The final publication is available at Springer via http://dx.doi.org/10.1007/978-3-642-54734-8_8

Multi-platform Semantic Representation of Interactive 3D Content*

Jakub Flotyński, Krzysztof Walczak

Poznań University of Economics,
Niepodległości 10, 61-875 Poznań, Poland
{flotynski, walczak}@kti.ue.poznan.pl
http://www.kti.ue.poznan.pl

Abstract. Three-dimensional user interfaces offer a powerful presentation medium enabling rich interactive visualization of various types of information on a view of the real world in mixed reality presentations. The use of interactive 3D content becomes increasingly popular in a variety of application domains, such as education, training, entertainment and social media, enhancing users' capabilities to access—in a meaningful way—various rich information sources, and potentially increasing their collective awareness. The coverage of various available presentation tools can improve reuse of 3D content elements in different presentations. However, the available approaches to 3D content design do not enable flexible methods of multi-platform 3D content presentation. The main contribution of this paper is a semantic representation of information that can be presented in the form of interactive 3D content. The proposed approach enables flexible and efficient creation of 3D presentations covering a wide range of target platforms—visualization tools, content representation languages and programming libraries. Referring to the semantics of particular 3D content elements facilitates content creation by non-IT-specialists, and it can improve indexing, search and analysis of 3D content on the web.

Keywords: 3D web, Semantic Web, 3D content, multi-platform, ontology

1 Introduction

Widespread use of interactive 3D technologies and multimedia platforms, including powerful and omnipresent mobile devices, has been recently enabled by the significant progress in hardware performance, the rapid growth in the available network bandwidth as well as the availability of versatile input-output devices. 3D technologies become increasingly used in various application domains, such as eduction, training, entertainment and social media, significantly enhancing possibilities of presentation and interaction with rich information sources, thus increasing collective awareness of their users.

However, a potentially high number of recipients of 3D content on the web, results in a diversity of hardware and software systems used, and thereby a multitude of available 3D content presentation. Currently, 3D content browsers and presentation tools are typically separately implemented for different hardware and software systems. However, in comparison to the development of individual 3D content browsers and presentation tools for different target systems, compatibility of 3D content representations with various popular presentation platforms could improve the reuse of common 3D

* This research work has been supported by the Polish National Science Centre Grant No. DEC-2012/07/B/ST6/01523.

content components and overall use of 3D content. In such approach, 3D content may be created once and then presented using different platforms. Moreover, such approach does not require users to install additional software, but it can leverage well-established 3D content browsers and presentation tools that may already be installed in the users' systems (e.g., Adobe Flash Player and WebGL or X3DOM-compliant web browsers). However, currently, development of 3D platforms is driven by large industry players in a competitive environment and the issue of multi-platform 3D content presentation is still neglected, resulting in fragmentation of content and technologies. This is an important obstacle preventing mass use of 3D interfaces for data presentation.

The main contribution of this paper is a multi-platform semantic representation of information that may presented in the form of interactive 3D content. The approach permits flexible and efficient creation of 3D content for a variety of target presentation platforms. In the proposed solution, once the structure of 3D content is designed, it can be automatically transformed into different final presentation forms, which are suited to different 3D content presentation platforms. The selection of the target platforms to be used is an arbitrary decision of a system designer. Referring to the semantics of particular 3D content components and the conformance to well-established Semantic Web standards enables 3D content representation that is independent of particular browsers and presentation tools, permits reflection of complex dependencies and relations between content components, and facilitates indexing, search and analysis of 3D content.

The remainder of this paper is structured as follows. Section 2 provides an overview of multi-platform and semantic representation of 3D content in the context of collective awareness systems. Section 3 contains a brief introduction to a method of semantic creation of 3D content. Section 4 focuses on the new multi-platform model of 3D content, which is used in the modelling method. Section 5 includes an example use of the model. Finally, Section 6 concludes the paper and indicates the possible future research.

2 Interactive 3D Content in Collective Awareness Systems

Several works have been devoted to 3D content presentation across different platforms in collective systems. In [1], a specific 3D browser plug-in for different web browsers has been described. In [2], an approach to hardware multi-platform 3D content presentation based on MPEG-4 has been explained. In [3], an approach to multi-platform visualization of 2D and 3D tourism information has been presented. In [4], an approach to adaptation of 3D content complexity with respect to the available resources has been proposed. In [5], a multi-platform on-line game has been presented. In [6], an approach to building multi-platform virtual museum exhibitions has been proposed.

Numerous works have been devoted to semantic 3D representation of information in collective systems. In [7], integration of X3D and OWL using scene-independent ontologies and the concept of semantic zones have been proposed. In [8], an ontology for X3D as well as semantic properties for coupling VR scenes with domain knowledge have been described. In [9], a 3D content representation based on reusable elements with specific roles has been introduced. In [10], an approach to generating virtual words upon domain ontologies has been considered. In [11], a concept of semantic entities in VR applications have been discussed.

3 Overview of the Method of Semantic 3D Content Creation

Although several approaches have been proposed for semantic modelling of 3D content, they lack general solutions for flexible creation of 3D representation of data retrieved from rich web information sources. Such representations could be created at arbitrarily high levels of abstraction, by domain experts, who are not required to be IT-specialists, to improve widespread dissemination of 3D content in different domains.

This section includes an outline of a method of semantic creation of interactive 3D content (proposed in [14]), which leverages the model of multi-platform 3D content, which will be proposed in the next section. The main goal of the method is to enable flexible 3D content creation by non-IT-specialists. The method can significantly reduce the design effort, by enabling the reuse and assembly of common elements with respect to their dependencies that may be described with domain-specific ontologies and knowledge bases. Furthermore, the possible use of various ontologies permits creation and analysis of 3D content at different (arbitrarily chosen) levels of semantic abstraction. Finally, 3D content described by commonly used concepts is platform- and standard-independent, and it may be transformed to final 3D representations encoded in different languages, e.g., depending on the target presentation platform used.

In the presented method, the creation of 3D content is a sequence of partly dependent activities (Fig. 1). Modelling of 3D content in the first three activities—design of a concrete representation of 3D content, mapping the concrete representation to domain-specific concepts, and design of a conceptual representation of 3D content—is performed by a developer and a domain expert. These activities leverage appropriate semantic models and produce semantic documents. The two following activities of the method—expanding the semantic representation and building the final representation—are performed automatically by specific software. The activities precede 3D content presentation, which may be done using various content presentation platforms.

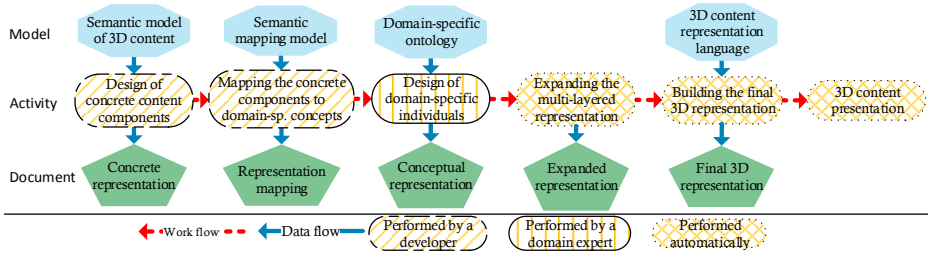

Fig. 1: Semantic creation of 3D content

The design of a concrete semantic representation provides particular components of 3D content to enable presentation of domain-specific concepts that will be further used by a domain expert in activity 3. This activity is typically performed by a developer—an IT-specialist with technical skills in 3D modelling. The result of this activity is a concrete semantic representation of 3D content, which is a knowledge base compliant with the platform-independent semantic model of 3D content (PIM–proposed in [12]). The model incorporates semantic elements (components and properties) that are directly related to 3D content, e.g., meshes, groups of objects, materials, viewpoints, events, etc.

The mapping of a concrete 3D content representation (created in activity 1) to domain-specific semantic concepts enables 3D presentation of domain-specific knowledge bases (that will be created in activity 3). The result of this activity is a semantic mapping document that is a knowledge base compliant with the conceptual model of 3D content (proposed in [13]). Mapping is performed by a developer once for a particular domain-specific ontology and a concrete representation, and it enables the reuse of concrete components and properties for forming 3D representations of various domain-specific individuals, which conform to the domain-specific ontology selected.

The design of a conceptual semantic representation enables creation of 3D content at a high level of abstraction with a domain-specific ontology. This activity can be performed many times for a particular domain-specific ontology, a concrete representation and a mapping. This is typically performed by a domain expert (a non IT-specialist), but can be also performed automatically based on some data sources. The result of this activity is a conceptual semantic representation of 3D content, which is a knowledge base compliant with the domain-specific ontology selected.

The following activities of the content creation process can be performed automatically. Expanding the representation multiplies the concrete components (created in activity 1), which are associated with domain-specific concepts, and assigns them directly to domain-specific individuals (created in activity 3). The result of this activity is an expanded representation of 3D content—a knowledge base compliant with the PIM.

The last activity in the content creation process is a transformation of the expanded representation to its final 3D counterparts, which are encoded using particular 3D content representation languages. This stage of the method is an extension of the approach proposed in [6], and it uses the multi-platform model of 3D content, which is the main contribution of this paper.

4 Multi-platform Model of 3D Content

Although numerous solutions have been proposed for creating semantic representations of 3D content, they do not enable flexible creation of multi-platform 3D content, which can visualize various types of information using a multitude of available content presentation tools.

In this section, a new multi-platform model of 3D content is proposed (Fig. 2a). The model extends the semantic model of 3D content (proposed in [12, 13]) and enables the last activity of the content creation method (proposed in [14])—building a final 3D content representation. The model consists of three parts—*platform-independent model of 3D content* (PIM), *mappings* (MPs) and *platform-specific template sets* (PSTSs). The PIM (cf. [12] and Section 3) is common for all *platform-independent 3D content representations*, which include semantic 3D content components, and which are agnostic on particular presentation platforms. Semantic individuals included in a PIR can be presented on different platforms when transformed with appropriate PSTSs. PSTSs may leverage available 3D content representation languages (e.g., X3D, Java), programming libraries (e.g., Java3D, Away3D) and game engines (e.g., Unity, Unreal). Each PSTS is mapped to the PIM by an MP. An individual PSTS and its corresponding MP are created once every time a new presentation platform is added to the system (Fig. 2b). Since a PSTS and an MP are introduced, they may be used for the development of various 3D

presentations that are permitted by the new platform (Fig. 2c). The particular parts of the proposed representation are described in the following subsections.

4.1 Platform-independent Representations

A *platform-independent representation* (PIR) of 3D content is a knowledge base comprised of semantic individuals, which reflect 3D content components, and properties, which reflect dependencies and relations between the components. Every PIR is compliant with the PIM. A PIR declaratively describes 3D content, as it is a set of semantic *statements* and *rules*.

Every *statement* is a triple consisting of a *subject*, a *property* and an *object*. *Statements* are used to describe aspects of 3D content that are related to geometry (e.g., shape), structure (e.g., sub-components), space (e.g., position), appearance (e.g., texture) and simple behaviour (e.g., animations).

Every *rule* is an *implication* including a *body* and a *head*, which are conjunctions of *statements*. A *rule* is interpreted in the following way: if the *body* is satisfied, then the *head* is also satisfied. Rules are used to describe complex behaviour of 3D content, in particular complex animations and interactions, which may depend on multiple factors, such as time, state of objects, user actions, etc.

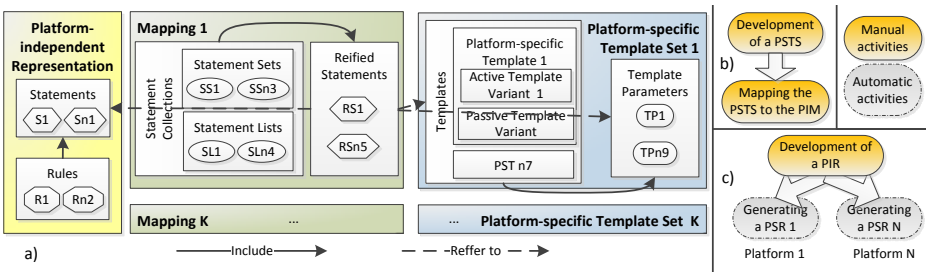

Fig. 2: Multi-platform model of 3D content (a), introducing a new platform to the system (b), and modelling multi-platform 3D content (c)

4.2 Platform-specific Representations

A *platform-specific representation* (PSR) of 3D content is a counterpart to a PIR, and it includes 3D objects and 3D scenes, which are encoded using a 3D content representation language or a programming library. Every PSR is a combination of parametrized *platform-specific templates* (PSTs), which are fragments of code (e.g., sequences of instructions) of a 3D content representation language. PSTs may be mapped to individual *statements* as well as to *statements* that are included in semantic *rules* of PIRs. For some of the target platforms used, this context of mapping may determine the form of the template used. Every PST may be given in two *variants* that differ depending on the context of its use. *Active template variants* (ATVs) are used for the independent *statements* and the *statements* included in the *heads* of *rules* (to express logical results), while *passive template variants* (PTVs) are used for the *statements* included in the *bodies* of *rules* (to express conditions). For instance, the *statement* `[object pim:shape "cone"]` may be transformed to an ATV `[object.shape = "cone"]` (which is encoded in an object-oriented language) or to a PTV `[object.shape.isEqual("cone")]`.

To enable flexible transformation of PIRs to PSRs, the granularity of PSTs should reflect the semantic granularity of their corresponding *statements*—PSTs should neither extend nor narrow the meaning of the *statements*.

Template parameters (TPs) enable the reuse of individual PSTs in different contexts—in combination with other PSTs, e.g., by joining or nesting PSTs. In the proposed approach, the 3D content representation languages and programming libraries, which are selected for encoding PSTs are typically determined by the target 3D content presentation platforms used. For instance, the use of the Bitmanagement BS Contact browser implies the selection of the declarative X3D language, while the use of the Adobe Flash Player—the selection of the imperative ActionScript language.

All PSRs that are presented on a common target platform conform to a common PSTS that is mapped to the PIM by a common MP.

4.3 Mappings

A *mapping* (MP) is a knowledge base linking an individual PSTS to the PIM. An MP specifies a transformation of PIRs to the corresponding PSRs that are to be presented on a common target platform. Semantic individuals contained in an MP do not influence the modelled 3D content. An MP consists of *reified statements* (RSs), which link *statements* (that may potentially occur in PIRs) to their corresponding PSTs (that may potentially occur in the resulting PSRs). Every RS is a semantic pattern that matches a group of possible *statements*. Linking a group of *statements* to an RS is performed by semantic generalization. A generalization may pertain to the subject, the property or the object of *statements*. For instance, a possible generalization (an RS) of the *statement* `[object pim:color "red"]` in terms of property, is the *statement* `[object pim:appearanceProperty "red"]`, while a possible generalization of the *statement* `[object rdf:type pim:Mesh3D]` in terms of object, is the *statement* `[object rdf:type pim:GeometricalComponent]`.

A target PSR, which is generated based on *statements* of a PIR, may require an exchange of TPs in a group of different PSTs, which are mapped to these *statements*, and it may require a specific order of PSTs. For this purpose, RSs may be gathered into *statement collections—statement sets* (SSs), which do not respect the order of PSTs, and *statement lists* (SLs), which do respect the order of PSTs in the resulting PSR. For instance, while a declarative PIR including the pair of *statements* `[light pim:intensity "10". light rdf:type pim:DirectionalLight.]` does not depend on the order of the *statements*, a corresponding imperative PSR may require the following order of the associated PSTs (instructions) `[DirectionalLight light = new DirectionalLight(); light.intensity = 10;]`.

5 Example of a Multi-platform 3D Content Representation

In this section an example multi-platform representation of 3D content is discussed. The example 3D scene (Fig. 3) could be used in, e.g., a collective educational or a virtual museum system. The scene includes a light source and a 3D model of a plough, which has been retrieved from a virtual museum of agriculture. A teacher can rotate the model (by dragging it) to present it in detail to students, who are distributed in the environment and, overall, who may use different 3D content browsers. In the example, the Adobe Flash Player is selected as a target presentation platform. However, in general, an arbitrary number of various platforms could be used.

Listing 1.1: A platform-independent representation of 3D content

```
01: S1: scene rdf:type pim:Scene.
02: S2: scene pim:backgroung "beige".
03: S3: light pim:intensity "10.0".
04: S4: light rdf:type pim:DirectionalLight.
05: S5: plough rdf:type pim:StructuralComponent.
06: S6: plough pim:includes box , wheel , frame.
07: S7: box, wheel, frame rdf:type pim:GeometricalComponent.
08: rdf:type(?MDE, pim:MouseDragEvent), pim:targetObject(?MDE, plough), pim:
    mouseDownX(?MDE, ?DOWN_POS_X), pim:mouseUpX(?MDE, ?UP_POS_X) -> pim:
    orientation(?plough, [0, 1, 0, ?UP_POS_X-?DOWN_POS_X]).
```

```
01: PST1: {
02:   //imports
03:   $declarations
04:   public function Main():void {
05:     var $object:View3D = new View3D
    ();
06:     $actions } }
07: PST2: { $object.backgroundColor =
    $data; }
08: PST3: { $object1.addChild($object2);
    }
09: PST4[0]: {
10:   // variables storing initial and
    final mouse positions
11:   var $start_pos_x:Number;
12:   var $end_pos_x:Number;
13:   ...
14:   private function onMouseDown(event
    :MouseEvent):void {
15:  if ($condition) {
16:    drag = true;
17:    //setting initial position
18:  }}
19:   private function onMouseUp(event:
    MouseEvent):void {
20:  if (drag) {
21:    //setting final position
22:    $actions
23:    _view.render(); }
24:  drag = false;} }
25: PST4[1]: {
26:   $view.addEventListener(MouseEvent.
    MOUSE_DOWN, $onMouseDown);
27:   $view.addEventListener(MouseEvent.
    MOUSE_UP, $onMouseUp); }
28: PST5: { event.target == $target }
29: PST6: { $object.rotate(new Vector3D(
    $data[0], $data[1], $data[2]),
    $data[3]); }
```

Listing 1.2: A platform-specific template set for an object-oriented language

Fig. 3: A final 3D content representation

```
01: SL1: { RS1: ?object rdf:type pim:
    ContentComponent.
02: RS2: ?object pim:dataProperty ?
    data.
03: RS1.$actions == RS2 }
04: SL2: { SS: { RS1: ?object1 rdf:
    type pim:ContentComponent.
05: RS2: ?object2 rdf:type pim:
    ContentComponent. }
06: RS3: ?object1 pim:objectProperty ?
    object2. }
07: SS1: { RS1: ?object rdf:type pim:
    Scene.
08: RS2: ?object rdf:type pim:
    MouseEvent.
09: RS1.$declarations == RS2[0]
10: RS1.$actions == RS2[1] }
11: SS2: { RS1: ?object rdf:type pim:
    MouseEvent.
12: RS2: ?object pim:targetObject ?
    target.
13: RS1[1].$condition == RS2 }
14: SS3 { RS1: ?object rdf:type pim:
    MouseEvent.
15: RS2: ?object pim:mouseDownX ?data.
16: RS1[1].$start_pos_x == RS2.?data }
17: SS4 { RS1: ?object rdf:type pim:
    MouseEvent.
18: RS2: ?object pim:mouseUpX ?data.
19: RS1[1].$end_pos_x == RS2.?data }
```

Listing 1.3: The mapping of statements to platform-specific templates

A PIR of the scene is presented in Listing 1.1, and it uses the PIM (the `pim` prefix) to describe the 3D content components regardless of any particular target platforms. A PSTS and an MP, which enable encoding of PIRs using the ActionScript imperative programming language and the Away3D library, are presented in Listing 1.2 (the `psts` prefix) and in Listing 1.3 (the `mp` prefix). The listings include only the PSTs and the RSs that are crucial for the discussion.

The S1 and S2 *statements* (1.1/1-2) create the `scene` and set its background with the appropriate color. The *statements* are processed according to the SL1 (1.3/1-3),

which includes two RSs. The RS1 links the S1 to the PST1 (1.2/1-6), while the RS2 links the S2 to the PST2 (1.2/7). The PST1 is a template of an ActionScript document and it initializes a 3D view, while the PST2 sets the background of a scene. The order of the RSs in the SL1 determines the order of the PST1 and PST2 in the resulting document (a PSR). The SL1 requires a new object to be created before any properties of this object are set (according to the imperative programming paradigm). If the PST that is linked to the RS1 includes the `$actions` TP, the PST linked to the RS2 is injected into this TP (1.3/3), else the PST of the RS2 follows the PST of the RS1 in the final PSR. The S3 and S4, which create a light source (1.1/3-4), are processed in the same order (determined by the SL1) as the previous *statements*, independently of their initial order in the PIR. The `plough`, which is a virtual museum artefact, is a `structural component` that consists of the three `geometrical components`—a `box`, a `wheel` and a `frame` (1.1/5-7). The SL2 (1.3/4-6) is applied to the S5-S7 (1.1/5-7)—in the resulting PSR, instructions that create new objects in the scene, precede instructions that link these objects by some properties (1.2/8). In (1.1/8), a rule is specified to enable rotation of the `plough` by dragging it. The PST4, which is responsible for event handling (1.2/9-24), is injected into the `$declarations` TP of the PST1, and the event handlers are added to the scene (1.2/25-27) as indicated by the SS1 (1.3/7-10). The target object of the event is verified by the PST5 (1.2/28), which is injected into (1.2/15) by the SS2 (1.3/11-13). The `DOWN_POS_X` and `UP_POS_X` variables reflect the initial and the final X coordinates when dragging the `plough`, and they are available within the rule. The values of these variables are set by the SS3 and SS4 (1.3/14-19). The PST6, which rotates the `plough` (1.2/29), is injected in (1.2/22) as no restrictions are given for its corresponding *statement* by any *statement collections*.

In the presented example, the resulting PSR (an ActionScript document) includes the PSTs in the required order, with the proper TPs specified. The PIR has been created with the Protégé editor. However, a tool for visual semantic modelling can be developed.

6 Conclusions and Feature Works

In this paper, a new approach to building multi-platform abstract representations of 3D content has been proposed. The presented solution has several important advantages in comparison to the available approaches to 3D content presentation. First, it is more convenient for environments, which cover various hardware and software systems, as the development of the proposed PSTSs and MPs requires less effort than the development of individual content models or content browsers. Second, the possible use of well-establish 3D content presentation tools, programming languages and libraries liberates users from the installation of additional software, which can improve the dissemination of 3D content. Moreover, semantic representation enables more efficient and flexible methods of indexing, search and analysis of the content regarding complex dependencies and relations between its components, rules of combining different components as well as complex declarative descriptions of content behaviour.

The following directions of future research are possible. First, the proposed approach needs to be fully implemented including the automatic transformation of semantic representations to final 3D content representations (encoded in, e.g., ActionScript or X3D), and it may be combined with the approach to transformation of 3D content description formats proposed in [6]. Second, the approach should be evaluated and com-

pared to other solutions in terms of the effort in the implementation of multi-platform 3D content presentations. Third, the complexity of mappings and templates for different target languages—imperative and declarative—could be compared. Furthermore, a visual modelling tool supporting the semantic creation of 3D content can be developed. Finally, a persistent link between semantic and final 3D content representations can be proposed to provide real-time synchronization of the content.

References

1. Mendes, C.M., Drees, D.R., Silva, L., Bellon, O.R.: Interactive 3D visualization of natural and cultural assets. In: Proc. of the 2nd workshop on eHeritage and digital art preservation. Firenze, Italy (2010)
2. Celakovski, S., Davcev, D.: Multiplatform real-time rendering of MPEG-4 3D scenes with Microsoft XNA. In: ICT Innovations 2009, 2010, Part 2, pp. 337-344
3. Almer, A., Schnabel, T., Stelzl, H., Stieg, J., Luley, P.: A tourism information system for rural areas based on a multi platform concept. In: Proc. of the 6th int. conf. on Web and Wireless Geographical Information Systems, pp. 31-41. Hong Kong, China, December 4-5 (2006)
4. Tack, K., Lafruit, G., Catthoor, F., Lauwereins, R.: Platform independent optimisation of multi-resolution 3D content to enable universal media access. In: The Visual Computer, Volume 22, Issue 8, pp. 577-590, August (2006)
5. Han, J., Kang, I., Hyun, C., Woo, J., Eom, Y.: Multi-platform online game design and architecture. In: Proceedings of the 2005 IFIP TC13 international conference on Human-Computer Interaction, pp. 1116-1119. Springer-Verlag Berlin, Heidelberg (2005).
6. Flotyński, J., Dalkowski, J., Walczak, K.: Building multi-platform 3D virtual museum exhibitions with Flex-VR. In:Proceedings of the 18th International Conference on Virtual Systems and Multimedia, pp. 391-398. Milan, Italy, September 2-5 (2012)
7. Pittarello, F., Faveri, A.: Semantic description of 3D environments: a proposal based on web standards. In: Proceedings of the 11th international conference on 3D web technology, pp. 85-95, Columbia, USA, April 18-21 (2006)
8. Kalogerakis, E., Christodoulakis, S., Moumoutzis, N.: Coupling Ontologies with Graphics Content for Knowledge Driven Visualization. In: VR '06 Proceedings of the IEEE conference on Virtual Reality, pp. 43-50, Alexandria, Virginia, USA, March 25-29 (2006)
9. Walczak, K.: Flex-VR: Configurable 3D Web Applications. In: Proceedings of the International Conference on Human System Interaction HSI, pp. 135-140, Kraków, May 25-27 (2008) ISBN: 1-4244-1543-8
10. De Troyer, O., Kleinermann, F., Pellens, B., Bille, W.: Conceptual modeling for virtual reality. In: Tutorials, posters, panels and industrial contributions at the 26th international conference on Conceptual modeling - Volume 83, Australian Computer Society, Inc. Darlinghurst, pp. 3-18, Australia (2007), ISBN: 978-1-920682-64-4
11. Latoschik, M.E., Frohlich, C.: Semantic Reflection for Intelligent Virtual Environments. In: IEEE Virtual Reality Conference 2007. Charlotte, USA, March 10-14 (2007)
12. Flotyński, J., Walczak, K.: Semantic Multi-layered Design of Interactive 3D Presentations. In: Proceedings of the Federated Conference on Computer Science and Information Systems, Kraków, Poland, 8-11 September (2013)
13. Flotyński, J., Walczak, K.: Conceptual Semantic Representation of 3D Content. In: LNBIP, Volume 160, pp. 244-257, Springer-Verlag, Berlin Heidelberg (2013)
14. Flotyński, J., Walczak, K.: Semantic Modelling of Interactive 3D Content. In: Proc. of the 5th Joint Virtual Reality Conf. Paris, France, December 11-13 (2013), accepted for publication