Semantic Modeling of Virtual Reality Training Scenarios

Krzysztof Walczak[1(✉)], Jakub Flotyński[1], Dominik Strugała[1], Sergiusz Strykowski[1], Paweł Sobociński[1], Adam Gałązkiewicz[1], Filip Górski[2], Paweł Buń[2], Przemysław Zawadzki[2], Maciej Wielgus[1], and Rafał Wojciechowski[3]

[1] Poznań University of Economics and Business, Niepodległości 10, 61-875 Poznań, Poland
walczak@kti.ue.poznan.pl
[2] Poznań University of Technology, Piotrowo 3, 60-965 Poznań, Poland
filip.gorski@put.poznan.pl
[3] Enea Operator sp. z o.o., Strzeszyńska 58, 60-479 Poznań, Poland

Abstract. Virtual reality can be an effective tool for professional training, especially in the case of complex scenarios, which performed in reality may pose a high risk for the trainee. However, efficient use of VR in practical everyday training requires efficient and easy-to-use methods of designing complex interactive scenarios. In this paper, we propose a new method of creating virtual reality training scenarios, with the use of knowledge representation enabled by semantic web technologies. We have verified the method by implementing and demonstrating an easy-to-use desktop application for designing VR scenarios by domain experts.

Keywords: Virtual reality · Semantic web · Training · Scenarios

1 Introduction

Progress in the quality and the performance of graphics hardware and software observed in recent years makes realistic interactive presentation of complex virtual spaces and objects possible even on commodity hardware. The availability of diverse inexpensive presentation and interaction devices, such as glasses, headsets, haptic interfaces, motion tracking and capture systems, further contributes to the increasing applicability of virtual (VR) and augmented reality (AR) technologies. VR/AR applications become popular in various application domains, such as e-commerce, tourism, education and training. Especially in training, VR offers significant advantages by making the training process more efficient and flexible, reducing the costs, and eliminating risks associated with training in a physical environment.

Employee training in virtual reality is becoming widespread in various industrial sectors, such as production, mining, gas and energy. However, building useful VR training environments requires competencies in both programming and

P. Bourdot et al. (Eds.): EuroVR 2020, LNCS 12499, pp. 128–148, 2020.
https://doi.org/10.1007/978-3-030-62655-6_8

3D modeling, as well as domain knowledge, which is necessary to prepare practical applications in a given domain. Therefore, this process typically involves IT specialists and domain specialists, whose knowledge and skills in programming and 3D modeling are usually low. Particularly challenging is the design of training scenarios, as it typically requires advanced programming skills, and the level of code reuse in this process is low. High-level componentization approaches commonly used in today's content creation tools are not sufficient, because the required generality and versatility of these tools inevitably leads to a high complexity of the content design process. Availability of appropriate user-friendly tools for domain experts to design VR training scenarios at the level of domain knowledge becomes therefore critical to enable reduction of the required time and effort, and consequently promote the use of VR in training.

A number of solutions enabling efficient modeling of 3D content using domain knowledge representation techniques have been proposed in previous works. In particular, semantic web provides standardized mechanisms to describe the meaning of any content in a way understandable to both users and software. However, it requires that the scenarios are designed by a knowledge engineering technician, which is not acceptable in practical VR training preparation. Thus, the challenge is to elaborate a method of creating semantic VR scenarios, which could be employed by users who do not have advanced knowledge and skills in programming and 3D modeling.

In this paper, we propose a new method of building VR training scenarios, based on semantic modeling techniques, with a user-friendly *VR Scenario Editor* (VRSEd) application implemented as an extension to Microsoft Excel, a tool commonly used by people in various domains. The editor enables domain experts to design scenarios using domain concepts described by ontologies. The presented approach takes advantage of the fact that in a concrete training scene and typical training scenarios, the variety of 3D objects and actions is limited. Therefore, it becomes possible to use a semantic database of available content elements and actions, and configure scenarios based on the existing building blocks using domain-specific concepts.

The work described in this paper has been performed within a project aiming at the development of flexible VR training system for electrical operators. All examples, therefore, relate to this application domain. However, the developed method and tools can be similarly applied to other domains, provided that relevant 3D objects and actions can be identified and semantically described.

The remainder of this paper is structured as follows. Section 2 provides an overview of the current state of the art in VR training applications, an introduction to the semantic web, and a review of approaches to semantic modeling of VR content. Section 3 describes our method of building VR training scenes. The proposed method of modeling training scenarios is described in Sect. 4. An example of a VR training scenario is presented in Sect. 5, while a discussion of the results is provided in Sect. 6. Finally, Sect. 7 concludes the paper and indicates possible future research.

2 Related Works

2.1 Training in VR

VR training systems enable achieving a new quality in employee training. With the use of VR it becomes possible to digitally recreate real working conditions with a high level of fidelity. Currently available systems can be categorized into three main groups: desktop systems, semi-immersive systems and fully immersive systems. Desktop systems use mainly traditional presentation/interaction devices, such as a monitor, mouse and keyboard. Semi-immersive systems use advanced VR/AR devices for presentation (e.g., HMD) or for interaction (e.g., motion tracking). Immersive systems use advanced VR/AR devices for both presentation and interaction. Below, examples of VR training systems within all of the three categories are presented.

The ALEn3D system is a desktop system developed for the energy sector by the Virtual Reality group of the Control Systems [23]. The system allows interaction with 3D content displayed on a 2D monitor screen, using a mouse and a keyboard [31]. The scenarios implemented in the system mainly focus on training the operation of power lines and include actions performed by a line electrician. The system consists of two modules: a VR environment and a course manager [22]. The VR environment can operate in three modes: virtual catalog, learning and evaluation. The course manager is a browser application that allows trainers to create courses, register students, create theoretical tests and monitor learning progress.

An example of a semi-immersive system is the IMA-VR system [19]. It enables specialized training in a virtual environment aimed at transferring motor and cognitive skills related to the assembly and maintenance of industrial equipment. The system was designed by CEIT and TECNALIA. The specially designed IMA-VR hardware platform is used to work with the system. The platform consists of a screen displaying a 3D graphics scene and a haptic device. This device allows a trainee to interact and manipulate virtual scene tools and components by touching while performing assembly and disassembly operations. The system provides various types of information during training, including a progress bar, technical descriptions of components and tools, meaningful information about operations and detailed error descriptions. In addition to the visual and haptic presentation, the most important information is also sent via audio messages. The system automatically records completed tasks and statistics (time taken, number of assists used and errors made, number of correct steps, etc.).

An example of a fully immersive AR system is the training system for the repairing electrical switchboards developed by Schneider Electric in cooperation with MW PowerLab [35]. The system is used to conduct training in operation on electrical switchboards and replacement of their parts. The system uses Microsoft HoloLens HMD. After a user puts on the HMD, the system scans the surroundings for an electrical switchboard. When a switchboard is located in the user's field of view, the system displays its name and is ready for operation.

The system can work in two ways: providing tips on a specific problem to be solved or providing general tips on operating or repairing the switchboard.

2.2 Semantic Web

The semantic web (the term proposed by Tim Berners-Lee [29]) provides a universal framework that allows data to be shared and reused across application, enterprise, and community boundaries. According to the WWW Consortium, the semantic web is a web of structured data, decoupling applications from data through a simple, abstract model for knowledge representation.

The basis of the semantic web are ontologies [42]. Ontology is a formal specification of a conceptualization of a given field, including the concepts used in that field, as well as the relationships between these concepts. The purpose of an ontology is to define uniform terminology and interpretation of terms [36]. Ontologies are sets of expressions that must be clearly understood and must be suitable for automatic processing by computer programs. Ontology instructions can either define general concepts or describe specific objects and events associated with them. Overall, an ontology consists of elements representing two different types of knowledge – terminology and assertions. Terminology, referred to as TBox (terminological box), is a formal representation of the classes and properties of objects in a given field, as well as the relationships between these classes and properties [10]. Assertions, referred to as ABox (assertional box), refer to specific objects (individuals, instances) in a specific fragment of the modeled reality, described by classes and properties specified in the TBox.

In 3D modeling, ontologies consisting of TBox instructions (TBox ontologies) correspond to 3D scene templates [18]. For example, a TBox ontology can specify classes of exhibitions in a virtual museum, with various categories of artifacts, such as statues, stamps and coins, as well as spatial properties of the artifacts [16]. 3D scene templates can describe many 3D scenes. Ontologies consisting of ABox instructions (ABox ontologies) describe individual 3D scenes or elements of 3D scenes. For example, an ABox ontology can describe a specific exhibition with artifacts in a virtual museum that meet the conditions set out in the TBox ontology – they belong to individual classes and are described by specific property values.

The basic element of the semantic web used to build ontologies is the Resource Description Framework (RDF) [43]. RDF is a data model that enables the creation of so-called resource expressions. It enables to describe resources available on the internet in a way "understandable" for computers (easily processable by computer programs). The Resource Description Framework Schema (RDFS) [44] and the Web Ontology Language (OWL) [41] are languages for building statements in RDF-based ontologies and knowledge bases.

RDF enables describing resources with expressions consisting of three elements: subject (resource described in the instruction), predicate (subject's property) and object (value of the property describing the subject) [43]. RDF also introduces basic concepts for describing resources, such as data types, sets and lists. RDF can be used with various types of content: text, graphic, audio and

other documents. The RDFS and OWL standards extend RDF with the possibility of creating class hierarchies and properties, restrictions, properties of these restrictions and operations on sets. In turn, Semantic Web Rule Language (SWRL) extends OWL with rules.

2.3 Semantic Modeling of VR Content

A number of works have been devoted to ontology-based representation of 3D content, including a variety of geometrical, structural, spatial and presentational elements. A comprehensive review of the approaches has been presented in [18]. Existing methods are summarized in Table 1. Four of the 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.

Table 1. Comparison of semantic 3D content modeling methods

Approach	Level of abstraction	
	Low (3D graphics)	High (application domain)
De Troyer et al. [8,11,12,27,32]	✓	General
Gutiérrez et al. [20,21]	✓	Humanoids
Kalogerakis et al. [25]	✓	–
Spagnuolo et al. [2,3,34]	–	Humanoids
Floriani et al. [9,30]	✓	–
Kapahnke et al. [26]	–	General
Albrecht et al. [1]	–	Interior design
Latoschik et al. [14,28,46]	–	General
Drap et al. [13]	–	Archaeology
Trellet et al. [37,38]	–	Molecules
Perez-Gallardo et al. [33]	✓	–

The method proposed in [8,11,12,27,32] enables content creation at both the low and a high abstraction levels. Different 3D content ontologies connected by mapping are used at particular levels. 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. The approach also enables combination of primitive actions (e.g., move, turn, rotate, etc.) to complex behavior intelligible to end users without the knowledge of computer graphics.

The method proposed in [20,21] 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 indices) 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 proposed in [25] 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 does not enable mapping between high- and low-level concepts, so it is unsuitable for modeling 3D content by domain experts.

A method of creating 3D humanoids has been proposed in [2,3,34]. After automatic segmentation of 3D models, the identified 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.

The method proposed in [9,30] 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 [26] leverages semantic concepts, services and hybrid automata to describe objects' behavior 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 [1], 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.

In [14,28,46], a general-purpose tool and a method of 3D content creation has been described. The method is based on actors and entities, which represent

3D content at a high level. They are described by shared state variables and are subject to events. In particular, the approach can be used in game design.

In [13], 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 [40] can be used to select objects to build a 3D visualization.

In [37,38], 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 content types and tasks to be done.

In [33], 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 presented review indicates that there is a lack of a generic semantic method that could be used for creating interactive VR training scenarios in different application domains. The existing ontologies are either 3D-specific (with focus on static 3D content properties) or domain-specific (with focus on a single application domain). They lack domain-independent conceptualization of actions and interactions, which could be used by non-technical users in different domains to generate VR applications with limited help from graphics designers and programmers. In turn, the solutions focused on 3D content behavior, such as [15,17], use rules, which only to a limited extent fit the semantic web concept [40].

3 Building VR Training Scenes

3D VR training environments in our approach are created using a variety of hardware and software tools, including 3D laser scanners [45], CAD packages [6], 3D modeling software [5] and game engines [39], and are annotated using databases. The process consists of five main stages, as described below.

1. Physical elements of the training environment infrastructure are scanned into a polygon mesh. The mesh is encoded in STEP format [4] to enable further editing in subsequent stages. STEP is a standardized and widely used textual data format for CAD software (ISO 10303-21). It enables conversion of point clouds into CAD drawings without the risk of loosing relevant information. At this stage, the models contain no information about the hierarchy and semantics of particular elements.
2. Idealized and optimized 3D representations of the infrastructure elements are created using CAD software based on the scans. The main goal of this stage is to isolate groups of 3D model components, which will then be edited at the next stages. A designer performs the following steps: importing the STEP models into the CAD environment, dividing 3D models into components,

grouping 3D model components, and exporting the 3D models to DWG [7]. In case of the models presented in this paper, the AutoCAD environment was used due to its rich functionality and the ability to import non-standard file formats. An important advantage of this software is low CPU usage, which results in the possibility of editing complex objects. However, AutoCAD is primarily a design package and does not provide the functionality required for building visually appealing VR models.

3. 3D visual models of the infrastructure elements are created with the use of a 3D modeling environment. This stage aims to correct and optimize 3D models that contain unnecessary and repetitive elements and geometry defects. At this stage, the designer performs the following steps: importing 3D models into the modeling environment, removing repetitive components, correcting the geometry of 3D models, creating several LOD (Levels of Detail) for efficient rendering, and exporting the 3D models into the FBX format [24]. FBX is the primary 3D model format supported by game engines, in particular, the Unity 3D engine [39]. An example of a 3D modeling package, which can be used at this stage, is 3ds Max [5].
4. VR training scenes are assembled from the 3D visual models with the use of a 3D scene editor tool, built as an extension of a game engine IDE. The designer performs the following steps: importing 3D models saved in FBX format, creating a hierarchy of objects in the 3D scene, setting the spatial properties of 3D objects, configuring points and axes of rotation of 3D model components, assigning materials and textures to 3D models, and instantiating repetitive components of the scene model. The designer can use a 3D point cloud from the 3D scanning process as a reference in building the scene.
5. Databases of scene objects and equipment are created. The scene object database is specific to a particular VR training scene and describes the structure of objects (e.g., switchboard) and elements (e.g., switches and indicators) of the infrastructure that can be used in scenarios for the given scene. The database can be partially automatically generated based on the scene content and then extended by domain-experts using specifically designed interactive forms. Each element is associated possible states, in particular, boolean states (e.g., on, off), discrete states (e.g., gauge mode) and continuous states (e.g., voltage). The database of equipment includes elements shared by all scenarios, such as protective equipment and tools for performing particular types of works, and is created manually.

4 Modeling VR Training Scenarios

When the 3D model of a training scene together with the associated databases is available (cf. Sect. 3), the next important step is to design a training scenario. Scenario describes VR scene's behavior and interactivity, but is also a means of conveying the training material. Typically, programming scenarios is a complex and time-consuming process of writing scripts in a programming language supported by the game engine (C# in case of Unity 3D), which must be performed by a programmer.

Often, the same 3D training scene may be used with multiple different training scenarios. The scenarios may cover standard procedures to be performed or non-typical repair and maintenance actions. In each case, it should be possible to simulate malfunction of some of the elements to test subject's behavior in more complex situations. Moreover, simulation of different conditions (season, weather, daytime) or constraints (available equipment, time) may further improve the quality and versatility of the training process. Therefore, it becomes critical to permit the design of training scenarios in a quick and easy manner by domain experts without low-level programming in VR.

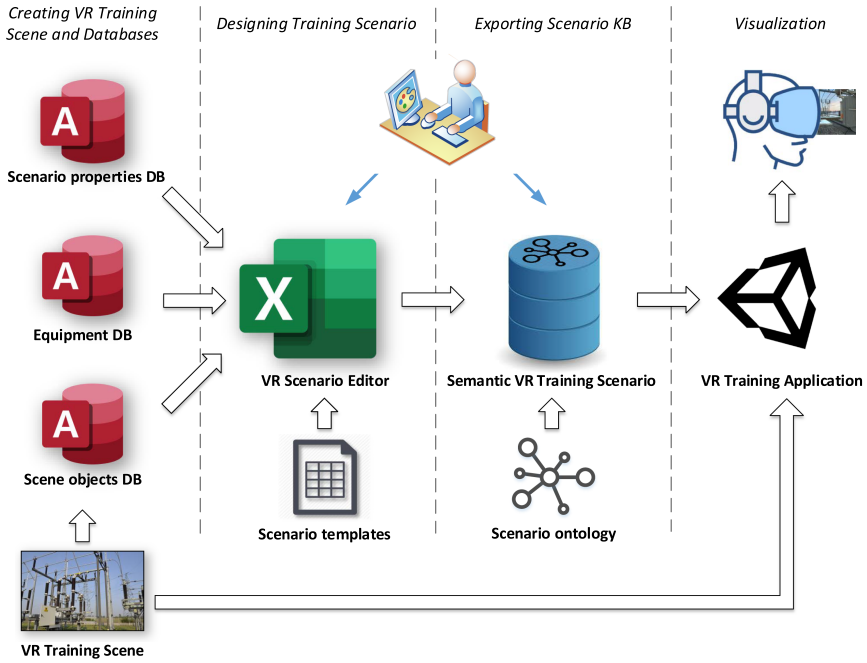

Fig. 1. Process of building a training scenario for a VR scene

4.1 Creating Training Scenarios

The process of creating a training scenario for a VR training scene in our approach is presented in Fig. 1. After a VR training scene and associated databases have been built, a training scenario can be created in three steps, as described below.

1. Designing a scenario using the VR Scenario Editor (VRSEd) application implemented as an extension to Microsoft Excel. The VRSEd editor provides several tools to support users in the design process. A scenario template is

imported to the editor. The template provides an overall structure and visual appearance of the scenario spreadsheet. Different templates may be used for different user groups or when the set of required attributes changes. Attribute values for selection lists in the scenario editor are retrieved from the scenario properties database. A scenario spreadsheet describes a sequence of training activities that should be executed by a single trainee, with possible trainee mistakes and system errors, e.g., lock the controller if possible, then switch off the transformer. The role of a trainer is reflected implicitly in the scenarios, by hints and comments shown to the trainee while training.

2. Exporting the scenario to a semantic VR scenario knowledge base, using a scenario exporter implemented in the VRSEd scenario editor. The knowledge base is encoded using RDF, RDFS and OWL standards and includes all information necessary for proper execution of the scenario in the VR Training Application. The knowledge base is an ABox compatible with the TBox scenario ontology. In other words, the scenario ontology specifies the terminology in the form of classes and properties, which is used in assertions specified in the scenario knowledge base. The generated scenario knowledge base consists of statements (RDF triples), which are counterparts to the particular rows of the scenario spreadsheet.
3. Importing the semantic VR scenario knowledge base into the VR Training Application implemented in the Unity 3D game engine. The scenario is loaded for a specific VR training scene described by the scenario. The correct assignment of the scene and the scenario is verified during the import. The knowledge base importer creates a hierarchy of script objects, which is then directly used during the scene runtime. One can import another scenario to the same VR training scene to provide different types of training.

4.2 Databases

The three databases shown in Fig. 1 provide all necessary data required by the VRSEd scenario editor to build a VR training scenario. All databases are currently implemented in Microsoft Access. In the next versions of the environment, in which remote access to databases will be required, the databases will be implemented in an SQL RDBMS.

Database of Scene Objects. For each VR training scene, a database of scene objects is created. The database contains 3 tables: *Objects*, *Elements* and *States*. The Objects table provides information about all infrastructure objects within the scene. The Elements table contains records corresponding to particular elements of objects, on which actions are performed or whose state depends on the user actions. The States table contains records representing all possible states of infrastructure objects' elements, on which actions can be performed.

Database of Equipment. The equipment database is common to all training scenarios created using the VRSEd scenario editor. The database contains

information about protective equipment (such as a protective visor, gloves or helmet) and specific work equipment, which may be required to perform particular tasks. The database contains also information how information about the need to use the particular kind of equipment should be presented to a user, and the representation of the equipment in a VR scene.

Database of Scenario Properties. In addition to data specific to a given virtual training scene, such as infrastructure objects, object elements and object states, the scenario editor must have access to all possible values of their properties. The list of values is common for all scenarios and is stored in the database of scenario properties. These values are included in the drop-down lists, when a user selects a property value. In particular, these are properties containing information on the types of work performed, types of equipment required for performing particular types of work, and types of protective equipment. The scenario properties database contains two tables: *Attributes* and *Fields*. The Attributes table contains attributes in the scenario tables that can be supplemented by selecting fields from drop-down lists. An example attribute is "Item mapping fidelity". The Fields table contains all possible values of attributes. Each attribute may be associated with multiple field values. Field values for "Item mapping fidelity" can be "High", "Medium", and "Low".

4.3 VR Scenario Editor Application

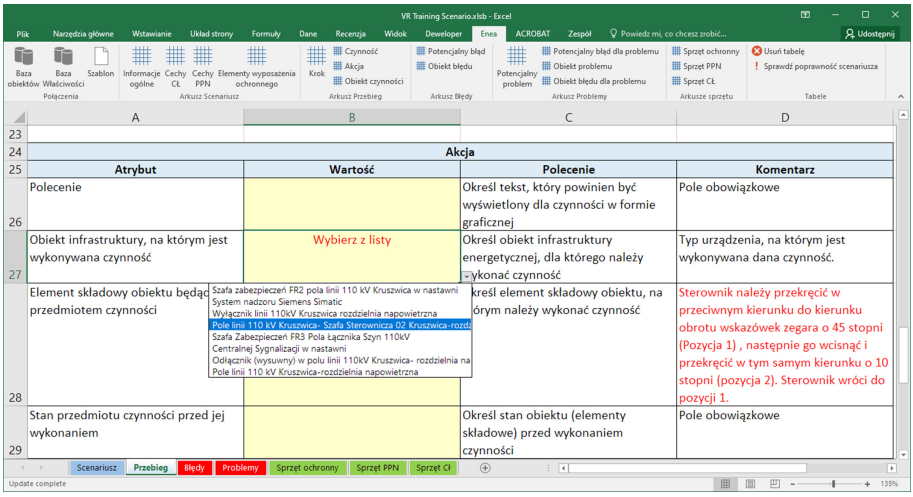

Fig. 2. The VR Scenario Editor implemented as extension to Microsoft Excel

The VR Scenario Editor (VRSEd) application has been implemented as an extension to Microsoft Excel (Fig. 2) to enable quick and efficient creation of training

scenarios by non-IT-specialists. The main advantages of using MS Excel as the editor implementation platform include the popularity of the software, which eliminates the need to install additional programs, the availability of numerous well-documented add-ons and programming libraries, as well as a wide community of users.

On top of the window (Fig. 2), the VRSEd toolbar is visible. It provides tools for importing databases, loading a scenario template, and creating different types of entities that may be used in a scenario (steps, activities, actions, objects, elements, problems, etc.).

Individual scenario sheets (visible on the bottom) contain descriptions of basic scenario properties, the course of the scenario, errors and problems that may occur in the scenario, and different types of equipment to be used. Each sheet consists of different types of tables. The Scenario sheet contains general information about the scenario and the required equipment.

The main sheet of the workbook provides information about the course of the scenario. It describes the training in the form of a logical sequence of *steps*, *activities*, and *actions* to be performed by a trainee. In each scenario, at least one step must be defined. Steps are divided into activities. Each activity can be associated with a problem that may occur during the training, an error that can be made, and the necessary equipment. Within an activity, the trainee performs actions on infrastructure objects that are described using the Action table (visible in Fig. 2). The Action table lists the *objects* and *elements* of the infrastructure, on which the action is carried out, the way their states change as a result of the action, and how this change in state is reflected in the training. The sequential form of scenarios is sufficient for a vast majority of training scenarios in the selected domain, with possible side threads represented by trainee's mistakes and system errors.

Various types of errors may be made during a training session, which are described in the Errors sheet. Each error is described by properties that specify how to inform the trainee about the error and how the system responds to the error. The Problems worksheet contains the table Potential problem showing events that may occur during the training that disrupt the work course. A problem may be associated with elements of objects in the scene that depend on the problem, i.e., their state changes. For any potential problem, the trainee may make an error when trying to resolve it.

The structure of scenario spreadsheets, which is based on tables, matches the structure of scenario knowledge bases based on RDF triples (cf. Sect. 2.2). When a scenario is exported into a semantic VR scenario knowledge base, every table is exported to multiple triples. The table identifier designates the subject of a triple (e.g., action), the attribute name in a particular row in the first column designates the predicate (e.g., infrastructure object id), and the attribute value in the row in the second column designates the predicate value (e.g., a particular object id selected from the list)—Fig. 2.

4.4 Ontology and Semantic Scenario Knowledge Base

A formal *scenario ontology* has been designed to enable semantic description of training scenarios in VR. The scenario ontology is a TBox, which specifies the classes and properties used to describe training scenarios (ABox), as well as relationships between these classes and properties. The scenario ontology has been implemented using the RDF, RDFS and OWL standards.

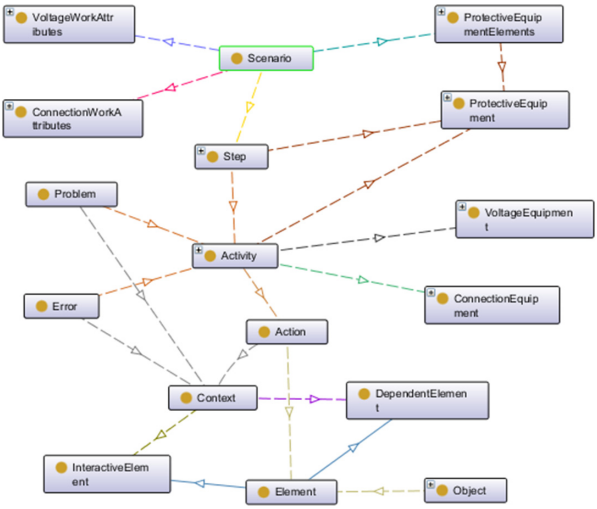

Fig. 3. Ontology of VR training scenarios

The entities specified in the scenario ontology, as well as the relations between them, are depicted in Fig. 3. The entities encompass classes (rectangles) and properties (arrows) that fall into three categories describing: the workflow of training scenarios, objects and elements of the infrastructure, and equipment necessary to execute actions on the infrastructure.

Every *scenario* is represented by an individual of the *Scenario* class. A scenario consists of at least one *Step*, which is the basic element of the workflow, which consists of at least one *Activity*. Steps and activities correspond to two levels of generalization of the tasks to be completed by training participants. Activities specify equipment required when performing the works. In the VR training environment, it can be presented as a toolkit, from which the user can select the necessary tools. Steps and activities may also specify protective equipment. *Actions*, which are grouped into activities, specify particular indivisible tasks completed using the equipment specified for the activity. Actions are executed on infrastructural components of two categories: *Objects* and *Elements*, which form two-level hierarchies. A technician, who executes an action, changes the *State* of an object's element (called *Interactive Element*), which may affect elements of this or other objects (called *Dependent Elements*). For example, a

control panel of a dashboard is used to switch on and off a transformer, which is announced on the panel and influences the infrastructure. N-ary relations between different entities in a scenario are represented by individuals of the *Context* class, e.g., associated actions, elements, and states. Non-typical situations in the workflow are modeled using *Errors* and *Problems*. While errors are due to the user, e.g., a skipped action on a controller, problems are due to the infrastructure, e.g., a controller's failure.

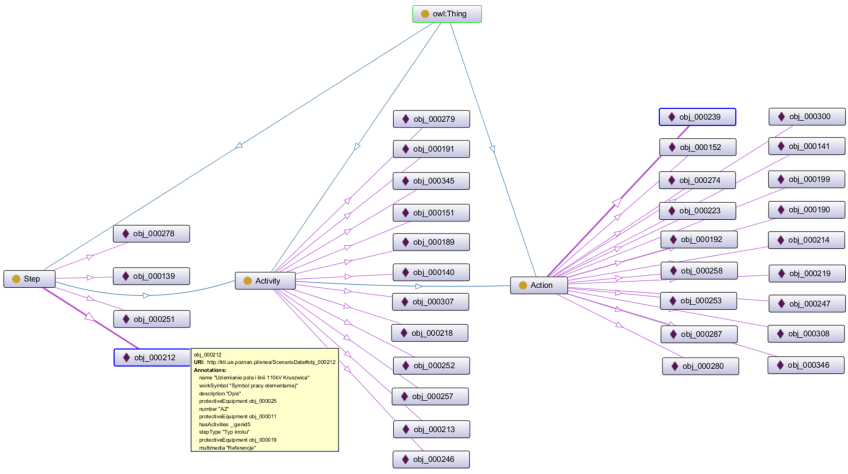

Fig. 4. VR training scenario described as a semantic knowledge base (fragment)

The scenario knowledge base is an ABox specifying a specific training scenario consisting of steps, activities and actions, along with its elements and infrastructure objects, which are described by classes and properties specified in the scenario ontology (Fig. 4). Scenario knowledge bases are encoded in OWL/Turtle. A scenario knowledge base is generated based on the scenario Excel workbook by the VRSEd KB exporter module. It is then imported into the VR Training Application by an importer module, which – based on the scenario KB – generates the equivalent object model of the scenario.

5 Example VR Training Scenario

A VR training scenario designed with VRSEd can be imported into the VR Training Application. In Fig. 5, a training scenario imported into the Unity 3D IDE is presented. On the left, the training scene is visible with scenario objects highlighted. On the right, three levels of the scenario, together with their properties, can be seen.

In Fig. 6 and 7, the VR Training Application, as seen by a trainee in the VR mode, is presented. The trainee can place hand in the walkie-talkie area and

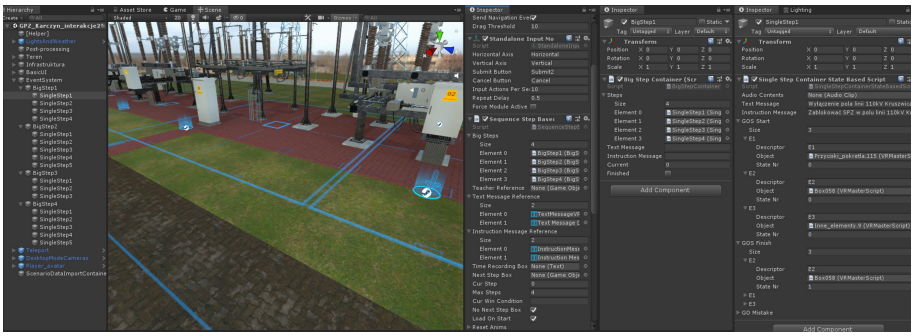

Fig. 5. Training scenario imported into Unity 3D IDE

Fig. 6. VR Training Application – execution of Step 1 Activity 1

press the controller button to display information about the step, activity and action to be performed (Fig. 8).

By changing the scenario in VRSEd, a designer can modify all attributes of any step, activity and action of the scenario. In Fig. 9, a scenario is presented that starts with the working area not properly marked. Marking of the work area is one of the mandatory steps before the real work can start. Therefore, the first step for the trainee will be to mark the area with an appropriate sign. In Fig. 10, the same scenario, but with the modified initial state, is presented. The working area is already clearly marked, which is visible in the model, and the trainee does not have to perform the action of marking. This change in VRSEd requires only selecting a different initial state from a list in one of the scenario rows. Performing this change directly in Unity 3D model and code would be a difficult and time-consuming operation.

Fig. 7. VR Training Application – execution of Step 1 Activity 2

Fig. 8. Avatar and walkie-talkie used to activate displaying of scenario commands

6 Results and Discussion

Training of employees in practical industrial environments requires the ability to design new and modify existing training scenarios efficiently. In practice, the number of scenarios is by far larger than the number of training scenes. In the case of training electrical operators of high-voltage installations, typically one 3D model of an electrical substation is associated with at least a dozen of different scenarios. These scenarios include learning daily maintenance operations, reactions to various problems that may occur in the installation as well as reactions to infrastructure malfunction.

The training scenarios are typically very complex. The “Karczyn” scenario used as an example in this paper covers only preparation for a specific maintenance work and consists of 4 steps, 11 activities, and 17 actions. For each action,

Fig. 9. Initial scenario state set to "Work Area Not Marked"

Fig. 10. Initial scenario state set to "Work Area Marked"

there are dependent objects (44 in case of this scenario). For each step, activity, action and object, the scenario provides specific attributes (9–10 for each item). For each attribute, the name, value, command and comment are provided. In total, the specification of the course of the scenario consists of 945 rows in Excel. In addition, there are 69 rows of specifications of errors and 146 rows of specification of problems. The scenario also covers protective equipment, specific work equipment, and others.

The generic scenario ontology (TBox) encoded in OWL takes 1,505 lines of code and 55,320 bytes in total. The "Karczyn" scenario saved in Turtle (which is a more efficient way of encoding ontologies and knowledge bases) has 2,930 lines of code and 209,139 bytes in total.

Implementation of the "Karczyn" scenario directly as a set of Unity 3D C# scripts would lead to very complex code, difficult to verify and maintain even by a highly-proficient programmer. The design of such a scenario is clearly beyond the capabilities of most domain experts dealing with everyday training of electrical workers.

The use of the VRSEd tool, together with a formal ontology described in this paper, enables concise representation of the scenario, and provides means of editing and verification of scenario correctness with a user-friendly and familiar tool. Moreover, by using different scenario templates, the tool can be customized for different user groups, providing branding, explanatory graphics, automation, hints and custom fields further simplifying the scenario design process.

An important aspect to consider is the size of the scenario representations. The total size of the "Karczyn" Unity 3D project is 58 GB, while the size of the executable version is only 1.8 GB. Storing 20 scenarios in editable form as Unity projects would require 1.16 TB of disk space. Storing 20 scenarios in the form of semantic knowledge bases requires only 4 MB of storage space (plus the size of the executable application). Such scenario representations can be easily exchanged over the network between different training sites.

The ability to save scenarios at any stage of their development in the form of Excel files further contributes to the increased solution's usability. The "Karczyn" scenario saved as an XLSX file requires 447 KB. One can easily create and store multiple versions of multiple scenarios on a typical laptop computer.

7 Conclusions and Future Works

The method of semantic modeling of VR training scenarios presented in this paper enables flexible and precise modeling of scenarios at a high level of abstraction using concepts specific to a particular application domain instead of forcing the designer to use low-level programming with techniques specific to computer graphics. The presented VRSEd editor, in turn, enables efficient creation and modification of the scenarios by domain experts. Hence, the method and the tool make the development of VR applications, which generally is a highly technical task, attainable to non-technical users allowing them to use in the design process concepts of their domains of interest.

Future works include several elements. First, the environment will be extended to support collaborative creation of scenarios by distributed users. It will require changing the document-based database implementation (currently with Microsoft Access) with a full relational SQL database management system supporting transactions and concurrent access of multiple users. Second, we plan to extend the training application to support not only the training mode, but also the verification mode of operation with appropriate scoring based on user's performance. Finally, we plan to extend the scenario ontology with concepts of parallel sequences of activities, which can be desirable for multi-user training, e.g., in firefighting.

Acknowledgments. The research work presented in this paper has been supported by the European Union from the European Regional Development Fund within the Smart Growth Operational Programme 2020–2024. The project is executed within the priority axis "Support for R&D Activity of Enterprises" of the National Centre for Research and Development under the contract POIR.01.01.01-00-0463/18.

References

1. Albrecht, S., Wiemann, T., Günther, M., Hertzberg, J.: Matching CAD object models in semantic mapping. In: Proceedings ICRA 2011 Workshop: Semantic Perception, Mapping and Exploration, SPME (2011)
2. Attene, M., Robbiano, F., Spagnuolo, M., Falcidieno, B.: Semantic annotation of 3D surface meshes based on feature characterization. In: Falcidieno, B., Spagnuolo, M., Avrithis, Y., Kompatsiaris, I., Buitelaar, P. (eds.) SAMT 2007. LNCS, vol. 4816, pp. 126–139. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-77051-0_15
3. Attene, M., Robbiano, F., Spagnuolo, M., Falcidieno, B.: Characterization of 3D shape parts for semantic annotation. Comput. Aided Des. **41**(10), 756–763 (2009). https://doi.org/10.1016/j.cad.2009.01.003
4. Autodesk: STEP (STP, STEP) Files (2019). https://knowledge.autodesk.com/support/3ds-max/learn-explore/caas/CloudHelp/cloudhelp/2020/ENU/3DSMax-Data-Exchange/files/GUID-B5F0FE98-B42C-48EC-AC94-0D1B25AD97F2-htm.html
5. Autodesk: 3ds Max (2020). https://www.autodesk.pl/products/3ds-max/overview
6. Autodesk: AutoCAD Civil 3D (2020). http://www.autodesk.com/products/autocad-civil-3d/overview
7. Autodesk: DWG Format (2020). https://www.autodesk.com/products/dwg
8. Bille, W., De Troyer, O., Pellens, B., Kleinermann, F.: 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, Belgium, pp. 17–26 (2005)
9. De Floriani, L., Hui, A., Papaleo, L., Huang, M., Hendler, J.: A semantic web environment for digital shapes understanding. In: Falcidieno, B., Spagnuolo, M., Avrithis, Y., Kompatsiaris, I., Buitelaar, P. (eds.) SAMT 2007. LNCS, vol. 4816, pp. 226–239. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-77051-0_25
10. De Giacomo, G., Lenzerini, M.: TBox and ABox reasoning in expressive description logics. In: Proceedings of the Fifth International Conference on the Principles of Knowledge Representation and Reasoning (KR 1996), vol. 1996, pp. 37–48 (1996)
11. De Troyer, O., Kleinermann, F., Mansouri, H., Pellens, B., Bille, W., Fomenko, V.: Developing semantic VR-shops for e-Commerce. Virtual Reality **11**(2–3), 89–106 (2007)
12. De Troyer, O., Kleinermann, F., Pellens, B., Bille, W.: Conceptual modeling for virtual reality. In: Grundy, J., Hartmann, S., Laender, A.H.F., Maciaszek, L., Roddick, J.F. (eds.) Tutorials, Posters, Panels and Industrial Contributions at the 26th International Conference on Conceptual Modeling - ER 2007. CRPIT, vol. 83, pp. 3–18. ACS, Auckland, New Zealand (2007)

13. Drap, P., Papini, O., Sourisseau, J.-C., Gambin, T.: Ontology-based photogrammetric survey in underwater archaeology. In: Blomqvist, E., Hose, K., Paulheim, H., Ławrynowicz, A., Ciravegna, F., Hartig, O. (eds.) ESWC 2017. LNCS, vol. 10577, pp. 3–6. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-70407-4_1
14. Fischbach, M., et al.: SiXton's curse - simulator X demonstration. In: Hirose, M., Lok, B., Majumder, A., Schmalstieg, D. (eds.) Virtual Reality Conference (VR), pp. 255–256. IEEE (2011). http://dx.doi.org/10.1109/VR.2011.5759495
15. Flotyński, J., Krzyszkowski, M., Walczak, K.: Semantic composition of 3D content behavior for explorable virtual reality applications. In: Barbic, J., D'Cruz, M., Latoschik, M.E., Slater, M., Bourdot, P. (eds.) EuroVR 2017. LNCS, vol. 10700, pp. 3–23. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-72323-5_1
16. Flotyński, J., Walczak, K.: Customization of 3D content with semantic meta-scenes. Graph. Models **88**, 23–39 (2016). https://doi.org/10.1016/j.gmod.2016.07.001
17. Flotyński, J., Walczak, K.: 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), 5–7 June 2017, p. Article No. 14. ACM, New York (2017). https://doi.org/10.1145/3055624.3075959
18. Flotyński, J., Walczak, K.: Ontology-based representation and modelling of synthetic 3D content: a state-of-the-art review. Comput. Graph. Forum **35**, 329–353 (2017). https://doi.org/10.1111/cgf.13083
19. Gavish, N., et al.: Evaluating virtual reality and augmented reality training for industrial maintenance and assembly tasks. Interact. Learn. Environ. **23**(6), 778–798 (2015). https://doi.org/10.1080/10494820.2013.815221
20. Gutiérrez, M.: Semantic virtual environments, EPFL (2005)
21. Gutiérrez, M., Thalmann, D., Vexo, F.: Semantic virtual environments with adaptive multimodal interfaces. In: Chen, Y.P.P. (ed.) MMM, pp. 277–283. IEEE Computer Society (2005)
22. Hoffman, H., Vu, D.: Virtual reality: teaching tool of the twenty-first century? Acad. Med.: J. Assoc. Am. Med. Coll. **72**(12), 1076–1081 (1997). https://doi.org/10.1097/00001888-199712000-00018
23. Instituto Nacional de Electricidad y Energías Limpias: Sala de prensa (2017). https://www.ineel.mx/detalle-de-la-nota.html?id=38
24. Jeong, T., Kim, Y.: A new lightweight file format based on FBX for efficient 3D graphics resource processing. J. Theor. Appl. Inf. Technol. **97**, 2393–2403 (2018)
25. Kalogerakis, E., Christodoulakis, S., Moumoutzis, N.: Coupling ontologies with graphics content for knowledge driven visualization. In: VR 2006 Proceedings of the IEEE Conference on Virtual Reality, Alexandria, Virginia, USA, pp. 43–50, March 2006
26. Kapahnke, P., Liedtke, P., Nesbigall, S., Warwas, S., Klusch, M.: ISReal: an open platform for semantic-based 3D simulations in the 3D internet. In: Patel-Schneider, P.F., et al. (eds.) ISWC 2010. LNCS, vol. 6497, pp. 161–176. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-17749-1_11
27. Kleinermann, F., De Troyer, O., Mansouri, H., Romero, R., Pellens, B., Bille, W.: Designing semantic virtual reality applications. In: Proceedings of the 2nd Intuition International Workshop, Senlis, pp. 5–10 (2005)
28. Latoschik, M.E., Tramberend, H.: Simulator X: a scalable and concurrent software platform for intelligent realtime interactive systems. In: Proceedings of the IEEE VR 2011 (2011)
29. Lugrin, J.L.: Alternative reality and causality in virtual environments. Ph.D. thesis, University of Teesside, Middlesbrough, United Kingdom (2009)

30. Papaleo, L., De Floriani, L., Hendler, J., Hui, A.: Towards a semantic web system for understanding real world representations. In: Proceedings of the Tenth International Conference on Computer Graphics and Artificial Intelligence (2007)
31. Patoni, R.: Alen 3D. https://www.scribd.com/document/245796121/ALEN-3D
32. Pellens, B., De Troyer, O., Bille, W., Kleinermann, F., Romero, R.: An ontology-driven approach for modeling behavior in virtual environments. In: Meersman, R., Tari, Z., Herrero, P. (eds.) OTM 2005. LNCS, vol. 3762, pp. 1215–1224. Springer, Heidelberg (2005). https://doi.org/10.1007/11575863_145
33. Perez-Gallardo, Y., Cuadrado, J.L.L., Crespo, Á.G., de Jesús, C.G.: GEODIM: a semantic model-based system for 3D recognition of industrial scenes. In: Alor-Hernández, G., Valencia-García, R. (eds.) Current Trends on Knowledge-Based Systems. ISRL, vol. 120, pp. 137–159. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-51905-0_7
34. Robbiano, F., Attene, M., Spagnuolo, M., Falcidieno, B.: Part-based annotation of virtual 3D shapes. In: 2013 International Conference on Cyberworlds, pp. 427–436 (2007)
35. Schneider Electric: HoloLens Application on Premset (2017). https://www.youtube.com/watch?v=RpXyagutoZg
36. Sikos, L.F.: Description Logics in Multimedia Reasoning. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-54066-5
37. Trellet, M., Ferey, N., Baaden, M., Bourdot, P.: 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), pp. 48–53. IEEE (2016)
38. Trellet, M., Férey, N., Flotyński, J., Baaden, M., Bourdot, P.: Semantics for an integrative and immersive pipeline combining visualization and analysis of molecular data. J. Integr. Bioinform. **15**(2), 1–19 (2018)
39. Unity Technologies: Unity (2020). http://unity.com/
40. W3C: SWRL: A Semantic Web Rule Language Combining OWL and RuleML (2004). http://www.w3.org/Submission/SWRL/
41. W3C: OWL 2 Web Ontology Language Structural Specification and Functional-Style Syntax (Second Edition) (2012). https://www.w3.org/TR/owl2-syntax/
42. W3C: Building the Web of Data (2013). http://www.w3.org/2013/data/
43. W3C: RDF 1.1 Concepts and Abstract Syntax (2014). https://www.w3.org/TR/rdf11-concepts/
44. W3C: RDF Schema 1.1 (2014). https://www.w3.org/TR/rdf-schema/
45. Walczak, K., Flotyński, J.: Inference-based creation of synthetic 3D content with ontologies. Multimedia Tools Appl. **78**(9), 12607–12638 (2019). https://doi.org/10.1007/s11042-018-6788-5
46. Wiebusch, D., Latoschik, M.E.: 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 (2012)