finally, we showed new intriguing possibilities for knowledge-based semantic driven information visualization with embedded decision making which can be useful for scientific simulation environments or many other visualization applications.

Our future work will be focused on the following issues regarding the further development of I3DVP applications:

- Further study of the graphics ontologies to evaluate their completeness for the description of scenes' content and further study of the type of semantic mappings and their ontological model based on the analogical reasoning AI literature.
- Further study performance and synchronization issues (as those described in [17]) of the I3DVP visualization engines in order to fully support real-time manipulation of the scenes and implementation of an industrial strength system.
- Automatic extraction of semantics from the content of the scenes.

References

[1] Chaomei C., Top 10 Unsolved Information Visualization Problems, IEEE Computer Graphics and Applications, 25(4), pp. 12-16, July-August 2005

[2] Tangelder J. W. H., Veltkamp R. C., A Survey of Content Based 3D Shape Retrieval Methods, Proceedings of Shape Modeling International, Genova, Italy, p. 145-156, June 2004

[3] Resnik P., Semantic Similarity in a Taxonomy: An Information-Based Measure and its Application to Problems of Ambiguity in Natural Language, Journal of Artificial Intelligence Research, vol. 11, pp. 95-130, 1999.

[4] Cera C. D., Regli W. C., Braude I., Shapirstein Y., Foster C. V., A Collaborative 3D Environment for Authoring Design Semantics, IEEE Computer Graphics and Applications, Special Issue on "Computer-Aided Design", 22(3), p.42–55, 2002.

[5] Razdan A., Rowe J., Tocheri M., Sweitzer W., Adding Semantics to 3D Digital Libraries, 5th International Conference on Digital Libraries (ICDL 2002), Singapore, pp 419-420, December 2002

[6] Peters S., Shrobe H, Using semantic networks for knowledge representation in an intelligent environment, 1st Annual IEEE International Conference on Pervasive Computing and Communications, Ft. Worth, TX, USA, 2003

[7] Luck, M. and R. Aylett, Applying Artificial Intelligence to Virtual Reality: Intelligent Virtual Environments, Applied Artificial Intelligence, 14(1), p. 3-32, 2000.

[8] Cavazza M., Palmer I., High level interpretation in dynamic virtual environments, Applied Artificial Intelligence, 14(1), p.125-144, 2000

[9] Latoschik M. E., Biermann P., Wachsmuth I., High-level Semantics Representation for Intelligent Simulative Environments, Proceedings of the IEEE VR2005, Bonn, Germany, pp. 283-284, March 2005

[10] Tsinaraki C., Christodoulakis S., A Multimedia User Preference Model that Supports Semantics and its Application to MPEG 7/21", In the proceedings of the Multimedia Modeling 2006 Conference (MMM 2006), January 2006, Beijing, China

[11] Tsinaraki C., Polydoros P., Christodoulakis S., Interoperability support for Ontology-based Video Retrieval Applications, In the Proceedings of Conference on Image and Video Retrieval (CIVR) 2004, Dublin/Ireland, July 2004

[12] Elliott C., An Embedded Modelling Language Approach to Interactive 3D and Multimedia Animation, IEEE Transactions on Software Engineering, 25(3), p. 291 – 308, May/June 1999.

[13] Brutzman D., Blais C., Harney, J., Visualizing Information Using SVG and X3D, Chapter 3, Springer-Verlag, November 2004.

[14] Halabala, P., Semantic Metadata Creation, Proceedings of 7th Central European Seminar on Computer Graphics CESCG 2003, Bratislava Comenius University, pp. 15-25, 2003

[15] Clay S.R., Wilhelms J., Put: Language-Based Interactive Manipulation of Objects, IEEE Computer Graphics and Applications, vol. 16, No. 2, pp. 31-39, March 1996.

[16] Latoschik M. E., Schilling, M., Incorporating VR Databases into AI Knowledge Representations: A Framework for Intelligent Graphics Applications, Proceedings of the Sixth IASTED International Conference on Computer Graphics and Imaging, Honolulu, Hawaii, pp. 79-84, August 2003.

[17] Schilling M., Latoschik M. E., Heumer G., Automatic Data Exchange and Synchronization for Knowledge-Based Intelligent Virtual Environments, Proceedings of the IEEE VR2005, Bonn, Germany, pp. 43-50, March 2005.

[18] Gutierrez M., Vexo F., Thalmann D., Semantics-based representation of Virtual Environments, International Journal of Computer Applications in Technology, vol.23, pp. 229-238, 2005

[19] Gutierrez M., Thalmann D., Vexo F., Semantic Virtual Environments with Adaptive Multimodal Interfaces, 11th International Conference on Multimedia Modelling, MMM2005, Melbourne, Australia, p. 277-283, January 2005.

[20] Crawford J., Access-Limited Logic - A Language for Knowledge Representation, Technical Report AI90-141, Ph.D. Thesis, Department of Computer Science, University of Texas at Austin, 1990.

[21] Martinez J. M., Koenen R., Pereira F., MPEG-7: The Generic Multimedia Content Description Interface, Part 1, IEEE MultiMedia, vol. 9, pp. 78-87, 2002

1087-8270/06 $20.00 © 2006 IEEE