




Inference-based creation of synthetic 3D content with ontologies

Krzysztof Walczak¹ · Jakub Flotyński¹ 

Received: 28 February 2018 / Revised: 26 September 2018 / Accepted: 17 October 2018 /
Published online: 31 October 2018
© The Author(s) 2018

Abstract

Creation of synthetic 3D content is typically a complex task, covering different geometrical, structural, spatial and presentational elements. The available approaches enable 3D content creation by programming or visual modeling of its elements. This demands expertise in computer science from content authors, limits the possibilities of using domain knowledge, and requires to determine all content details within the content creation process. In this paper, we propose a new method of 3D content creation, which is based on ontologies. Ontologies are the foundation of the semantic web, enabling the use of knowledge in various domains. In the proposed method, the use of ontologies facilitates 3D content creation by domain experts without skills in programming and computer graphics. In addition, due to the use of ontologies, the method enables automated reasoning, liberating the authors from determining many elements of the created content, which can be inferred by the content generation algorithm on the basis of explicitly specified content elements. The method has been implemented and evaluated. It simplifies 3D content creation in comparison to the available approaches by reducing the number of activities that must be completed by the content authors. Hence, the proposed method can increase the use of 3D content in different application domains.

Keywords 3D web · Virtual and augmented reality · Semantic 3D · Ontologies · Semantic web

1 Introduction

Synthetic 3D content is the key part of virtual reality (VR) and augmented reality (AR) applications, which become increasingly popular in various domains, such as prototyping, medicine, education, training, tourism, scientific visualization, entertainment and cultural

✉ Krzysztof Walczak
walczak@kti.ue.poznan.pl

Jakub Flotyński
flotynski@kti.ue.poznan.pl

¹ Poznań University of Economics and Business, Niepodległości 10, 61-875 Poznań, Poland

heritage. Creation of synthetic 3D content is a complex task, which typically includes design of geometrical, structural, spatial and presentational elements. Despite numerous works in this domain, difficulty associated with creation of synthetic 3D content is still one of the main obstacles for efficient development of VR/AR applications in many domains.

Creation of 3D content is possible with a number of well-established technologies, which may be divided into three main groups. The first group encompasses libraries (e.g., Direct3D [39], Java3D [40] and Away3D [10]), which enable programming of 3D content with imperative languages, such as C++, Java and ActionScript. In imperative programming, stress is put on specification of the steps to be made to achieve desirable results. The second group includes declarative 3D content formats, such as VRML [50], X3D [51] and XML3D [18]. In declarative programming, stress is put on the desirable results instead of the steps necessary to achieve the results. The third group comprises 3D modeling tools, which combine visual 3D modeling with programming of content. Examples of such tools are Blender [28], Maya [9] and 3ds Max [8]. Such tools usually make the process easier and more efficient.

The potential of VR/AR applications can be fully exploited only if synthetic 3D content is created with efficient and flexible methods, which conform to the recent trends in the development of information systems. The main motivations for our approach (depicted in Fig. 1) are as follows.

1. 3D content creation should be available to domain experts, who are usually not IT-specialists. This requires 3D content creation methods that are understandable to non-programmers by focusing on specification of high-level domain-specific goals to be achieved instead of specific low-level technical steps to be completed to achieve the goals. Low-level details related to 3D graphics could be created by graphic designers in a reusable form and further linked to domain-specific concepts by developers. However, the available approaches either require the use of low-level concepts related to 3D graphics or restrict designers to use domain-specific tools that cannot be easily adapted to other domains.
2. 3D content is typically composed of various geometrical, structural, spatial and presentational elements. To liberate 3D content authors from determining all such elements,

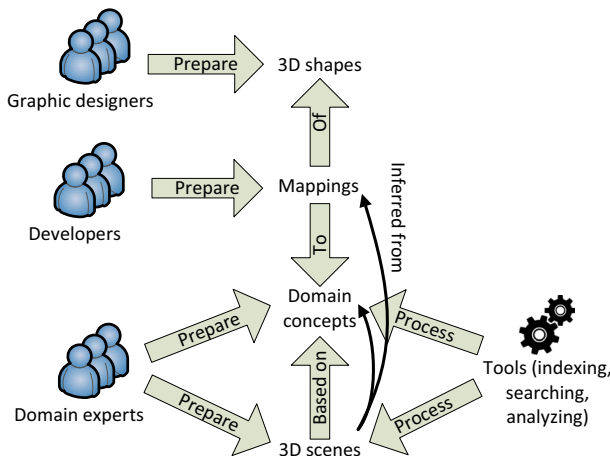


Fig. 1 Motivations for inference-based creation of 3D content with ontologies

the content creation process should enable *inference*, which is the process of specifying tacit (not explicitly specified) content elements on the basis of explicitly specified elements. Inference is done automatically—by software referred to as a *reasoning engine*. Possible use of inference in 3D modeling covers a variety of cases. For instance, every 3D object that has a material assigned can be inferred to belong to the objects-with-material class, even if not explicitly specified as such, and further visualized in a 3D scene, in which only the objects of this class are presented. In a 3D zoo, the specification of subclasses of some species by a user may be sufficient to infer which animals belong to these subclasses, using the recursive subclass of relation between classes. In a 3D building, different 3D models for external and internal doors could be selected by a reasoning engine on the basis of inferring in what a type of wall (external or internal) the doors are. However, the available approaches to 3D modeling do not benefit from inference in such a way and require specification of all 3D content elements to be presented or specification of algorithms that create the elements.

3. 3D content accessible on the web should be suitable for exploration with queries (e.g., by search engines and analytical tools) for indexing, searching and analyzing of content. This is not enabled by the available 3D content formats and programming languages, which are inappropriate for query-based exploration or enable only structural queries (e.g., with XQuery) without the possibility of exploring domain-specific semantic properties of the content.

Potential applications of the aforementioned concepts are very broad and encompass a variety of domains. For instance, in medicine, 3D models of a human body could be composed by physicians. 3D models of damaged organs would be automatically selected by a reasoning engine using inference on the rules and facts describing diseases affecting the organs. The resulting body models could be searched and explored with semantic queries by students for educational purposes. In cultural heritage, 3D exhibitions could be composed by museum curators using inference on rules and facts describing artifacts, with regards to desirable intersections of artifacts' types (armors, clothes, coins, dishes, etc.), ages, provenance and ethnic groups. Generated 3D exhibitions would be suitable for indexing and searching on the web in response to visitors' queries. In e-commerce, 3D shops could be designed by marketing specialists using inference to select desirable classes of products (e.g., food, sport accessories and electronics) to be placed on desirable placeholders. Customers could effectively navigate in such shops using queries about desirable products, while marketing specialists would have feedback from processing semantic logs of customers' choices.

However, the aforementioned requirements are not satisfied by the available approaches to 3D content creation. Further progress in 3D modeling is possible thanks to the use of semantics [48, 57], broadly understood as concepts describing different content properties at different levels of abstraction. Currently, the main approach to describe the semantics of content is the semantic web, which has been derived from knowledge and metadata representation. The semantic web aims at the evolutionary development of the current web towards a distributed database linking structured content described by ontologies [11]. Semantic ontology-based representation of content makes it intelligible to humans and interpretable to computers, achieving a new functionality of web applications, which can interpret the meaning of particular content elements as well as their relations using concepts (classes, objects and properties) specific to any domain. This can lead to much better methods of creating, indexing, searching, querying and analyzing of content. The use of the semantic web for describing 3D content recently gains increasing attention of the research

community in different applications such as 3D content retrieval [45], design of industrial spaces [43], archaeology [19] and molecular visualization [46, 47].

The main contribution of this paper is the *Method of Inference-based 3D Content Creation* (MICC). The method is based on ontologies, which provides several important advantages in comparison to the previous approaches to 3D content creation. First, it enables content representation at an arbitrary level of abstraction, in particular specific to an application or domain, at the same time hiding technical details specific to 3D graphics. This has been achieved by general domain-agnostic mapping between domain-specific concepts and graphics-specific concepts. Hence, MICC outperforms previous solutions of semantic 3D modeling, which are strictly related to particular application domains. Moreover, 3D content can be created with focus on the desirable elements instead of the steps that are necessary to create the elements. The aforementioned features can foster creation of 3D content by authors without knowledge of programming and graphics design, in particular domain experts who are not IT-specialists. Second, the use of ontologies enables reasoning, liberating content authors from specifying elements that could be inferred from the explicitly specified elements. This can reduce users' effort and time spent on content preparation. Third, declarative 3D content representation enabled by the method can be semantically explored with queries (e.g., expressed in the SPARQL language [55]), which makes the method compliant with the current trends in the web development.

Taking into account the features of the proposed method, it may be successfully used to create several categories of 3D scenes:

1. Scenes that are comprised of multiple objects belonging to common classes in a particular domain. For example, in a VR museum exhibition, there are collections of objects with common presentational properties—coins, armors, weapons, tools, etc.
2. Scenes that conform to some constraints that can be satisfied in various ways, e.g., generate exemplary virtual chambers with furniture, paintings and decorations belonging to general classes without specifying exact classes of the objects.
3. Scenes that satisfy constraints, whose verification requires implementation of complex (e.g., recursive) algorithms. For instance, in a scene with museum paintings highlight only paintings related to the successors to a particular sovereign.

The remainder of this paper is structured as follows. Section 2 provides an overview of the current state of the art in semantic representation of 3D content. In Section 3, our new method is presented along with an example of inference-based 3D content creation. In Section 4, the architecture and implementation of the scene design environment based on the method is outlined. In Section 5, the evaluation of the method and the environment is presented, followed by discussion in Section 6. Finally, Section 7 concludes the paper and indicates possible directions of future research.

2 Related works

In [26], a comprehensive review of approaches that use ontologies to support 3D representation and 3D modeling process has been presented. The main semantic web standards used to create ontology-based 3D content representations are the Resource Description Framework (RDF) [53], the RDF Schema (RDFS) [54] and the Web Ontology Language (OWL) [52]. RDF is a data model based on triples (subject - property - object). RDFS and OWL are languages providing classes and properties that enable comprehensive description of objects. These standards enable design of ontologies, which are specifications of

conceptualization [29] for a domain. Ontologies may describe arbitrary objects as well as classes and properties of objects.

Following [26], *3D content representation* in this paper is defined as a document that specifies different 3D content components and properties, in particular geometrical, structural, spatial and presentational. Semantic 3D content representations are instances of *3D content ontologies*, which are schemes specifying different possible classes and properties of 3D content components.

As explained in the paper, the essential activity of ontology-based 3D content creation, which is also the main advantage over other modeling approaches, is the inference of tacit knowledge. Knowledge about 3D content, which may be expressed at the low or high abstraction levels, can be used to infer implicit content properties from explicit properties. Therefore, content authors are liberated from specifying all content properties.

A number of works have been devoted to ontology-based representation of 3D content including a variety of geometrical, structural, spatial and presentational elements. However, the majority of the approaches are used like typical 3D content formats, in which all 3D content elements have to be explicitly specified by the authors, without capabilities of knowledge inference. The available methods are summarized in Table 1. Four methods address the low (graphics-specific) abstraction level, while six methods address a high (general or domain-specific) abstraction level. Three of those methods may be used with different domain ontologies.

The method proposed in [12, 16, 17, 34, 42] enables content creation at both the low and a high abstraction levels. At the particular levels, different 3D content ontologies connected by mapping are used. Low-level ontologies may be created by graphic designers, while high-level ontologies may be created by domain experts. Mapping of low- to high-level ontologies adds interpretation to graphical components and properties. For instance, a mapping links particular meshes in 3D scenes to particular pieces of furniture in a virtual shop to enable intuitive search and presentation to customers [16]. The approach also enables combination of primitive actions (e.g., move, turn, rotate, etc.) to composite behavior intelligible to end users without the knowledge of computer graphics. The method requires authors to specify all components and properties of 3D content.

Table 1 Comparison of methods of ontology-based 3D content creation

Approach	Level of abstraction		Inference
	Low (3D graphics)	High (application domain)	
Troyer et al. [12, 16, 17, 34, 42]	✓	general	–
Gutiérrez et al. [30, 31]	✓	humanoids	–
Kalogerakis et al. [32]	✓	–	✓
Spagnuolo et al. [6, 7, 44]	–	humanoids	✓
Floriani et al. [15, 41]	✓	–	–
Kapahnke et al. [33]	–	general	–
Albrecht et al. [5]	–	interior design	✓
Latoschik et al. [20, 35, 56]	–	general	–
Drap et al. [19]	–	archaeology	✓
Trellet et al. [46, 47]	–	molecules	✓
Perez-Gallardo et al. [43]	✓	–	✓

The method proposed in [30, 31] also enables 3D content creation at both low and high abstraction levels. Ontologies used in the method include graphical 3D content components (e.g., shapes and textures) and properties (e.g., coordinates and indicies) as well as high-level domain-specific components (e.g., body parts) and properties (e.g., joint attributes, descriptors of articulation levels, 3D animations of face and body, and behavior controllers). The method does not use knowledge inference.

The method proposed in [32] enables 3D content creation at the low abstraction level. The used ontology provides components and properties that are equivalents of X3D nodes and attributes, e.g., textures, dimensions, coordinates and LODs. The method enables inference on semantic rules. For instance, if an individual is of the atom class (body), generate a sphere to represent it in an ontology for chemical compounds (head). Thus, final 3D content is based on both the explicit and implicit (inferred) knowledge. However, the method does not enable mapping between high- and low-level concepts, so it is unsuitable for modeling 3D content by domain experts.

Another method of creating 3D humanoids has been proposed in [6, 7, 44]. After automatic segmentation of 3D models, the distinguished body parts are semantically annotated. Two modes of annotation have been developed. Automatic annotation is completed by software considering topological relations between content elements (e.g., orientation, size, adjacency and overlapping). Manual annotation is completed by a user equipped with a graphical tool. Although the method benefits from inference in annotating of 3D models, it is limited to virtual humans.

The method proposed in [15, 41] enables creation of non-manifold 3D shapes using low-level properties. Once 3D shapes are segmented, graphical properties are mapped to a shape ontology and form an ontology-based low-level shape representation. The ontology specifies diverse geometrical properties of shapes: non-manifold singularities (e.g., isolated points and curves), one-dimensional parts, connected elements, maximal connected elements, the number of vertices, the number of non-manifold vertices, the number of edges, the number of non-manifold edges and the number of connected elements. It permits representation of such objects as a spider-web, an umbrella with wires and a cone touching a plane at a single point.

The tool described in [33] leverages semantic concepts, services and hybrid automata to describe behavior of objects in 3D simulations. The tool has a client-server architecture. The client is based on a 3D browser, e.g., for XML3D, while the server is built of several services enabling 3D content creation. A graphical module maintains and renders 3D scene graphs. A scene module manages global scene ontologies, which represent the created simulations. A verification module checks spatial and temporal requirements against properties of content elements. An agent module manages intelligent avatars, e.g., their perception of the scene. The user interface enables communication with web-based and immersive virtual reality platforms. Ontology-based content representations are encoded in XML using the RDFa and OWL standards, and linked to 3D content encoded in XML3D.

In [5], a method of 3D content creation based on point clouds has been proposed. At the first stage of the method, an input point cloud is analyzed to discover planar patches, their properties (e.g., locations) and relations. Then an OWL reasoner processes a domain ontology including conceptual elements that potentially match the analyzed patches. Next, matching elements are selected and configured to build a high-level representation in the interior design domain. Created representations are ontology-based equivalents to the input point clouds. The method uses knowledge inference to analyze 3D scenes, e.g., context of use of 3D objects and connections between objects.

In [20, 35, 56], a general-purpose tool and method of 3D content creation has been described. The method is based on actors and entities, which represent 3D content at a high level, are described by shared state variables and are subject to events. In particular, the approach can be used in game design.

In [19], a method and software for representing underwater archaeological objects in 3D have been presented. In the approach, a Java-based application generates an ontology representing objects. Further, queries encoded in the SWRL language [49] can be used to select objects to build a 3D visualization.

In [46, 47], an approach to semantic representation of 3D molecular models has been proposed. The approach combines different input (e.g., interaction using different haptic and motion tracking devices) and output (e.g., presentation in 2D and 3D) modalities to enable presentation and interaction suitable for particular types of content and tasks to be done. The approach uses inference on ontologies describing the content and SPARQL queries sent by users to visualize selected objects.

In [43], a system for 3D recognition of industrial spaces has been presented. The method used in the system recognizes objects in point clouds presenting interiors of factories. The recognized objects, their properties and relations, which are specific to 3D graphics, are further semantically represented using ontologies. On this basis, topological relations between objects are inferred.

The use of knowledge inference in the available approaches is limited only to selected classes of semantic objects and properties at a particular level of abstraction. General methods of 3D modeling that enable the use of knowledge inference in different application domains at different abstraction levels are still lacking.

3 Method of inference-based 3D content creation

The main contribution of this paper is the *Method of Inference-based 3D Content Creation* (MICC). The method is a part of the approach to *Semantic Modeling of Interactive 3D Content* (SEMIC) outlined in [21, 22]. In those works, we focused on an approach to separation of concerns between different 3D modeling users, who may be graphics designers, developers and domain experts. The distinction of different modeling steps to be completed by people with diverse skills better organizes the 3D modeling process and reduces the role of IT-specialists. This paper extends our previous articles by detailed presentation of rules and an algorithm that enable link between arbitrary domain and 3D graphics. MICC offers important advantages over the previous approaches to 3D modeling, including ontology-based solutions. It enables semantic representation of 3D content in different application domains—using concepts intelligible to domain experts. Such concepts, which provide high-level content representation, are mapped to concepts specific to 3D graphics, which provide low-level content representation. In the 3D content creation process, authors focus on desirable properties of 3D content. Due to applying knowledge inference, all content properties that have not been explicitly specified, but are logical consequences of explicit properties, are inferred by a standard reasoning engine. This reduces users' effort in 3D modeling as proven in Section 5.

The general assumption of MICC is the reuse of a *library of 3D content components* for creating different *semantic 3D scenes* using inference on *domain ontologies* (Fig. 2). The main activities of the method are: *library development*, which is completed once for a particular domain or application, and *3D scene creation*, which can be completed multiple times with a given library. Within the activities, different steps are distinguished. The steps

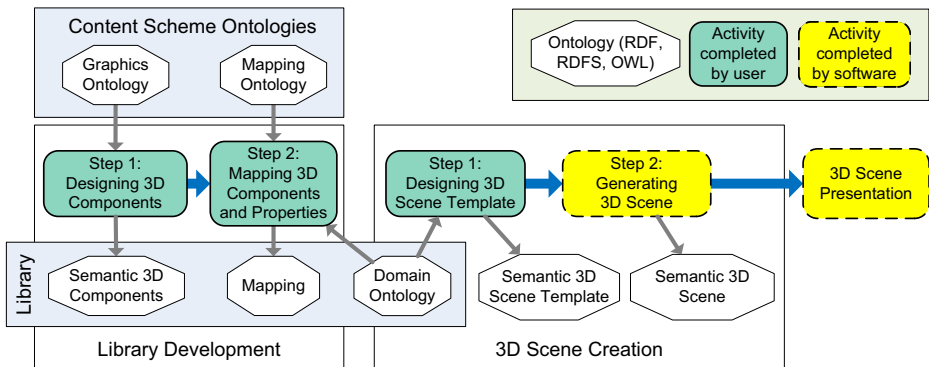


Fig. 2 Creation of 3D content with the MICC method

are completed in the declarative fashion—with the specification of desirable 3D content properties instead of a sequence of actions that must be executed to set the properties (like in imperative programming). The steps use different ontologies to produce 3D content. *Library development* and *3D scene creation* may be completed using the Scene Design Environment (SDE, cf. Section 4) or individual tools for 3D modeling (e.g., Blender [28] or 3ds Max [8]) and semantic modeling (e.g., Protégé [3]). Within the steps, responsibilities are divided between different users and software.

The following sections describe the main MICC activities and steps, along with an example, in which different parts of a 3D scene in a virtual museum of agriculture are created by different users and software.

3.1 Development of a library of 3D content components

A library consists of independent 3D content components, which may be further reused to compose multiple 3D scenes. The components are encoded using the semantic web standards—RDF, RDFS and OWL. Using the standards permits inference and semantic exploration of 3D components when building 3D scenes, e.g., select a texture presenting brick to cover a wall, select an animation of jumping to be attached to a character. A library is created once for a particular domain ontology. For instance, individual libraries are required to enable modeling of machines, architecture, interiors or museums. A domain ontology may be provided by experts in a particular field of knowledge or, if it does not exist yet, it may be designed by IT-professionals in collaboration with such experts specifically for the use in MICC.

3.1.1 Step 1: designing 3D components

This step provides 3D components, which will enable use of domain classes and properties (specified in a *domain ontology*) for 3D modeling in further steps of MICC. 3D components are OWL classes of: 3D shapes (e.g., meshes), structural objects (e.g., hierarchies of meshes), appearance descriptors (e.g., textures, materials) and animation descriptors (e.g., interpolators, sensors) with specific values of OWL datatype and object properties. 3D components are designed with regards to the intended manner in which domain classes and properties should be presented. Modeling of 3D components typically requires the use of specific hardware or software tools, e.g., creating 3D meshes requires a 3D scanner or a

3D modeling tool, while drawing textures requires a 2D graphical editor. 3D components are classes of the basic low-level elements of semantic 3D scenes, which are described by semantic datatype and object properties. Instances of 3D components represent neither particular coherent 3D scenes nor particular compositions of 3D objects, but they can be flexibly reused to compose various complex 3D objects and scenes. 3D components are subclasses of application-independent graphical classes and properties (referred to as *3D properties*) specified in the *Graphics Ontology*, which are counterparts to widely-used concepts of 3D graphics, present in numerous 3D formats. 3D components need to be designed with regards to the domain classes and properties that will be used in *3D scene creation*. For instance, different 3D meshes represent specific car models, while textures determine their appearance, etc. This step is typically completed by a graphic designer.

In the example of museum exhibition, in Step 1 of the *library development*, a graphic designer creates several virtual objects that represent real museum artifacts. First, the graphic designer uses specific modeling tools to create 3D components—a 3D scanner to capture the geometry of: a granary, a woman statuette, a sower, a smoker, a ring, a seal and a badge, and a 2D graphical tool—to prepare textures for the 3D models (Fig. 3). The components are encoded as an OWL ontology that conforms to the *Graphics Ontology*. The components may be created using SDE (cf. Section 4), semantic modeling tools (e.g., Protégé) or specifically designed plugins for graphical editors (e.g., Blender [28] or 3ds Max [8]). For every 3D model and every texture in the example, 3D components described by properties have been generated (Listing 1). The ontology is encoded in RDF, RDFS and OWL using the syntax of the RDF Turtle format [4]. Some components and properties that are not crucial for the example have been skipped. In the example, the woman statuette is to be used by domain experts in three different forms: as clay and glassy virtual artifacts (Lines 3-25) as well as a painted virtual artifact with the inherent texture mapping (27-34). The other 3D models are to be used in single forms (36-44).

3.1.2 Step 2: mapping 3D components and properties to domain concepts

Mapping 3D components and properties to domain classes and properties enables presentation of 3D scenes described in a particular application domain, e.g., linking a tail animation to a 3D animal model, inclusion of multiple meshes representing engine parts

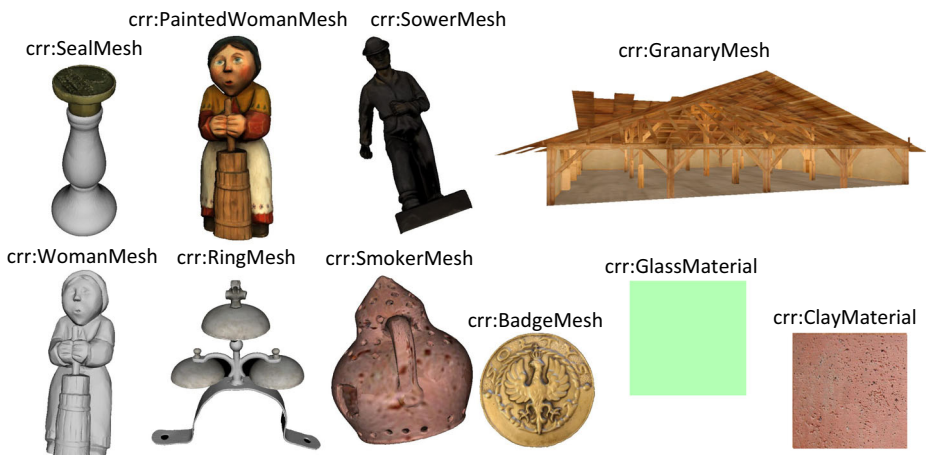


Fig. 3 Visualization of example 3D components

```

1  Prefixes: Graphical Ontology (go), Semantic 3D Components (sc)
2
3  sc:WomanMesh rdf:type owl:Class ;
4    rdfs:subClassOf go:Mesh3D ,
5    [ rdf:type owl:Restriction ;
6      owl:onProperty go:meshData ;
7      owl:hasValue "woman.obj" ] .
8
9  sc:ClayMaterial rdf:type owl:Class ;
10   rdfs:subClassOf go:TextureMaterial ,
11   [ rdf:type owl:Restriction ;
12     owl:onProperty go:texture ;
13     owl:hasValue "clay.png" ] ,
14   [ rdf:type owl:Restriction ;
15     owl:onProperty go:transparency ;
16     owl:hasValue 0 ] .
17
18  sc:GlassMaterial rdf:type owl:Class ;
19   rdfs:subClassOf go:ColorMaterial ,
20   [ rdf:type owl:Restriction ;
21     owl:onProperty go:color ;
22     owl:hasValue "green" ] ,
23   [ rdf:type owl:Restriction ;
24     owl:onProperty go:transparency ;
25     owl:hasValue 0.7 ] .
26
27  sc:PaintedWomanMash rdf:type owl:Class ;
28   rdfs:subClassOf sc:WomanMesh ,
29   [ rdf:type owl:Restriction ;
30     owl:onProperty go:texture ;
31     owl:hasValue "statueTexture.png" ] ,
32   [ rdf:type owl:Restriction ;
33     owl:onProperty go:textureCoordinates ;
34     owl:hasValue "..." ] .
35
36  {sc:GranaryMesh,..., sc:BadgeMesh}
37   rdf:type owl:Class ;
38   rdfs:subClassOf go:Mesh3D ,
39   [ rdf:type owl:Restriction ;
40     owl:onProperty go:meshData ;
41     owl:hasValue "..." ] ,
42   [ rdf:type owl:Restriction ;
43     owl:onProperty go:texture ;
44     owl:hasValue "..." ] .

```

Listing 1 The ontology including example 3D components

within a complex structural 3D engine model. Mapping is performed once for a particular set of 3D components and graphical properties and a domain ontology. Mapping enables reuse of components and properties for presenting various 3D scenes, which conform to the domain ontology. Mapping may cover only some parts of a domain ontology in practical applications.

This step is typically completed by a developer with skills in semantic modeling, using SDE or a semantic editor (e.g., Protégé [3]). Mapping is performed according to *mapping patterns*, which are sequences of activities frequently repeated by modeling users to provide a link of 3D components and properties to domain classes and properties. To create a mapping, developers use *complex mapping patterns*, which are sequences of *simple mapping patterns*.

Simple mapping patterns Simple Mapping Patterns (Fig. 4) are basic activities completed by users while mapping 3D components and properties to domain classes and properties.

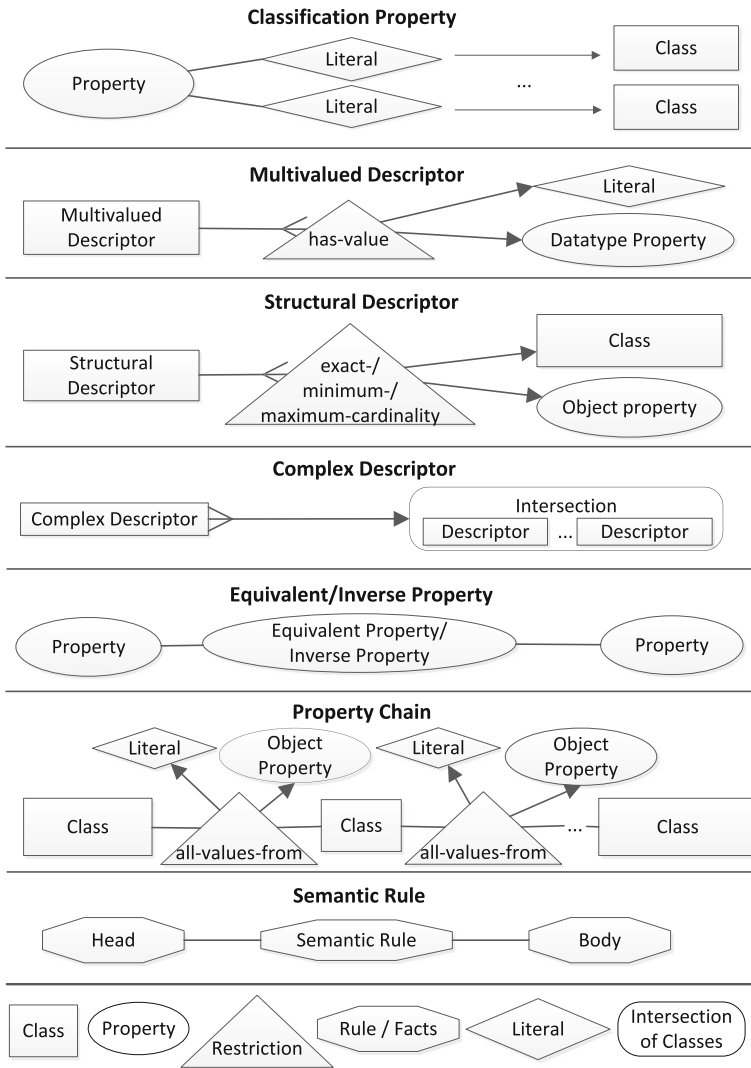


Fig. 4 Simple mapping patterns

Generic classes and properties for accomplishing simple mapping patterns are contained in the *Mapping Ontology*.

The *classification property* pattern enables reflection of an OWL property P whose values are literals by a set of OWL classes. For every distinct classification value of P, an individual class is created and it is specified as an equivalent to a has-value OWL restriction on P with the required P value. Consequently, every OWL individual that has a particular classification value of P assigned, belongs to one of the created classes. For instance, objects made of metal, wood and plastic may belong to MetalObject, WoodenObject and PlasticObject classes (*isMadeOf*=<metal, wood, plastic>). Every created class may be further described by different properties, using other simple mapping patterns.

The *multivalued descriptor* pattern enables specification of the desirable OWL datatype properties for OWL individuals of a common OWL class. To make a class a multivalued descriptor, it needs to be specified as a subclass of the intersection of OWL has-value restrictions. Every OWL has-value restriction indicates a required value for one of the desirable datatype properties. For instance, every gold object is yellow and reflects light—one restriction specifies color, while the other specifies shininess.

The *structural descriptor* pattern enables creation of a complex structure of OWL classes that are linked by OWL object properties. To make a class a structural descriptor, the class needs to be specified as a subclass of the intersection of minimum-cardinality, exact-cardinality or maximum-cardinality restrictions. For instance, every physical object is made of a material, every animated object has an animation assigned. The linked class may also be a structural descriptor, thus creating a complex structure of linked classes. In addition, structural descriptors can be extended with OWL datatype properties by applying the multivalued descriptor mapping pattern.

The *complex descriptor* pattern enables specification of the desirable OWL datatype properties and OWL object properties of OWL individuals based on multiple OWL classes that are assigned to the individuals. In contrast to the previous patterns, which enable mapping *one domain class-to-many 3D properties*, complex descriptors enable mapping *many domain classes-to-many 3D properties*, making desirable property values conditional on classes assigned to the individual and classes not assigned to the individual. For instance, the color of a wooden object is brown, while the color of a waxen object is, e.g., pink, but it also depends on the object temperature. For every distinguishable combination of classes, a separate class (a complex descriptor) is created and it is specified as an equivalent to the intersection of the classes that are required and the complements of the classes that are not required. Due to the use of the complements of classes, the close world assumption has to be made to enable the use of this pattern. Every complex descriptor can be further extended with datatype properties and object properties by applying the multivalued descriptor and the structural descriptor mapping patterns.

The *equivalent property* pattern enables specification of an OWL property as an equivalent to another OWL property. Equivalent properties are processed in the same manner. A number of properties may be specified as equivalent. For instance, the *includes*, *contains* and *incorporates* properties may be equivalents even if defined in different ontologies. The *inverse property* simple mapping pattern enables the specification of an inverse property [2]. For instance, *includes* and *is included in* are inverse properties.

The *property chain* pattern enables connection between OWL individuals of two different OWL classes by linking the classes via mediating classes that are subclasses of all-values-from restrictions. Every mediator class is specified as a subclass of an all-values-from restriction that indicates the next class in the chain using an OWL object property. The linked class is also a subclass of an all-values-from restriction. For instance, in a traditional interior, a room may include only objects made of wood. Indicating an object as included in the room determines the object to be wooden.

The *semantic rule* pattern (RL) is the most general pattern. It overtakes the previous patterns in terms of expressiveness. This pattern is used to create logical implications that determine selected OWL properties of OWL individuals (in the head of the rule) on the basis of other properties of individuals (in the body of the rule). For instance, every object *standing* on a table has the *y* coordinate calculated on the basis of the *y* coordinate of the table and the *heights* of both the objects.

Complex mapping patterns The use of particular simple mapping patterns for linking particular domain concepts to 3D components and properties is determined by complex mapping patterns as depicted in Fig. 5.

Presentable objects (POs) are OWL individuals that represent 3D models in the scene. This mapping pattern has the following steps. For each domain class C whose individuals need to have independent representations in the scene, create a separate *Presentable Object Class* (POC) and specify it as a superclass of C. Next, specify OWL datatype and object 3D properties that are inherent to the individuals of POC. Use the structural descriptor pattern to link POC with its descriptors using object properties (Fig. 5-I), e.g., incorporating subobjects in a hierarchy, indicating materials and animations. Further, use the multivalued descriptor pattern to assign the required datatype properties, e.g., colors, coordinates and dimensions. In such a way, every domain individual of C in the scene will be described by all the properties assigned to POC. If C occurs in a hierarchy of domain classes, its ascendant domain classes should be described first. Additional presentational effects that are not inherent to the ascendant classes, should be described directly for C.

Descriptive Classes (DCs) are OWL classes assigned to POs in order to determine their 3D properties. This mapping pattern has the following steps. Each domain class C that may be assigned to POs to specify their presentational properties, but does not identify independent entities to be presented, specify as a DC and apply one of the following rules (Fig. 5II).

1. If C exclusively determines different 3D properties that are not collectively determined by other domain classes (mapping *one domain class-to-many 3D properties*), use the structural and multivalued descriptor patterns to determine desirable datatype and object properties of C individuals.

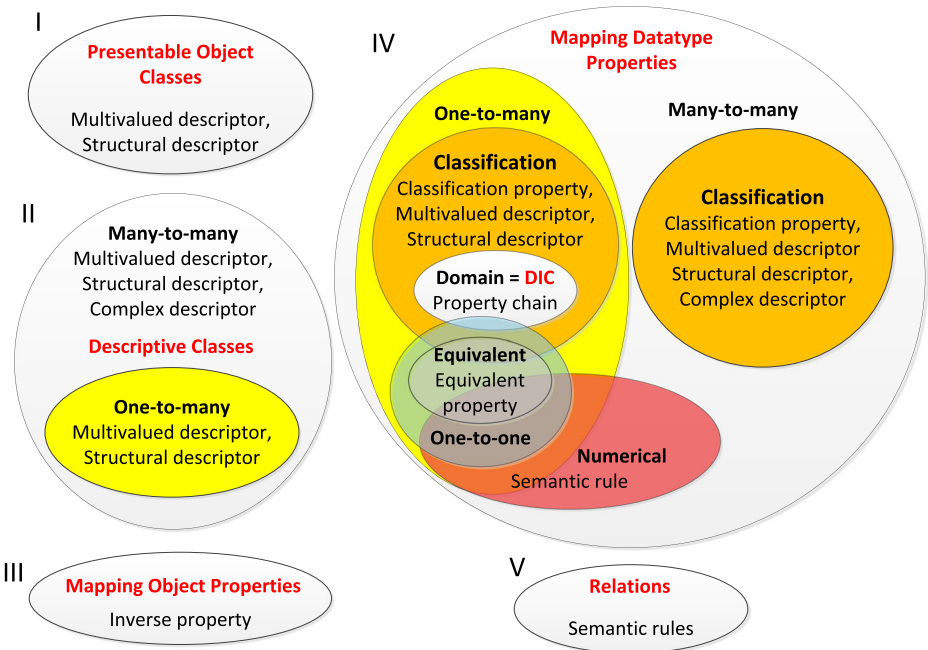


Fig. 5 Complex mapping patterns

2. If C collectively determines different 3D properties with other domain classes (mapping *many domain classes-to-many 3D properties*), first, use the complex descriptor pattern to create a separate DC for every considered combination of the classes that are assigned and the classes that are not assigned to the individual. Second, use the structural and multivalued descriptor patterns for each of these DCs to specify their desirable datatype and object properties.

Like in the case of mapping hierarchies of POCs, mapping hierarchies of DCs should be completed first for the ascendant DCs and second for the descendant DCs.

Descriptive Individuals (DIs) are OWL individuals assigned to POs using *Mapping Object Properties* (MOPs) in order to determine their presentational effects. This mapping pattern has the following steps. For each domain object property that links DIs to POs or to other DIs, use the inverse property pattern to create its inverse object property, if it does not exist (Fig. 5III). Maintaining bidirectional links (object properties and their inverse object properties) between individuals is necessary to enable application of the property chain mapping pattern (which uses object properties to link DIs to DIs and to POs).

Mapping Datatype Properties (MDPs) are OWL datatype properties assigned to POs in order to determine their presentational effects. This mapping pattern has the following steps. To each domain OWL datatype property (DP) that needs to have presentational effects on the described POs, apply one of the following rules (Fig. 5IV).

1. If DP exclusively determines particular 3D properties, regardless of other datatype properties, DCs and DIs assigned to the described PO (mapping *one domain property-to-many 3D properties*), apply one of the following rules.
 - (a) If the domain of DP is a POC, apply one of the following rules.
 - i. If DP is equivalent to a 3D datatype property, indicate this fact using the equivalent property pattern (mapping *one domain property-to-one 3D property*).
 - ii. If DP is a classification property (its domain is a finite OWL class of literals), use the following combination of mapping patterns. First, use the classification property pattern to create a separate DC for each distinct value of DP. Second, extend the DCs by applying structural descriptors and multivalued descriptors to assign required 3D object and datatype properties.
 - (b) If the domain of DP is a class of DIs and its range is a set of classification data, apply the following combination of mapping patterns. First, use the classification property pattern to create a separate DC for each distinct DP value. Second, use the property chain pattern to specify the path between the DI class and the described POC. Third, extend the DCs using the structural and multivalued descriptor patterns to specify the desirable 3D object and datatype properties of their POs.
2. If the range of DP is a set of numerical data, for which a formula can be specified to determine the values of the linked 3D properties on the basis of DP, use the semantic rule pattern.
3. If DP collectively determines different 3D properties in combination with other DPs, DCs and DIs assigned to the POs (mapping *many domain properties-to-many 3D properties*), perform the following steps. First, use the classification property pattern to specify a separate DC for every distinct value of every considered DP. Second, use the structural and multivalued descriptor patterns to specify object and datatype properties

of the DCs. Third, use the complex descriptor pattern to create a new DC that is the intersection of the appropriate DCs.

Like in the case of hierarchies of POCs and DCs, mapping domain datatype properties should start with ascendant properties and only these domain subproperties that introduce additional presentational effects (in comparison to their superproperties) should be additionally described.

Each domain object property whose domain and range are POCs, specify as an RL and complete the semantic rule pattern (Fig. 5V), determining the values of desirable 3D properties of the participants of the RL on the basis of domain properties of other participants.

Each domain class C that has no independent representation in the created content, and for which there are at least two domain object properties that link C with some POCs, specify as an RL and create a rule describing dependencies between particular properties of the POCs.

Example of mapping In Step 2 of the *library development* in the example, a developer or a technician creates a mapping (Fig. 6 and Listing 2—the RDF Turtle and Prolog-like syntax) including semantic statements that link domain classes and properties to 3D components (created in Step 1) and properties. The `dso:Woman` (3-4) artifact is a PO class and a subclass of the `sc:WomanMesh`, so it inherits the geometrical properties specified in Step 1. Every instance of `dso:Woman` can be made of clay or glass (as indicated by

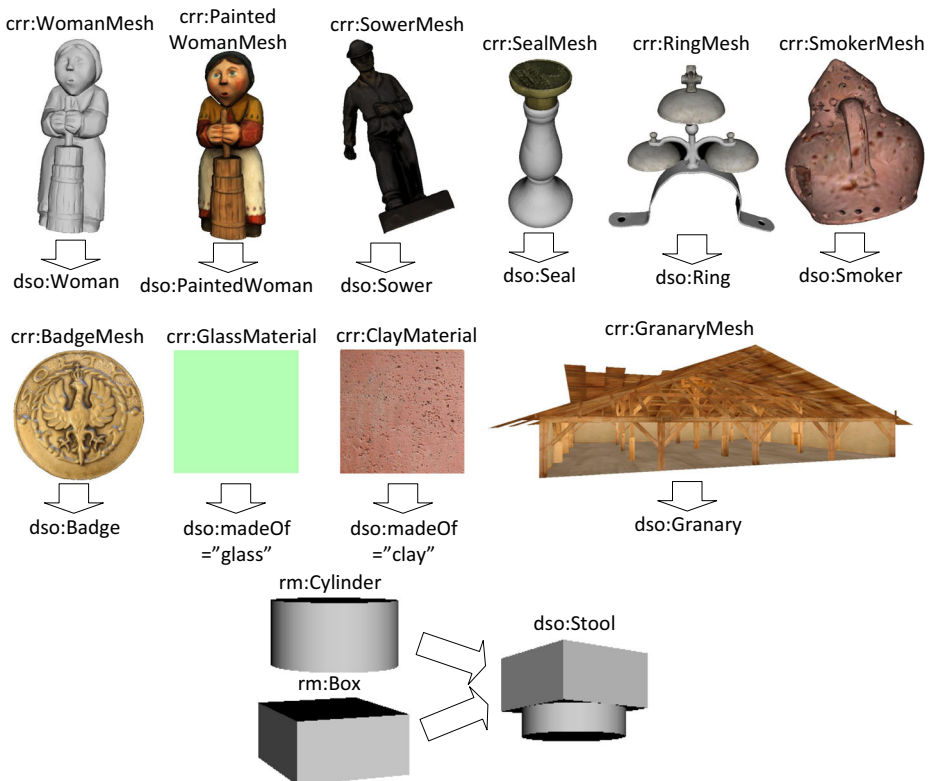


Fig. 6 An example of mapping 3D components and properties to domain concepts


```

1  Prefixes: Graphical Ontology (go), Semantic 3D Components (sc), Mapping Ontology (mo),
    Mapping (mp), Domain Ontology (do)
2
3  do:Woman rdfs:subClassOf
4    mo:PresentableObject , sc:WomanMesh.
5
6  {mp:ClayObject,mp:GlassyObject}
7    rdf:type owl:Class ;
8    rdfs:subClassOf mo:DescriptiveClass ;
9    owl:equivalentClass
10   [ rdf:type owl:Restriction ;
11     owl:onProperty do:madeOf ;
12     owl:hasValue "{clay,glass}" ] ,
13   [ rdf:type owl:Restriction ;
14     owl:onProperty go:material ;
15     owl:onClass {sc:ClayMaterial,sc:GlassMaterial} ;
16     owl:qualifiedCardinality 1 ].
17
18 do:PaintedWoman rdfs:subClassOf
19   mo:PresentableObject , sc:PaintedWomanMesh.
20
21 {do:Granary,...,do:Badge} rdfs:subClassOf mo:PresentableObject ,
22   {sc:GranaryMesh, ..., sc:BadgeMesh}.
23
24 {mp:Box,mp:Cylinder} rdfs:subClassOf {go:Box,go:Cylinder} ,
25 [ rdf:type owl:Restriction ;
26   owl:onProperty go:size ;
27   owl:onClass {mp:BoxSize,mp:CylinderSize} ;
28   owl:qualifiedCardinality 1 ] ,
29 [ rdf:type owl:Restriction ;
30   owl:onProperty go:position ;
31   owl:onClass {mp:BoxPos,mp:CylinderPos} ;
32   owl:qualifiedCardinality 1 ].
33
34 do:Stand rdfs:subClassOf mo:PresentableObject , go:StructuralComponent ,
35 [ rdf:type owl:Restriction ;
36   owl:onProperty go:includes ;
37   owl:onClass go:Cylinder ;
38   owl:qualifiedCardinality 1 ] ,
39 [ rdf:type owl:Restriction ;
40   owl:onProperty go:includes ;
41   owl:onClass go:Box ;
42   owl:qualifiedCardinality 1 ].
43
44 {mp:BoxSize,mp:CylinderSize} rdf:type owl:Class ;
45 rdfs:subClassOf go:Vector ,
46 [ rdf:type owl:Restriction ;
47   owl:onProperty x ;
48   owl:hasValue "..." ] ,
49 [ rdf:type owl:Restriction ;
50   owl:onProperty y ;
51   owl:hasValue "..." ] ,
52 [ rdf:type owl:Restriction ;
53   owl:onProperty z ;
54   owl:hasValue "..." ].
55
56 {mp:BoxPos,mp:CylinderPos} rdf:type owl:Class ; ...
57
58 do:incorporates rdfs:subPropertyOf mo:BinaryRelation ;
59   owl:equivalentProperty go:includes.
60
61 do:standsOn rdfs:subPropertyOf mo:BinaryRelation.
62 {go:x(APos, AX),go:z(APos, AZ)} :-
63   do:standsOn(A, B) ,
64   go:position(A, APos) ,
65   go:position(B, BPos) ,
66   {go:x(BPos, BX),go:z(BPos, BZ)} ,
67   {AX=BX,AZ=BZ} ,
68 go:y(APos, AY) :-
69   do:standsOn(A, B) ,
70   go:position(A, APos) ,
71   go:position(B, BPos) ,
72   go:y(BPos, BY) ,
73   go:size(B, BSize) ,
74   go:sy(BSize, BSY) ,
75   go:size(A, ASize) ,
76   go:sy(ASize, ASY) ,
77   AY = BY + (ASY + BSY)/2.

```

Listing 2 The ontology mapping 3D components and properties to domain concepts

`do:madeOf`), thus having an appropriate material assigned using proper DCs (6-16). In contrast to clay and glassy artifacts, every `do:PaintedWoman` PO has a texture assigned, as indicated by its superclass (18). Mapping other classes of the domain ontology to PO classes is performed in a similar fashion (21-22). Moreover, two basic shapes (`mp:Box` and `mp:Cylinder`) are created (24-32) and assembled into the `do:Stand` PO class (34-42). Every `mp:Box` and `mp:Cylinder` included in a `do:Stand` has dimensions and position (44-56). The `do:incorporates` RL is an equivalent to the `co:includes` (58-59), while the `do:standsOn` RL determines the *x*, *y* and *z* coordinates of an individual by semantic rules (61-77). Once mapping is completed, the library is ready to be used by domain experts for modeling various 3D scenes in a particular field of application.

3.2 3D scene creation

3D scene creation is completed at a high level of abstraction that is determined by the domain ontology used. This activity can be performed multiple times, when a new 3D scene is required for a VR/AR application, using a particular library of 3D components, which are mapped to the domain ontology.

3.2.1 Step 1: designing a 3D scene template

In this step, a domain expert uses domain classes and properties to build a *3D scene template*, which will be further automatically expanded into a 3D scene. The domain expert does not need to directly use 3D components and properties as they are hidden behind the mapping. The resulting 3D scene template is an instance of the domain ontology. The template consists of facts (statements) and rules (implications), which declaratively represent 3D content at the high (domain-specific) level of abstraction. Both facts and rules are on domain individuals (instances of domain classes) and domain properties. In general, this step is independent of the previous steps, and a 3D scene template could be created before 3D components and the mapping were created, e.g., when a domain expert designs an accurate digital equivalent to a known real space, using well-defined domain concepts. However, when designing non-existing virtual 3D objects or scenes (e.g., a planned virtual museum exhibition or 3D city model), 3D components and a mapping are useful to enable preview of the results while modeling.

In Step 1 of the *3D scene creation* in the example, a domain expert creates a 3D scene template (Listing 3) including individuals of domain classes, which are described by domain properties. Both the classes and the properties are already mapped to 3D components and properties. The domain expert creates several artifacts (3) and three woman statuettes (5-9). The first two statuettes are made of different materials (clay and glass), while the third statuette is inherently covered by a texture (as specified in the mapping). Furthermore, eight stands are created (11). Their *x*, *y* and *z* coordinates are declaratively specified by rules (13-21). In Line 20, the cut-off and the negation-as-failure operators are used to determine a stand, for which no *x* coordinate has been specified. Every artifact is placed on a stand (23-27). Every artifact that is not on any stand, is placed on a stand on which there is no artifact yet. It is only important to deploy all the artifacts on some stands, but it is not important, on which stand a particular artifact is placed. An artifact is placed on a stand by calculating the *X*, *Y* and *Z* coordinates of the artifact with respect to the coordinates of the stand, according to the rule in the mapping. Finally, all the artifacts and stands are incorporated in the granary (29-32).

```

1  Prefixes: Domain Ontology (do), Semantic 3D Scene Template (st)
2
3  {st:granary,...,st:badge} rdf:type {do:Granary,...,do:Badge}.
4
5  {st:clayWoman,st:glassyWoman}
6    rdf:type do:Woman ;
7    do:madeOf "{clay,glass}".
8
9  st:paintedException rdf:type do:PaintedException.
10
11 {st:stand1,...,st:stand8} rdf:type do:Stand.
12
13 st:standPositions(Index, N) :-
14   Index<N, do:Stand(S), st:noPosition(S),
15   X is Index div 4, assert(do:x(S, X)),
16   Z is Index mod 4, assert(do:z(S, Z)),
17   assert(do:y(S, 0)),
18   NewIndex is Index+1,
19   st:standPositions(NewIndex, N).
20 st:noPosition(S) :- do:x(S, X), !, false.
21 st:noPosition(S).
22
23 st:standsOn(A, B) :- do:Artifact(A), do:Stand(B), st:notOnOthers(A), st:
   nothingOnIt(B).
24 st:notOnOthers(A) :- st:standsOn(A, B), !, fail().
25 st:notOnOthers(A).
26 st:nothingOnIt(B) :- st:standsOn(A, B), !, fail().
27 st:nothingOnIt(B).
28
29 do:incorporates(X, Y) :-
30   do:Granary(X),
31   (do:Artifact(Y) ; do:Stand(Y)),
32   X!=Y.

```

Listing 3 A semantic 3D scene template

3.2.2 Step 2: generating the 3D scene

Step 1 of the *library development* and Step 1 of the *3D scene creation* have encompassed all the activities that must be resolved by a human: the specification of how domain concepts should be represented by 3D components and the specification of what domain individuals should form the modeled 3D scene. So far, 3D components and properties have been linked to domain classes and properties. However, since the 3D scene template consists of domain individuals (instances of domain classes), instances of 3D components must be generated and directly linked to the domain individuals in the template.

To enable presentation of domain individuals, the 3D scene template is expanded according to the mapping in the *3D scene generation* step. It is a process of automatic creation of individuals of 3D components, linking them to domain individuals by object properties, and describing them by datatype properties. Scene generation is based on the inference of tacit knowledge. Scene generation is performed in the following three stages. At Stage I, hidden facts are inferred from RLs. The facts may be related to classes, properties and individuals at different (low and high) levels of abstraction. At Stage II, DCs are collected for the particular POs to determine object and datatype properties of the POs. Finally, at Stage III, individuals are generated and linked to POs by object properties. Also, datatype properties are assigned to POs and to new individuals generated to determine their presentational effects.

In these three stages, the semantic 3D scene template is expanded to a semantic 3D scene. In contrast to the template, the scene specifies all individuals of 3D components that are necessary to present the domain individuals in 3D.

Scene generation algorithm. The input data of the scene generation algorithm are the ontology of 3D components, the domain ontology, and the mapping between these two (see Fig. 2). The algorithm consists of the following stages.

- I. **Reasoning on RLs:** perform reasoning on RLs (created with the *semantic rule* mapping pattern).

Reasoning on RLs leads to the inference of DCs, DIs, MOPs and MDPs assigned to POs. The inferred facts will be used at the next stages.

- II. **Collecting DCs:**

- A. For all the domain properties used, determine their equivalent properties (created with the *equivalent property* mapping pattern).
- B. For all the domain object properties used, determine their inverse properties (created with the *inverse property* mapping pattern).
- C. For every statement on a domain object property P_1 [PO_1, P_1, PO_2], create the statement [PO_2, P_2, PO_1], where P_2 is the inverse property of P_1 .
- D. To every PO, assign DCs (created with the *multivalued descriptor* and *structural descriptor* mapping patterns) on the basis of the DIs directly or indirectly linked to the PO (by all-values-from restrictions created with the *property chain* mapping pattern).
- E. To every PO, assign DCs (created with the *classification property* mapping pattern) on the basis of the MPs of the PO.
- F. To every PO, assign DCs (created with the *complex descriptor* mapping pattern) on the basis the DCs already assigned in Steps II-D and II-E.

In the result of Stage II, appropriate DCs are assigned to particular POs. The DCs describe 3D object and datatype properties of the POs. Therefore, the DCs may be used to generate individuals of 3D components, link them to the POs by 3D object properties and set 3D datatype properties of the POs—to provide 3D representations of the POs.

- III. **Assigning properties:** for every PO, set variable I to the PO and:

- A. For every OWL cardinality restriction class with a constraint [*Property, Class, N*] that I belongs to:
 - 1) if there is no other cardinality restriction on *Property* and *Class* that has already been processed for I, create N individuals of *Class* and link them to I by *Property*;
 - 2) for every created individual I_2 of *Class*, set I to I_2 and go to Step III-A.
- B. For every has-value restriction [*Property, Value*] that I belongs to, add *Property* with *Value* to I.

At this stage, inconsistencies between different restrictions should be reported, e.g., multiple orientations specified for a PO. The resulting meta-scene represents individuals described by 3D datatype properties and related one to another by 3D object properties.

Example of 3D scene generation In Step 2 of the 3D scene creation in the example, the 3D scene template is expanded according to the mapping into a *semantic 3D scene*. The scene includes eight artifacts (Fig. 7). During the scene generation, individuals of 3D components are created and linked to the domain individuals of the 3D scene by 3D object properties. Next, 3D datatype properties of the generated individuals are properly set (Listing 4). All the artifacts are specified as individuals of $go:Mesh3D$ (3). For the clay and the glassy woman statuettes (5-7), appropriate individuals representing materials are generated (9-12).



Fig. 7 Visualization of the generated 3D scene

`st: paintedWoman` is created in a similar way (14). Next, every `do: Stand` is expanded to a `go: StructuralComponent` that includes a `go: Cylinder` and a `go: Box` with appropriate dimensions and relative positions (16–31). For every `do: Stand` and every `do: Artifact`, a position is determined (33–35). Finally, all the individuals (stands and artifacts) are included in the granary (37).

```

1  Prefixes: Graphical Ontology (go), Semantic 3D Scene (ss), Semantic 3D Scene Template (st)
2
3  {st:granary, ..., st:badge} rdf:type go:Mesh3D.
4
5  {st:clayWoman, st:glassyWoman}
6    rdf:type go:Mesh3D ;
7    go:material {ss:clayMaterial, ss:glassMaterial}.
8
9  {ss:clayMaterial, ss:glassMaterial}
10   rdf:type {sc:ClayMaterial, sc:GlassMaterial} ;
11   {go:texture, go:color} {"clay.png", "green"} ;
12   go:transparency {0, 0.7}.
13
14  st:paintedWoman rdf:type sc:PaintedWomanMash.
15
16  {st:stand1, ..., st:stand8}
17   rdf:type go:StructuralComponent ;
18   go:includes {ss:cylinder1, ..., ss:cylinder8} , {ss:box1, ..., ss:box8}.
19
20  {ss:cylinder1, ..., ss:cylinder8, ss:box1, ..., ss:box8}
21   rdf:type {go:Cylinder, go:Box} ;
22   go:size {ss:size1, ..., ss:size16} ;
23   go:position {ss:pos1, ..., ss:pos16}.
24
25  {ss:size1, ..., ss:size16}
26   rdf:type go:Vector ;
27   go:x "...";
28   go:y "...";
29   go:z "...".
30
31  {ss:pos1, ..., ss:pos32} rdf:type go:Vector ; ... .
32
33  {st:stand1, ..., st:stand8} go:position {ss:pos17, ..., ss:pos24}.
34
35  {st:clayWoman, ..., st:badge} go:position {ss:pos25, ..., ss:pos32}.
36
37  st:granary go:includes {st:stand1, ..., st:stand8} , {st:clayWoman, ..., st:
    badge}.

```

Listing 4 The textual representation of the generated 3D scene

In the next steps of MICC, the 3D scene is transformed to a presentable format by the environment and visualized. The transformation may be format-independent [23] or format-dependent [24].

4 Scene design environment

MICC is independent of any software and hardware architectures and can be implemented as standalone or network applications. In this paper, the Scene Design Environment (SDE), which is an implementation of MICC based on the service-oriented architecture (SOA), is presented. The use of SOA complies with the semantic web approach, in which ontologies are distributed and shared across the network. Furthermore, it enables development of flexible multi-user environments, which separate client-side graphical operations from computationally expensive server-side semantic processing. Also, access to 3D content in distributed mobile environments is possible through the services.

SDE consists of three modules: the *3D Component Design Module*, the *Mapping Module* and the *Template Design Module*. The modules are used, respectively, in the subsequent steps of MICC: *designing 3D components*, *mapping 3D components and properties to domain concepts* as well as *designing a 3D template* and *generating the 3D scene*. The implementation of the modules consists of different layers, which are sets of software components (Fig. 8). The environment comprises a *Client* and a *Server* in a RESTful SOA architecture.

4.1 Client

The Client is a Python application, which enables visual manipulation of 3D content in the Blender modeling tool. The Client transforms Blender-specific 3D components and properties, which are processable and presentable by the tool, to equivalent semantic 3D components and properties, which are processable by the reasoning engine in the Server, and vice versa. The multi-layered architecture of the Client conforms to the Model-View-Controller

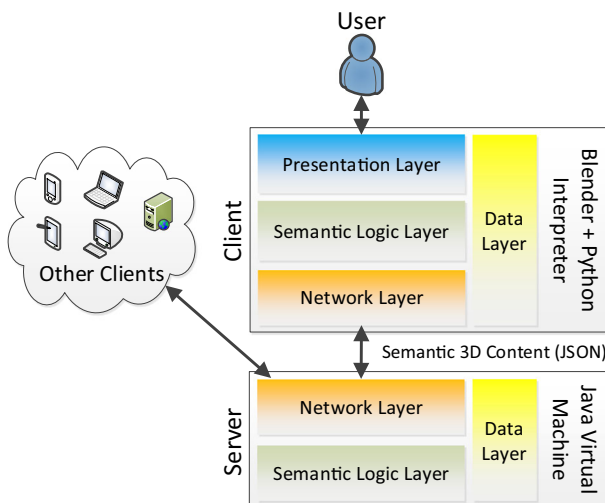


Fig. 8 The architecture of the Scene Design Environment for inference-based 3D content creation

(MVC) design pattern. It consists of four layers: the *Semantic Logic Layer* (controller), the *Presentation Layer* (view) as well as the *Network* and *Data Layers* (model).

The Client has been implemented using the Blender API [13]. Blender has been used because it is an advanced open-source environment with extensive documentation, tutorials and several versions available for different operating systems. However, the Client could be also implemented using other 3D modeling environments.

The *Presentation Layer* is responsible for handling requests from a user and presenting 3D content in Blender. It employs the Blender graphics engine, which allows a user to present and manipulate 3D content in the visual environment, as well as new GUI elements (panels), which extend the standard Blender GUI. The graphics engine enables access to a number of Blender tools, such as transformations, editors and scene graphs. It is accessible through specific classes and properties of the Blender API. The panels make the SDE modules available to users. The panels consist of buttons, menus and properties. Particular panels allow users to perform the subsequent modeling activities according to SEMIC. The GUI of the Client is depicted in Fig. 9.

The *Semantic Logic Layer* is responsible for processing user requests, and creating and managing 3D content in the Blender format that should be presented to a user or converted

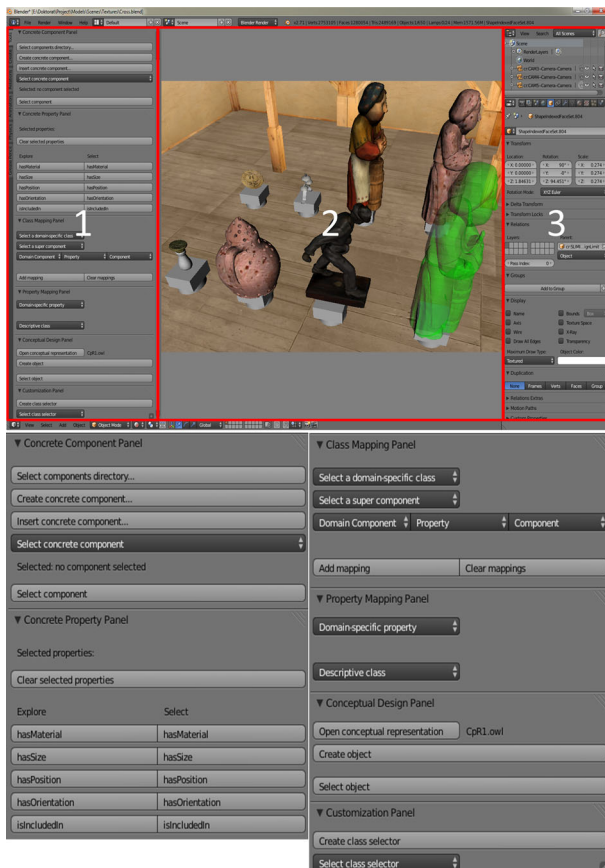


Fig. 9 The graphical user interface of SDE: the SDE Client panels (1-zoomed below) along with the *User Perspective* (2) and standard Blender tools (3)

to semantic 3D content and sent to the Server. Semantic 3D content is transferred between the Client and the Server using the *Network Layers*, which implement RESTful web services.

Semantic 3D content retrieved from the Server is converted to Blender-specific 3D content using a Blender mapping that is specified in the *Data Layer*. This mapping enables bi-directional transformation of both types of 3D content. The transformation of Blender-specific content to semantic content is performed every time the content is semantically manipulated (using an SDE panel), e.g., when a 3D object is extended or composed in Blender and it must be verified whether this modification influences other objects in the scene by explicit or implicit semantic links between the objects. The transformation of semantic 3D content to Blender-specific content is performed every time content is retrieved from the Server and presented in Blender.

4.2 Server

The Server is a Java application, which processes semantic 3D content according to users' actions. The Server processes, stores and provides semantic 3D content to clients. Since semantic 3D content is encoded using the semantic web standards (RDF, RDFS and OWL), the Server is independent of particular 3D formats, languages, browsers and modeling environments. Hence, it could be used with different software clients installed on various devices. The multi-layered architecture of the Server consists of three layers: the *Network Layer*, the *Semantic Logic Layer* and the *Data Layer* (Fig. 8). The *Network Layer* is responsible for communication with clients—retrieving and sending semantic 3D content. The *Semantic Logic Layer* is responsible for processing the semantic 3D content, in particular, inferring tacit knowledge (content components and properties), e.g., colors, textures, materials and animations. The *Data Layer* is responsible for storing semantic 3D content as well as the ontologies to which the content conforms.

5 Evaluation

The evaluation of MICC has been done using the implemented SDE. The evaluation covers use cases in modeling different 3D scenes by a user with calculation of metrics, a theoretical analysis of the computational complexity of the scene generation algorithm, which builds 3D scenes using knowledge inference, as well as an evidence of the theoretical analysis based on performance tests.

5.1 3D content creation by users

MICC has been evaluated in terms of user's effort in 3D content creation in SDE using three metrics: the *number of keystrokes and mouse clicks*, *mouse distance* and *time* required to accomplish modeling activities. While, in the evaluation, the number of keystrokes and mouse clicks is representative mainly for measuring the complexity of the MICC method and SDE UI (e.g., the design of and links between windows and controls), mouse distance is representative mainly for measuring the complexity of manual operations performed on the modeled 3D content (e.g., zooming a view, dragging and dropping content elements, etc.). Time covers the entire modeling process—both activities performed by the user and the

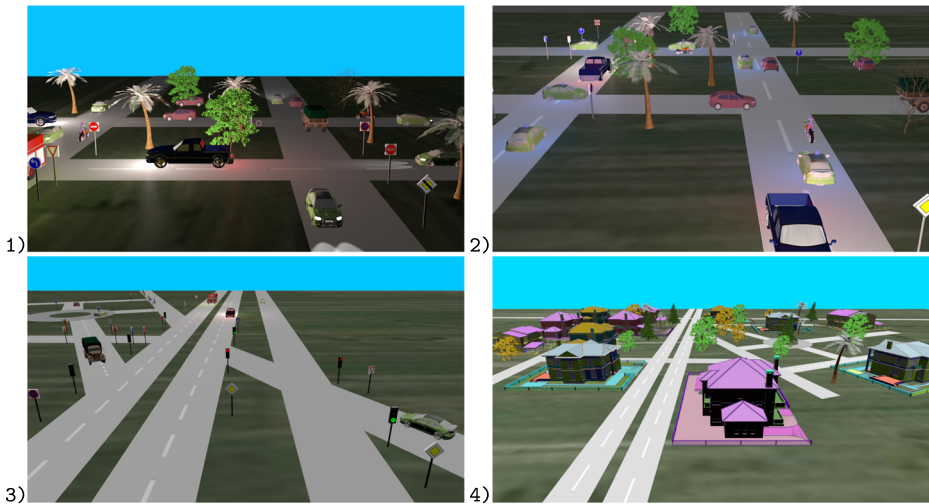


Fig. 10 3D scenes used for the method evaluation: Scene 1 – crossroads with cars using low and high beams; Scene 2 – crossroads with priority vehicles; Scene 3 – crossroads with traffic lights; Scene 4 – a housing with trees

processing of content performed by SDE. The metrics have been measured for the creation of the four 3D scenes (Fig. 10).¹

Every scene has been individually modeled 10 times by a user using Blender and SDE, and the average results have been calculated for every metric and every environment. Further, *average gain* has been calculated for every metric as the average ratio of the result obtained in Blender to the result obtained in SDE. The modeling environments offer different capabilities and are characterized by different degrees of complexity of 3D content creation. Blender is a complex modeling environment with a number of advanced functions directly related to 3D computer graphics, requiring relatively high skills in 3D modeling. SDE is an environment enabling creation of 3D content based on domain concepts linked to 3D components and properties, which makes 3D modeling more intelligible to non-IT-specialists. The tests were performed using a computer equipped with the Intel Core 2 Duo CPU 2.53GHz, 4 GB RAM, NVIDIA GeForce 9500M and Windows 7 Professional. On the computer, both modeling environments—Blender 2.71 as well as SDE (including the Client and the Server)—have been installed.

The average values of the metrics have been calculated for the creation of the 3D scenes (Table 2). The metrics depend on the number of objects in the scene, their structural complexity as well as the number of object properties set while creating the scene. In addition, time required for modeling depends on the geometrical complexity of the objects, since the rendering of more complex shapes requires more computations. The number of objects is the lowest in Scene 2 and the highest in Scene 3, while the structural and geometrical complexity of objects is the lowest in Scene 2 (relatively simple models of traffic signs and traffic lights) and the highest in Scene 4 (relatively complex models of trees and houses). Content creation with Blender requires almost four times more keystrokes and mouse clicks than content creation with SDE. The high value of average gain is caused by the possibility

¹In the scenes, the following models have been used: [http://www.blendswap.com/blends/view/\[68326,68817,69406,56683,71006,2949,58595,61605,63269,64383,67283,66103,69906\]](http://www.blendswap.com/blends/view/[68326,68817,69406,56683,71006,2949,58595,61605,63269,64383,67283,66103,69906]).

Table 2 Metrics calculated for the creation of 3D scenes (B—Blender, S—the scene design environment)

Metric	Scene 1		Scene 2		Scene 3		Scene 4		Average gain (B/S)
	B	S	B	S	B	S	B	S	
Keystrokes and mouse clicks	864	224	674	170	1311	316	1326	333	3,99
Mouse distance (inch)	2543	939	2235	951	4181	1653	4171	1340	2,68
Time (s)	1390	854	1204	744	1721	1118	2504	1145	1,74

of determining a number of object features by setting individual domain properties in SDE. For instance, the color of leaves of a tree and their relative position to the branches are determined by setting the `leafs` property to `green` or `yellow`. Such facility is not available in Blender, in which particular properties must be individually specified. Conceptual modeling with domain properties also implies gain in mouse distance (2.68), as the user is liberated from performing manually complex operations, such as selecting different components of a car, and navigating across different controls of the modeling environment to set individual coordinates of road signs. The average gain in the time required for modeling the scenes (1.74) is also caused by the difference in the number of properties that have to be changed in both the environments.

5.2 Computational complexity of the scene generation algorithm

The analysis of the computational complexity of the algorithm is related to the description of its stages presented in Section 3.2.2. In general, reasoning on RLs that are based on semantic rules, is undecidable, which implies that the algorithm is undecidable. The computational complexity of the algorithm may be determined if no RLs are used in the processed ontologies, and thereby, no undecidable reasoning is performed. For each step of the algorithm, computational complexity has been determined in Table 4. The notation used in the formulas is explained in Table 3.

For the aforementioned formulas, the upper bound of computational complexity may be specified. The number of distinct classes or individuals is less or equal to $2 \cdot n_S$, since classes and individuals may occur in statements as either the subject or the object. The number of distinct properties is less or equal to n_S , since in every statement, only one property occurs. Thus, the overall computational complexity of Stage II is polynomial with the upper bound $O(n_S^3)$. The overall computational complexity of Stage III is polynomial with the upper bound $O(n_S^2)$. To conclude, the scene generation algorithm is undecidable when using RLs, and it has the polynomial computational complexity equal to $O(n_S^3)$, when using no RLs (Table 4).

5.3 Performance of the scene generation algorithm

The performance of the scene generation algorithm implemented in SDE has been evaluated. The tests have been done for scenes whose number of 3D objects changed over the range 5 to 50 with the step equal to 5. Every object in the scene is described by a position, orientation, scale and material. For every number of objects, 20 scenes have been generated and average results have been calculated.

The performance of the algorithm has been evaluated in terms of time required to generate semantic 3D scenes (Fig. 11). Time required for scene generation linearly increases with the increase of the number of 3D objects in the scene. The tests cover creation of

Table 3 Notation used in the computational complexity formulas

n_P	The number of distinct domain properties used in the 3D scene template
n_{OP}	The number of distinct domain object properties used in the 3D scene template
n_S	The number of distinct statements in the sum of the ontology of 3D components, mapping and 3D scene template
$n_{S/OP}$	The number of distinct statements with domain object properties in the 3D scene template
n_{DI}	The number of distinct DIs in the 3D scene template
n_{PO}	The number of distinct POs in the 3D scene template
$n_{PO/CD}$	The average number of distinct POs per CD in the 3D scene template
n_{Cx}	The number of distinct complex descriptors in the 3D scene template
$n_{CD/Cx}$	The average number of distinct DCs per complex descriptor in the 3D scene template
$n_{MP/PO}$	The average number of distinct MPs per PO in the 3D scene template
$n_{CD/PO}$	The average number of distinct DCs per PO in the 3D scene template
n_c	The number of distinct cardinality restrictions in the 3D scene template
n_{hv}	The number of distinct has-value restrictions in the 3D scene template
N	The average cardinality of cardinality restrictions in the 3D scene template

interconnected instances of 3D components for every domain individual, on the basis of DCs (Section 5.2/Step II-E, Stage III). Since the average number of distinct MPs per PO as well as the number of restrictions in the ontologies are constant, the computational complexity of scene generation is linear and equal to $O(n_{PO})$, which is less or equal to $O(n_S)$, where

Table 4 The computational complexity of the scene generation algorithm steps

Stage	Step	Explanation	Complexity
II	A	All the domain properties used in the 3D scene template are analyzed	$O(n_P)$
	B	All the domain object properties used in the 3D scene template are analyzed	$O(n_{OP})$
	C	All the statements with object properties in the 3D scene template are analyzed	$O(n_{S/OP})$
	D	All the DIs in the 3D scene template are analyzed	$O(n_{DI})$
	E	For every PO, $n_{MP/PO}$ MPs are analyzed, on average	$O(n_{PO} * n_{MP/PO})$
	F	For every complex descriptor CD, $n_{CD/Cx}$ DCs are analyzed, on average. For every CD, all of its POs are analyzed	$O(n_{Cx} * n_{CD/Cx} * n_{PO/CD})$
III	A	For every PO, n_c cardinality restrictions are analyzed and for each of them N individuals are generated, on average	$O(n_{PO} * n_c * N)$
	B	For every PO, n_{hv} has-value restrictions are analyzed	$O(n_{PO} * n_{hv})$

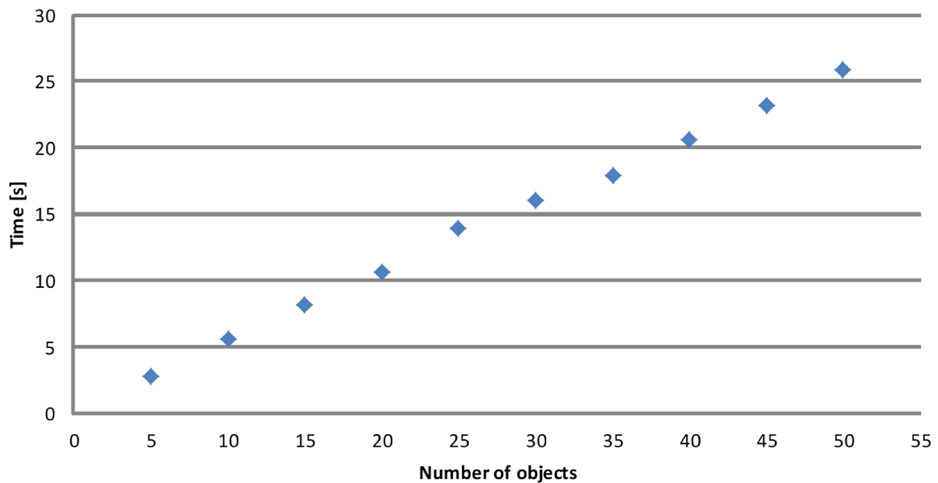


Fig. 11 Time of generating semantic 3D scenes

n_{PO} and n_S are the numbers of POs and statements in the created 3D scene. For instance, if every tree of a class includes two sub-objects of some classes (leaves and branches), which have individual materials and relative positions (described by other classes), the creation of a particular tree covers the creation of the individuals representing the sub-objects, their materials and relative positions, and assigning appropriate datatype properties to these individuals. As shown in the graph, time required for processing more complex 3D scenes on a moderate computer may be tens of seconds. For single objects, time is much shorter, and the results seem to be instant.

6 Discussion

6.1 Test results

The test results show that the proposed method has higher efficiency of 3D content creation than the manual approach available in visual 3D modeling environments represented by Blender. The results show that the user's effort in 3D content creation with MICC is much lower than the effort in 3D modeling with other environments. The advantage includes both the modeling actions accomplished by users (mouse distance, the number of keystrokes and mouse clicks) and time required for modeling. The predominance follows from modeling 3D content at a conceptual level, with domain-specific classes and properties. This permits the modeling environment to be more oriented on a particular application domain. Thereby, content can be created by users without considerable expertise in 3D modeling, in particular domain experts, who are liberated from direct specification of details related to computer graphics. The modeling actions, which do not demand changing multiple properties attainable on different panels, become less complex and less time-consuming.

6.2 Alternative 3D content representations

3D content representations that are alternative to ontology-based representations are: imperative programming languages with libraries (e.g., OpenGL [1] and Direct3D [39]),

declarative 3D formats (e.g., X3D [51] and XML3D [18]) as well as declarative programming languages (e.g., Prolog). Inference-based content creation is not available in imperative languages, including languages used in 3D modeling environments. In imperative languages, inference must be implemented in advance. This requires more effort in implementation from content authors, who must consider possible consequences of explicitly specified content properties in the context of a particular use case.

Declarative 3D content formats have not been intended for knowledge inference and they address representation of components and properties specific to 3D content, without the conceptual level specific to an application or domain. These features limit the possibilities of 3D modeling with such formats in comparison to ontologies.

In contrast to the aforementioned approaches, declarative logic programming languages enable knowledge inference at different levels of abstraction. Like in the case of ontologies, such languages are processed by reasoning engines, which liberate content authors from specifying all content properties and from implementing reasoning algorithms on their own. Moreover, on the one hand, due to the use of rules (implications), which may express arbitrary links between properties of classes and objects, such languages have higher expressiveness than ontologies encoded in RDF, RDFS and OWL. On the other hand, reasoning on rules is undecidable, which may prevent getting results in general.

7 Conclusions and future works

The use of semantics for building VR/AR applications gains increasing attention in the research community. However, although a number of methods and tools have been developed on the basis of the semantic web approach, they do not benefit from the possibilities of inferring tacit knowledge during the 3D content creation process. Inference of tacit knowledge can liberate content authors from specifying all 3D content components and properties, which can be determined as consequences of some explicitly specified components and properties. Hence, the overall effort in 3D modeling can be reduced.

The proposed method MICC uses inference of tacit knowledge in linking 3D graphics with an arbitrary application or domain for which 3D content is being created. Such a general domain-independent approach goes beyond the current state of the art in 3D modeling. The method has been implemented and evaluated. In comparison to widely used 3D modeling tools, it reduces users' effort in modeling in terms of the required time as well as the number of executed actions. MICC enables 3D content creation using concepts specific to different application domains. This opens new opportunities to develop tools for domain experts who are not IT-specialists. Moreover, semantically encoded 3D content is more suitable for indexing, searching and analyzing than 3D content encoded in non-semantic formats and languages.

The possible directions of future research are mainly related to the extension of the current semantic 3D content format with time to enable creation of dynamic 3D content. In particular, the solution proposed in [25, 27] could be applied. Then, methods of exploring non-deterministic paths in the execution of VR/AR applications (e.g., by giving some assumptions about possible future application execution), and querying about 3D content properties at different points in time could be devised. Third, methods and tools enabling 4-dimensional modification of 3D content (including time) with user queries can be developed, e.g., adding new objects' activities. Next, the development of user-friendly graphical modeling tools could benefit from the syntax of the representation, which is relatively simple in comparison to the syntax of imperative languages. Such tools could liberate users

from programming, which could further improve the overall dissemination of 3D content created by domain experts. In addition, mapping of 3D components to domain-specific concepts could be done automatically on the basis of machine learning methods using multiple sources of annotated 3D objects. Such an approach is increasingly used in different domains, covering feature modeling [14] and career prediction [38]. Finally, the approach could also be used to semantically represent real world state recognized on the basis of data retrieved from sensors, in particular representing actions and more complex daily activities [36, 37].

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Open Access This article is distributed under the terms of the Creative Commons Attribution 4.0 International License (<http://creativecommons.org/licenses/by/4.0/>), which permits unrestricted use, distribution, and reproduction in any medium, provided you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license, and indicate if changes were made.

References

1. Opendl (2018). <https://www.opendl.org>
2. Owl web ontology language reference (2018). <http://www.w3.org/TR/owl-ref>
3. Protégé (2018). <http://protege.stanford.edu/>
4. Rdf 1.1 turtle (2018). <http://www.w3.org/TR/turtle>
5. Albrecht S, Wiemann T, Günther M, Hertzberg J (2011) Matching cad object models in semantic mapping. In: Proceedings ICRA 2011 Workshop: Semantic Perception, Mapping and Exploration, SPME
6. Attene M, Robbiano F, Spagnuolo M, Falcidieno B (2007) Semantic annotation of 3d surface meshes based on feature characterization. In: Proceedings of the semantic and digital media technologies 2nd international conference on semantic multimedia, SAMT'07. Springer, Berlin, pp 126–139
7. Attene M, Robbiano F, Spagnuolo M, Falcidieno B (2009) Characterization of 3d shape parts for semantic annotation. *Comput Aided Des* 41(10):756–763. <https://doi.org/10.1016/j.cad.2009.01.003>
8. Autodesk (2018) 3ds max. <http://www.autodesk.pl/products/autodesk-3ds-max>
9. Autodesk (2018) Maya. <https://www.autodesk.pl/products/maya>
10. Away3D (2017) Away3d. <http://away3d.com/>
11. Berners-Lee T, Hendler J, Lassila O et al (2001) The semantic web. *Sci Amer* 284(5):28–37
12. Bille W, De Troyer O, Pellens B, Kleiner mann F (2005) Conceptual modeling of articulated bodies in virtual environments. In: Thwaites H (ed) Proceedings of the 11th international conference on virtual systems and multimedia (VSMM). Archaeolingua, Ghent, pp 17–26
13. Blender (2017) Blender api documentation. http://www.blender.org/api/blender_python_api_2.73_release
14. Chaudhuri S, Kalogerakis E, Giguere S, Funkhouser T (2013) Attribit: content creation with semantic attributes. In: Proceedings of the 26th Annual ACM Symposium on User Interface Software and Technology, UIST '13. ACM, New York, pp 193–202
15. De Floriani L, Hui A, Papaleo L, Huang M, Hendler J (2007) A semantic web environment for digital shapes understanding. In: Semantic multimedia. Springer, pp 226–239
16. De Troyer O, Kleiner mann F, Mansouri H, Pellens B, Bille W, Fomenko V (2007) Developing semantic vr-shops for e-commerce. *Virtual Reality* 11(2-3):89–106
17. De Troyer O, Kleiner mann F, Pellens B, Bille W (2007) Conceptual modeling for virtual reality. In: Grundy J, Hartmann S, Laender AHF, Maciaszek L, Roddick J (eds) Tutorials, posters, panels and industrial contributions at the 26th int. conference on conceptual modeling - ER 2007. CRPIT, vol 83. ACS, Auckland, pp 3–18
18. DFKI (2017) Computergraphics Lab of the Saarland University, I.V.C.I.: Xml3d. <http://xml3d.org/>
19. Drap P, Papini O, Sourisseau JC, Gambin T (2017) Ontology-based photogrammetric survey in underwater archaeology. In: European semantic web conference. Springer, pp 3–6
20. Fischbach M, Wiebusch D, Giebler-Schubert A, Latoschik ME, Rehfeld S, Tramberend H (2011) Sixton's curse - Simulator X demonstration. In: Hirose M, Lok B, Majumder A, Schmalstieg D (eds) Virtual Reality Conference (VR), 2011 IEEE. pp 255–256. <https://doi.org/10.1109/VR.2011.5759495>

21. Flotyński J, Walczak K (2013) Semantic modelling of interactive 3D content. In: Proceedings of the 5th joint virtual reality conference. Paris, France
22. Flotyński J, Walczak K (2014) Conceptual knowledge-based modeling of interactive 3D content. *The Visual Computer* 31(10):1287–1306. <https://link.springer.com/article/10.1007/s00371-014-1011-9>
23. Flotyński J, Walczak K (2014) Semantic representation of multi-platform 3D content. *Comput Sci Inf Syst* 11(4):1555–1580
24. Flotyński J, Walczak K (2016) Customization of 3D content with semantic meta-scenes. *Graph Model* 88:23–39
25. Flotyński J, Walczak K (2017) Knowledge-based representation of 3D content behavior in a service-oriented virtual environment. In: Proceedings of the 22nd International Conference on Web3D Technology, Brisbane (Australia), p. Article No 14. ACM, New York
26. Flotyński J, Walczak K (2017) Ontology-based representation and modelling of synthetic 3d content: a state-of-the-art review. *Computer Graphics Forum* 36(8):329–353
27. Flotyński J, Krzyszkowski M, Walczak K (2017) Semantic composition of 3d content behavior for explorable virtual reality applications. In: Barbic J, D’Cruz M, Latoschik ME, Slater M, Bourdot P (eds) *Virtual reality and augmented reality 14th euroVR international conference, EuroVR 2017*. Laval, France, December 12–14, 2017. Proceedings in lecture notes in computer science. Springer, pp 3–23
28. Foundation B (2017) Blender. <http://www.blender.org>
29. Gruber T Encyclopedia of database systems. On-line (accessed March 28, 2015). <http://tomgruber.org/writing/ontology-definition-2007.htm>
30. Gutiérrez M (2005) Semantic virtual environments
31. Gutiérrez M, Thalmann D, Vexo F (2005) Semantic virtual environments with adaptive multimodal interfaces. In: Chen YPP (ed) *MMM*. IEEE Computer Society, pp 277–283
32. Kalogerakis E, Christodoulakis S, Moutoutzis N (2006) Coupling ontologies with graphics content for knowledge driven visualization. In: *VR ’06 Proceedings of the IEEE conference on Virtual Reality, Alexandria*, pp 43–50
33. Kapahnke P, Liedtke P, Nesbigall S, Warwas S, Klusch M (2010) Isreal: an open platform for semantic-based 3d simulations in the 3d internet. In: *International semantic web conference (2)*. pp. 161–176
34. Kleinermann F, De Troyer O, Mansouri H, Romero R, Pellens B, Bille W (2005) Designing semantic virtual applications. applications. In: Proceedings of the 2nd INTUITION International Workshop, Senlis, pp. 5–10
35. Latoschik ME, Tramberend H (2011) Simulator x: a scalable and concurrent software platform for intelligent realtime interactive systems. In: Proceedings of the IEEE VR 2011
36. Liu Y, Nie L, Han L, Zhang L, Rosenblum DS (2016) Action2activity: Recognizing complex activities from sensor data. arXiv:1611.01872
37. Liu Y, Nie L, Liu L, Rosenblum DS (2016) From action to activity: Sensor-based activity recognition. *Neurocomputing* 181:108–115. <http://www.sciencedirect.com/science/article/pii/S0925231215016331>, big data driven intelligent transportation systems
38. Liu Y, Zhang L, Nie L, Yan Y, Rosenblum DS (2016) Fortune teller: Predicting your career path. In: Proceedings of the thirtieth AAAI conference on artificial intelligence, AAAI’16. AAAI Press, pp 201–207. <http://dl.acm.org/citation.cfm?id=3015812.3015842>
39. Microsoft (2017) Direct3d 11.1 features. <https://msdn.microsoft.com/en-us/library/windows/desktop/>
40. Oracle (2017) Java3d. <http://www.oracle.com>
41. Papaleo L, De Floriani L, Hendler J, Hui A (2007) Towards a semantic web system for understanding real world representations. In: Proceedings of the Tenth International Conference on Computer Graphics and Artificial Intelligence
42. Pellens B, De Troyer O, Bille W, Kleinermann F, Romero R (2005) An ontology-driven approach for modeling behavior in virtual environments. In: Meersman R, Tari Z, Herrero P (eds) Proceedings of on the move to meaningful internet systems 2005: Ontology mining and engineering and its use for virtual reality (WOMEUVR 2005) workshop. Springer, Cyprus, pp 1215–1224
43. Perez-Gallardo Y, Cuadrado JLL, Crespo AG, de Jesús CG (2017) Geodim: a semantic model-based system for 3d recognition of industrial scenes. In: *Current trends on knowledge-based systems*. Springer, pp 137–159
44. Robbiano F, Attene M, Spagnuolo M, Falcidieno B (2007) Part-based annotation of virtual 3d shapes. *2013 Int Conf Cyberworlds* 0:427–436
45. Sikos LF (2017) 3D model indexing in videos for content-based retrieval via x3d-based semantic enrichment and automated reasoning. In: Proceedings of the 22nd international conference on 3D web technology. ACM, p 19
46. Trellet M, Férey N, Flotyński J, Baaden M, Bourdot P (2018) Semantics for an integrative and immersive pipeline combining visualization and analysis of molecular data. *J Integr Bioinform* 15(2):1–19

47. Trellet M, Ferey N, Baaden M, Bourdot P (2016) Interactive visual analytics of molecular data in immersive environments via a semantic definition of the content and the context. In: 2016 workshop on immersive analytics (IA). IEEE, pp 48–53
48. Van Gool L, Leibe B, Müller P, Vergauwen M, Weise T (2007) 3d challenges and a non-in-depth overview of recent progress. In: 3DIM, pp 118–132
49. W3C (2004) Swrl. On-line. <http://www.w3.org/Submission/SWRL/>
50. (2017) W3C: Vml virtual reality modeling language. <https://www.w3.org/MarkUp/VRML>
51. W3C (2018) Getting started with x3d. <http://www.web3d.org/getting-started-x3d>
52. W3C (2018) Owl. <http://www.w3.org/2001/sw/wiki/OWL>
53. W3C (2018) Rdf. <http://www.w3.org/TR/2004/REC-rdf-concepts-20040210>
54. W3C (2018) Rdfs. <http://www.w3.org/TR/2000/CR-rdf-schema-20000327>
55. W3C (2018) Sparql query language for rdf. <http://www.w3.org/TR/2008/REC-rdf-sparql-query-20080115>
56. Wiebusch D, Latoschik ME (2012) Enhanced Decoupling of Components in Intelligent Realtime Interactive Systems using Ontologies. In: Software Engineering and Architectures for Realtime Interactive Systems (SEARIS), proceedings of the IEEE Virtual Reality 2012 workshop, pp 43–51
57. Zahariadis T, Daras P, Laso-Ballesteros I (2008) Towards future 3d media internet. NEM Summit, pp 13–15



Dr. hab. eng. Krzysztof Walczak (male): Associate Professor in the Department of Information Technology at PUEB, holds a habilitation degree (higher Ph.D. degree) in computer science (multimedia systems). His research interests focus on multimedia systems, virtual and augmented reality, distance teaching and learning, semantic web and databases. He is the author or co-author of two books, several book chapters and 120 research papers in journals and conference proceedings. He also holds several US and European patents in these domains. He was acting as a technical coordinator in numerous research and industrial projects. He is a founding member and an Executive Committee member of the EuroVR Association, a member of the Board of Directors of the VSMM (International Society on Virtual Systems and Multimedia), a member of the Web3D Consortium and a member of the ACM (Association for Computing Machinery).



Dr. eng. Jakub Flotyński (male): Assistant Professor in the Department of Information Technology at PUEB, holds a Ph.D. degree in computer science (multimedia systems). His research interests include virtual and augmented reality, semantic web, service-oriented architectures and mobile applications. He has participated in several research and industrial projects in these fields. He is the author or co-author of 34 research papers. He is an ACM member, and he was a Program Co-Chair of the International ACM Conference on 3D Web Technology in 2017 and 2018.