

Designing Behaviour-rich Interactive Virtual Museum Exhibitions

Krzysztof Walczak and Wojciech R. Wiza

Department of Information Technology
The Poznan University of Economics, Poland

Abstract

In this paper a new method of modelling and dynamic composition of behaviour-rich interactive 3D virtual museum exhibitions is described. The method enables museum experts, without advanced knowledge in computer programming, to create 3D virtual exhibitions accessible both locally inside the museums and remotely over the Internet. The method uses a novel Beh-VR modelling approach, which is compatible with standards such as X3D and VRML. The method has been successfully applied in several museums. Examples of interactive 3D exhibitions are provided.

Categories and Subject Descriptors (according to ACM CCS): H.5.1 [Information Interfaces and Presentation]: Multimedia Information Systems - Artificial, augmented, and virtual realities; I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism; K.3.1 [Computers and Education]: Computer Uses in Education

1. Introduction

Interactive 3D technologies have reached the level of maturity that makes possible their use in a diversity of real life applications. Cultural heritage is a notable example of such an application domain. With the popularisation of 3D technologies, museums start to create digital surrogates of their physical artefacts. Such surrogates can be presented both locally inside the museum and remotely over the Internet, can be displayed in their reconstructed original environments, can be easily exchanged between museums, while the original artefact may remain safe in a museum storage.

However, 3D technology has much bigger potential for virtual museum exhibitions than just enabling creation of "digital copies" of artefacts. The real benefits come from the interactivity and the behaviour of objects and exhibitions. With the help of interactive 3D techniques, every museum exhibition can be transformed into an interesting and educational experience. Young people, accustomed to highly interactive 3D games, can easily accept such form of presentation. This may be a good opportunity for cultural heritage institutions to attract more visitors – especially young generations of visitors.

Creation of complex high-quality 3D contents is one of the most important challenges in the domain of virtual reality. Virtual exhibitions, to be successful, must be interactive and engaging, must be of high quality and, most importantly, must be created and managed by museum staff, such as curators or exhibition designers. However, creating complex interactive 3D contents is not an easy task. Using traditional techniques it requires programming and designing skills that go much beyond the capabilities of average museum expert. On the other hand museums cannot depend on programmers in the process of creating exhibitions. Such requirement would render the process much more time consuming and expensive.

A number of solutions exist for *programming* behaviour of fixed 3D scenes. However, in most practical applications such predefined scenes are not sufficient. Real-life applications require the contents to be created *dynamically*, i.e. instead of fixed programmed scenes, dynamically generated contents are used. Such contents may depend on data provided by a content designer (e.g., a museum expert designing the exhibition) or an end-user (e.g., a museum visitor searching for interesting objects).

In this paper, we propose a novel approach to modelling and dynamic composition of behaviour-rich 3D virtual museum exhibitions. In this approach, called Beh-VR, virtual exhibitions are built of specifically designed objects, called VR-Beans, that can be arbitrarily composed into complex 3D scenes. The presented approach enables splitting the content creation process into several independent stages that can be performed by users with different sets of skills.

The rest of this paper is organized as follows. In Section 2 the current state of the art in the domain of multimedia interfaces to museum information systems is presented. In Section 3 the Beh-VR approach to dynamic modelling of behaviour-rich VR contents is described. In Section 4 dynamic creation of VR contents is discussed. Section 5 provides examples how the Beh-VR approach can be applied to build virtual museum exhibitions. Finally, Section 6 concludes the paper and indicates future works.

2. State of the Art

Modern museums exploit multimedia technologies to attract visitors – both visitors coming to the museum in person and visitors on the Internet – browsing museum web sites and virtual exhibitions.

For the first type of visitors museums prepare on-site interfaces, such as kiosks, permitting them to browse through museum's digital collections and learn about artefacts. In most cases these interfaces take advantage of touch screen displays, while their content is programmed using technologies such as MS PowerPoint, PDF, HTML and Flash. The first two are very simple, but they lack real interactivity and complex behaviour.

Use of HTML, displayed through a web browser, is justified by quick and easy implementation of such an interface. HTML browsers and editors are available free of charge, and HTML is reasonably simple to learn. Inexperienced users may create content using sophisticated WYSIWYG editors. However, simplicity of this solution is double-edged: presentations created with HTML have limited interactivity and complex content is often browser-specific. Creation of highly interactive content (using technologies such as JavaScript, ActiveX, AJAX) usually exceeds capabilities of non-IT experts. Most importantly, HTML permits only presentation of 2D content, which is not sufficient for building engaging virtual exhibitions.

The most common technology currently used for creation of kiosk interfaces is Macromedia Flash. Flash technology permits creation of eye-pleasant, graphically-rich, interactive presentations. With the use of Flash technology sophisticated educational scenarios may be presented. The main disadvantage of Flash is its complexity. Creation of an interesting virtual exhibition requires a lot of work and experienced programmers. Very often the resulting presentations, while spectacular, do not offer easy customization, data dynamism,

and further expansion capabilities [Mus07]. Another problem is weak support for 3D content within Flash presentations.

Some museums invest in building custom interfaces [Wis02, WV05, Tys03]. Such solutions permit creation of interactive interfaces and use of user-friendly authoring environments. However, these interfaces are mostly focused on 2D not enabling 3D contents presentation, produce contents which is non-standard and therefore may be used only on displays inside the museum. Additionally, such contents cannot be exchanged between museums that use different IT systems.

The second type of visitors, that museums want to attract, are the visitors that come to museum web sites from the Internet. In this realm, use of HTML technology is prevailing, while Flash interfaces, also used, are less frequent [Bri07, Gug07].

The most promising form of presenting virtual exhibitions is interactive 3D (in some cases called VR). Museums start to apply 3D technologies in their interfaces accessible both inside the museums and on the web. Most frequently interactive photographs in form of QuickTime VR are used, however true 3D standards, such as X3D and VRML, become also increasingly popular. In some cases, custom complex interfaces prepared with the use of 3D authoring tools (e.g., 3ds max and Virtools) [3ds07, Vir07] are used.

X3D/VRML standards are particularly interesting because they are not bound to any specific platform, enable presentation of the contents both inside the museum and over the Internet. They also offer true 3D interaction enabling a user to freely navigate in a 3D space and manipulate the objects. Creation of static X3D/VRML contents is relatively simple as most modelling tools and packages offer export to these formats.

The most important problem related to the use X3D/VRML for creating museum exhibitions is the creation of interactive and behaviour-rich 3D contents. Programming content behaviour using JavaScript and ROUTEs is time consuming, error-prone, and requires advanced knowledge in 3D programming.

3. The Beh-VR Approach

3.1. Beh-VR Overview

The *Beh-VR* approach enables modelling behaviour of dynamically created virtual reality contents. Beh-VR scenes can be dynamically composed by selecting sets of virtual objects. The virtual objects, called *VR-Beans*, are constructed according to some specific rules. Each VR-Bean is associated with a scenario that governs its appearance and behaviour in a virtual scene. These scenarios specify what actions are performed by objects either at specific points in time or as a result of user actions or interactions between

objects. The Beh-VR approach is based on standard technologies and is fully compatible with X3D/VRML formats.

The Beh-VR approach supports dynamic content composition by clearly identifying high-level independent objects that constitute a virtual scene. Such objects can be easily combined into virtual scenes without the need to perform low-level 3D graphics design or programming. Therefore it is possible to build user-friendly authoring tools oriented on domain-specific contents. This characteristic of Beh-VR makes it particularly well suited for building VR applications in domains such as cultural heritage and education, where the contents should be developed by domain experts and not computer graphics specialists.

3.2. VR-Beans

VR-Beans are independent reusable VR objects encoded in a standard content description language (such as VRML/X3D) but constructed according to some specific rules. Due to the use of these rules, VR-Beans can be automatically incorporated in a Beh-VR scene and can communicate with the scene and with each other. VR-Beans are analogous to high-level programming elements or widgets commonly used in modern programming languages.

The main element controlling each VR-Bean is a *scenario script*. Scenario scripts are encoded in *VR-BML* (Virtual Reality Behaviour Modelling Language) – a specifically designed high-level XML-based language [Wal06]. Each script contains specification of behaviour of a single VR-Bean object. It describes what happens when the object is initialized, what actions are performed by the object and what are responses of the object to external stimuli. Each behaviour script may create any number of scene components, which are geometrical or aural manifestations of the VR-Bean in a virtual scene, but there may be also VR-Beans that do not directly manifest themselves.

Implementation details of VR-Beans are platform specific (e.g., the content description language and the programming language). In the examples presented in this paper we use VRML/X3D for content description and Java for scenario script interpretation.

3.3. VR-BML Scenarios

A scenario script consists of VR-BML *commands*. VR-BML uses a hybrid approach based on both declarative programming for high-level elements (e.g., event actions) and imperative programming for low-level elements (e.g., algorithm details). This hybrid approach enables VR-BML to take best of the two worlds enabling a programmer to concentrate on important elements, and leave the common elements to the Beh-VR framework.

A scenario script may contain three main sections: the *initialize* section, which describes what happens when the

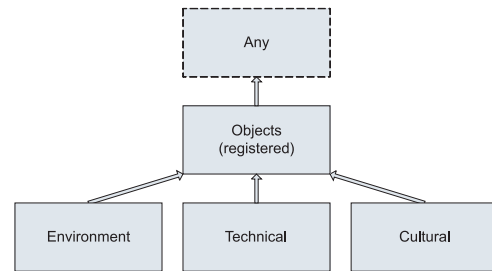


Figure 1: Example of a simple categorisation tree

object appears in the scene, a number of *action* statements, which describe what actions are performed by the object as a result of changes or events in the scene (time, scene properties, user interactions), and a number of *methods* that can be explicitly called by other objects in the scene.

3.4. Identification of VR-Beans

Beh-VR scenes are composed of dynamically selected independent VR-Beans. Therefore, there must be a mechanism to identify VR-Beans that are currently in the scene and their roles to allow the scene components to communicate. In Beh-VR this is accomplished by the process of *registration* and *discovery*.

Each VR-Bean can register itself in the scene in an arbitrary number of hierarchically organized dynamic categories. In addition, each VR-Bean is automatically assigned to the category "any". The tree of categories is built dynamically as the objects assign themselves to the categories. There are no constraints on the structure of the tree, and the tree itself does not define semantics of the categories. Ontology associated with the categories is application dependent. Sample tree of categories used in a virtual museum exhibition is presented in Fig. 1.

For example, a VR-Bean responsible for a presentation scenario in a dynamic virtual reality exhibition may retrieve all objects in the "objects/cultural" category, sort them by their age (retrieved from object metadata) and display them sequentially accompanied by an audio description. The object may register itself in another category "objects/technical".

3.5. Communication Between VR-Beans

Since the Beh-VR scenes are dynamically composed of independent VR-Bean objects, a crucial element are communication mechanisms between the objects. There are three communication mechanisms in the Beh-VR approach: *public values*, *assigned values*, and *method invocation*. Each of these mechanisms may be efficiently used for different purposes.

Public values are named public expressions that can use variables and events from a single VR-Bean. Communication via public values corresponds to broadcasting values. A VR-Bean executes (once) a *publish* command and the public value remains active until the end of the simulation or until an object publishes a different value with the same name. Since the pool of public values is global, only one object can publish a value of a given name at a given time. The characteristics of public values render them most useful for implementing virtual scene control elements. For example, in a virtual museum, a slider that allows users to "move in time" between centuries publishes a value "historical_date". All cultural objects assigned to this room read this value and appear in the scene or disappear to reflect the periods in time when they were used in the past.

In many cases, public values do not have to be processed by VR-BML scripts, but may be directly assigned to events controlling scene components. To achieve this, the *assigned values* mechanism can be used. Assigned values are expressions assigned to input events of scene components. Again, assignment is performed once by the use of a single command. Since the assignment, whenever the value of the expression changes (e.g., as a result of a changing public value), an event is being sent to the assigned field. Assigned values are convenient when one or more elements have direct influence on selected objects in the scene. Continuing the example of a virtual museum, assigned values can be used to control the scale of cultural objects in a virtual exhibition space. When a user changes the scale factor, the scale is automatically applied to all cultural objects without any additional conditions or processing.

As opposed to the first two methods of communication, the third one – *method invocation* – is explicit. A VR-Bean script may invoke methods of other VR-Beans. A method consists of VR-BML commands, which may change the state of the VR-Bean, alter its representation in the virtual scene, invoke other methods, etc. A method can be invoked on a single VR-Bean or on a whole list of VR-Beans. Such list may be retrieved from the pool of registered objects (e.g., all objects in a category) or constructed in any other way. Each method may have any number of parameters. Specification of formal parameters is provided in the method declaration, while actual parameter values are set in the method call. A method invocation may be either synchronous or asynchronous. Synchronous method invocation blocks the invoking script until the method finishes execution, asynchronous method invocation does not block the invoking script. Figure 2 provides a summary of the VR-Bean communication mechanisms supported by Beh-VR.

3.6. VR Design Patterns

Dynamic modelling of interactive behaviour-rich virtual reality contents requires imposing some kind of known semantic structure on the contents. For example, a virtual exhibi-

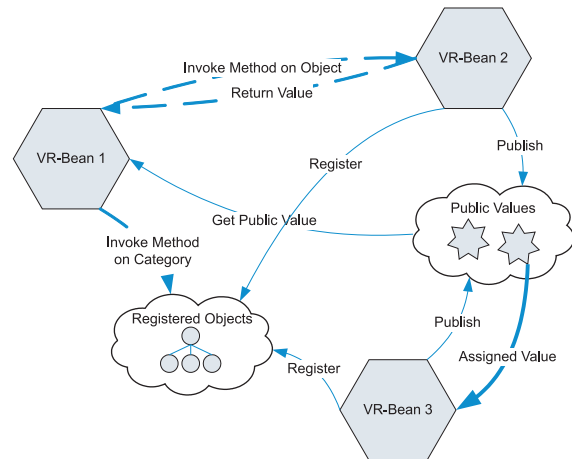


Figure 2: Communication between VR-Beans

tion may consist of a reconstruction of museum interior and a set of artefact models. A designer can select different room models and different sets of artefact models to setup different exhibitions. The process may be performed by the designer using a simple application or can be even automatic - can be performed by some software in response to a user query. Without such a known structure, a VR application is a single piece of code without semantics, which is difficult to develop, maintain and – in particular – to dynamically compose.

To solve this problem the notion of *VR design patterns* has been introduced. The term design pattern is commonly used in software engineering and means a "general repeatable solution to a commonly occurring problem in software design". Per analogy, a VR design pattern is a repeatable solution to a commonly occurring problem in VR design. The VR design patterns that can be applied to build virtual museum exhibitions include:

- Scene
- Template - Environment
- Template - Environment - Objects
- Template - Environment - Reactive Objects
- Template - Environment - Scenario - Reactive Objects
- Template - Environment - Scenario - Autonomous Objects

The first VR design pattern – *Scene* – is the simplest way of implementing a VR application. With this pattern a VR application is simply a model of a VR scene. This model may consist of geometrical objects, interaction elements, scripts that govern object behaviour, etc. The scene is created once (e.g., using a graphical authoring tool such as 3ds max [3ds07] and Maya [may07]). In some cases more complex object behaviour may be added to the scene using behaviour programming packages such as Virtools [Vir07].

This approach is common in today's VR systems. Typically, in museums, such 3D scenes consist of scanned artefacts or 3D reconstructions of interiors (e.g., museum rooms). However, this approach has serious limitations following the fact that the content is prepared once and it is stored in its final form. Neither the exhibition designer, museum guide nor end-user can influence the presented content or the form of presentation.

The second VR design pattern – *Template* – is the simplest pattern that is sufficient to build dynamic VR applications. With this pattern applications are modelled using one or more parameterised templates. The final VR scenes, which are presented to users, are dynamically generated based on the templates and sets of parameter values. The parameters can influence the structure of a scene, its geometry and behaviour [WC03].

The next VR design pattern – *Template - Environment - Objects* – is more powerful because it provides a clear distinction between the *template*, which is responsible for generation of the scene – its structure, contents and behaviour, and the *environment* and the *objects* – which are separate content elements. Using the same template and different environments and sets of objects it is possible to create different virtual scenes. The selection of objects can be performed by the exhibition designer, museum guide or even an end-user. This design pattern formed the foundation of the ARCO Virtual Museums System [ARC07].

The design pattern – *Template - Environment - Reactive Objects* – is similar to the previous one, with the difference that the objects can exhibit specific reactive behaviour to events generated in the virtual scene. For example, a model of a chest may open when clicked, a model of a vase may reproduce sound when hit, etc. The major difference in comparison to the previous pattern, is that the behaviour is object-specific and not programmed in the template.

The next design pattern – *Template - Environment - Scenario - Reactive Objects* – is an extension of the previous pattern. In addition, a scenario is added – typically another VR-Bean – which is responsible for generating events controlling the exhibition exploration process and stimulating particular objects. For example, a scenario may guide a user from one object to another, start voice descriptions for each object and activate objects, so they can demonstrate how they functioned. This design pattern is very powerful and is sufficient for most cases of virtual museum exhibitions.

The last pattern – *Template - Environment - Scenario - Autonomous Objects* – is the most advanced solution for demanding 3D exhibitions. In this pattern, objects may exhibit their own behaviour driven by time, virtual exhibition status and interactions with other objects. In this pattern, scenario guides the exhibition, but particular course of presentation may differ, depending on the set of objects currently in display, user behaviour, etc. As an example, when a user spends

more time observing objects of a particular style, other objects of the same style may activate themselves to attract user's attention.

4. Dynamic VR Contents

4.1. Dynamic Content Composition

In order to enable museum experts to create virtual exhibitions and museum visitors to adjust the exhibitions to their requirements, the content must be dynamically composed. The dynamic composition in our solution is based on the X-VR approach, which consists of a database model for VR, called X-VRDB, and dynamic modelling language, called X-VRML, that adds dynamic modelling capabilities to VR content description standards such as VRML and X3D [WC03].

In the presented approach, the X-VRML templates are used to dynamically generate VR contents based on a structure of VR presentations stored in a database. The use of presentation templates enables separation of the process of designing complex virtual scenes and programming object behaviour from the process of designing actual contents, allowing the latter to be easily performed by domain experts without extensive knowledge in computer science and 3D technologies.

4.2. Presentation Structure

The structure of dynamic VR presentations is determined by a hierarchy of presentation spaces stored in a content database (see Fig. 3 - top). The presentation spaces are conceptually similar to folders that may contain three types of elements: *instances of content templates*, *instances of behaviour templates*, and *instances of content objects*.

A template instance is a template supplied with actual values of some of its formal parameters. A single template can have an arbitrary number of instances in different spaces. A content object instance is a content object together with values of object presentation parameters. Again, the same object may have instances in more than one presentation space. The template parameters and content object parameters can be provided by a content designer while setting up a presentation. Different presentations can be achieved by the creation of template instances derived from the same template but supplied with different sets of parameter values. This concerns both the content templates and the behaviour templates. In some presentations, parameters that are not fixed by content designer can be changed by an end-user.

Content templates are used to dynamically compose virtual scenes. Simplest templates generate scenes by creating VR-Beans corresponding to content objects. More complex templates may additionally include background elements such as a model of a room. Each of the content objects may include its own VR-BML scenario script. In some

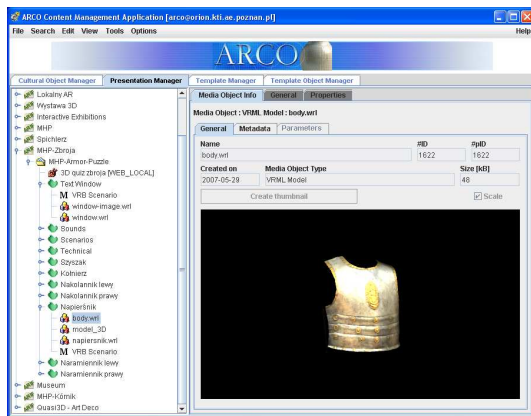
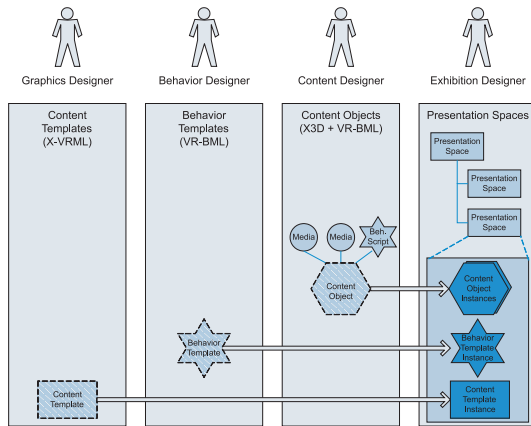


Figure 3: Database model of dynamic VR presentations (top) and ARCO Presentation Manager for designing virtual exhibitions (bottom).

cases, however, it is useful to have the same (or similar) behaviour shared by a number of objects. To achieve this, an instance of a behaviour template, i.e. an X-VRML template of VR-BML scripts, implementing common object behaviour may be also included in a presentation space.

A content designer can create an interactive presentation by simply collecting content objects, setting their visualization and behavioural parameters, and creating instances of templates. The process of designing complex interactive VR presentations can be performed by the use of a simple application connected to the content database (see Figure 3 - bottom) [WW05].

5. Using Beh-VR for Virtual Museums

The Beh-VR concept has been applied in the *ARCO Virtual Museum System* [ARC07] to create a set of interactive exhibitions for real museums. Below, two sample interfaces are described: *Kórnik Castle Virtual Tour* and *Armour Puzzle Game*.

Kórnik Castle, a small castle near Poznan in western part of Poland, is famous for its Moorish Hall styled after the Alhambra Palace. The Hall contains cultural objects commemorating Polish-Swedish War in XVII century. Virtual exhibition interface has been constructed with a high definition digital model of the Moorish Hall where a user may take a guided tour around the 3D environment (see Figure 4).

During the tour a visitor may observe architectural details of the Hall and listen to a video guide presenting historical background of the war. During the tour, navigation stops from time to time to enable the visitor to watch clips taken from the "Potop (The Deluge)" movie – feature film about the war.

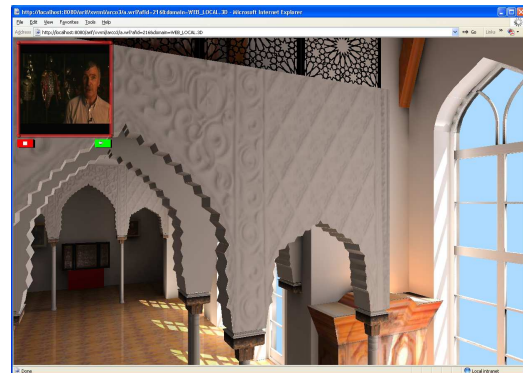
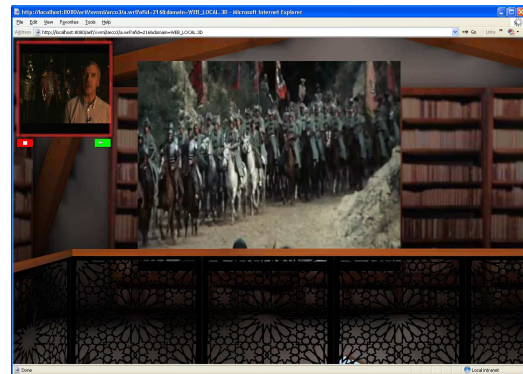


Figure 4: Kórnik Castle virtual tour interface.

Virtual tour interface consists of the digital model of the Moorish Hall, a Camera performing a virtual tour, camera-fixed screen with the video guide and three Screens positioned within the Hall presenting movie clips. The interface is based on the *Template - Environment - Scenario - Reactive Objects* VR design pattern (see Section 3.6). All components, except the Hall 3D environment – which is embedded in the visualisation template – are implemented as VR-Beans.

The main scenario, which is assigned to the Camera object, describes movements of the camera around the Hall.

```

<Scenario version="1">
  <Initialize>
    <Load file="camera.wrl" co="cam"
      pos="0,2,0" active="false"/>
    <Activate co="cam" active="true"/>
    <Register name="cam" category="obj/tech"/>
  </Initialize>

  <Action cond="{true}" t="135000" cnt="10000">
    <RotateTo co="cam" ang="1.57" axis="0,1,0"
      time="5500" wait="{false}" dir="right"/>
    <MoveTo co="cam" tgt="9,3.5,2"
      time="5500" wait="{true}"/>

    <Call ob="scr_1" mtd="slon()" wait="{true}"/>

    <RotateTo co="cam" ang="{4.71}" axis="0,1,0"
      time="6000" wait="{false}" dir="left"/>
    <MoveTo co="cam" tgt="8,5,1"
      time="8000" wait="{true}"/>
    ...
  </Action>
</Scenario>

```

Figure 5: Scenario assigned to the Camera Beh-VR object

When the camera approaches a position of a movie Screen, the Screen appears in the user's viewport and a user can watch the movie clip. After a scenario-defined period of time, the Screen hides and a user is automatically moved to the next position, in the meantime examining the Hall and listening to the guide. When the last movie clip finishes, the user is conveyed back to the starting point and the scenario loops. Using controls located under the video guide screen, the user may turn the guide on and off.

An excerpt of the scenario code is presented in Figure 5. The code is composed of an `<Initialize>` section and an `<Action>` section. Within the `<Initialize>` section the VRML model of the camera is loaded into the scene. The model contains visual representation of the camera, viewpoint bound to the camera object, and fixed screen with the guide clip. The camera object is then activated and the initialization finishes with registration of the Beh-VR component. Movement of the viewpoint is performed through the camera object animated using the Beh-VR script defined in the `<Action>` section.

The `<Action>` statement is executed repeatedly every 135 seconds. The `<Action>` command finishes after 10000 iterations. The camera object (together with the associated viewpoint) is animated through a number of `<RotateTo>` and `<MoveTo>` commands. The `<RotateTo>` command turns the component by the angle specified by the `ang` and `dir` attributes around the axis specified in the `axis` attribute. The `time` attribute specifies how long the rotation lasts, while the `wait` attribute specifies if the execution of

```

<Scenario version="1">
  ...
  <Method name="slon">
    <MoveTo co="scr" tgt="-6,2,0"
      time="2000" wait="{true}"/>
    <Wait time="30000"/>
    <MoveTo co="scr" tgt="-6,-10,0"
      time="2000" wait="{true}"/>
  </Method>
  ...
</Scenario>

```

Figure 6: Scenario assigned to the Screen Beh-VR object

the next command should wait until execution of the current command finishes. The `<MoveTo>` command animates a component to a position defined in the `tgt` attribute. When the Camera component is approaching assigned point, the script calls a remote method (`slon()`) defined in the Screen object. The method is called by the `<Call>` command. The `<Call>` command indicates the target object and method invocation. Declaration of the method is presented in Figure 6.

Another example of an interactive exhibition is Armour Puzzle Game (see Figure 7). Using this interface a user plays a game placing particular armour elements in correct places on the body skeleton. Each part must pass through a predefined path, therefore elements must be placed in a specific order. If a part is placed in a correct location, it cannot be moved any more – any other part that should be placed further along the path will be blocked and the game must be restarted. Such scenario imposes higher level of difficulty.

The game has been designed to be used on a touch screen display in a museum stand. In most cases touch screens are much more sensitive to touch ("click" or "double click") than to drag gestures. Therefore, an interface designed to be used on a touch screen, should implement all possible interactions using point-and-touch paradigm, rather than point-and-drag paradigm. The interface is based on the *Template - Environment - Scenario - Autonomous Objects* VR design pattern (see Section 3.6). All game elements are implemented as VR-Beans. Scenario of the game is defined as a separate object. The scenario defines game rules as well as positions and movements of parts of the armour. When a user touches an armour part, the part is moved from one place to another by calling appropriate method in the armour part behaviour scenario. Such an approach permits every object to behave in a different way (e.g., include rotation during the movement). Additionally, objects exhibit autonomous behaviour (animation, sound), which depends on their location and state, making the game more dynamic and interesting.

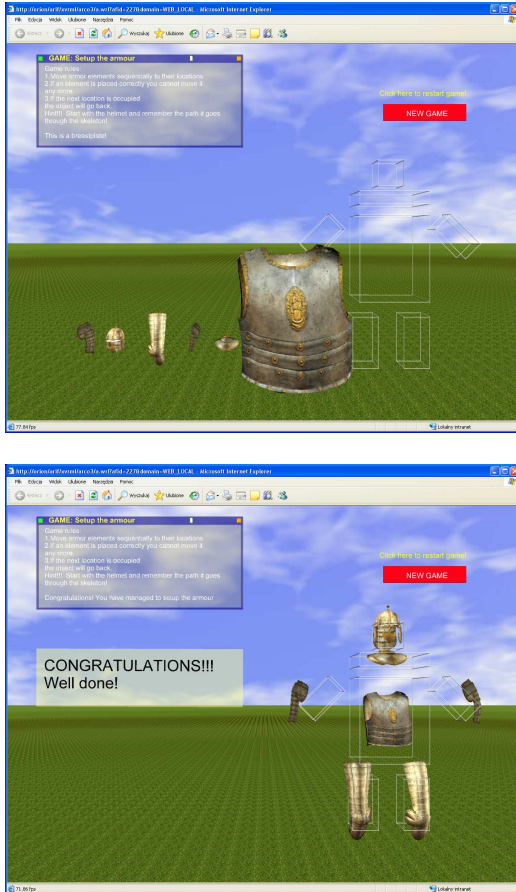


Figure 7: Armor Puzzle Game interface during the game (top) and completed (bottom).

6. Conclusions

In this paper we have presented a method of dynamic modelling of 3D contents behaviour, called Beh-VR, along with its application for building virtual museum exhibitions. The presented approach enables dividing the content creation chain into several stages, such as preparation of object models, design of presentation templates, programming of object behaviour and designing actual exhibitions. These tasks can be performed by different users with different sets of skills.

The Beh-VR approach enables dynamic composition of virtual scenes with complex interactivity and behaviour, enabling domain experts to create advanced 3D presentations without programming. Moreover, the dynamic modelling approach enables end-users (e.g., museum visitors) to select the contents or the presentation method.

The method has been successfully used for building interactive 3D interfaces for several museums. The achieved results demonstrate high potential of the presented method for building advanced virtual museum exhibitions.

Future works include extending the VR-BML language with more advanced commands, implementation of user-friendly tools for designing virtual exhibitions, and testing the approach on mobile platforms.

Acknowledgements

3D models and video sequences courtesy of the Polish History Museum [Pol07] and TVP S.A.

References

- [3ds07] Autodesk 3ds max. <http://www.autodesk.com/3dsmax>, 2007.
- [ARC07] Arco virtual museums system. <http://www.arco-museums.eu/>, 2007.
- [Bri07] The british museum. <http://www.thebritishmuseum.ac.uk/>, 2007.
- [Gug07] Guggenheim museum. <http://www.guggenheim.org/>, 2007.
- [may07] Autodesk maya. <http://www.autodesk.com/maya>, 2007.
- [Mus07] Virtual museum of warsaw uprising. <http://wirtualnemuzeum.1944.pl/>, 2007.
- [Pol07] Polish history museum. <http://www.muzhp.pl/>, 2007.
- [Tys03] TYSON N.: The conveyor project - multimedia authoring made easy. In *Proc. of ICHIM 2003, Ecole du Louvre, Paris* (2003).
- [Vir07] Virtools. <http://www.virttools.com/>, 2007.
- [Wal06] WALCZAK K.: Beh-VR: Modeling behavior of dynamic virtual reality contents. In *The 12th International Conference on Virtual Systems and Multimedia VSMM 2006*, in: H. Zha et al. (Eds.): *Interactive Technologies and Sociotechnical Systems, Lecture Notes in Computer Sciences 4270, Springer Verlag Heidelberg* (2006), pp. 40–51.
- [WC03] WALCZAK K., CELLARY W.: X-vrml for advanced virtual reality applications. *Computer* 36, 3 (2003), 89–92.
- [Wis02] WISE S.: The why, what, and how of a custom, authoring and publishing system: The creation of pachyderm. *SPECTRA* 29, 1 (2002), 30–33.
- [WV05] WISE S., VYAS S.: Exart (exploring with artists' real tools): A table-based interface for families. In *From Content to Play: Family-Oriented Interactive Spaces in Art and History Museums: J. Paul Getty Museum Symposium* (2005).
- [WW05] WALCZAK K., WOJCIECHOWSKI R.: Dynamic creation of interactive mixed reality presentations. In *VRST '05: Proceedings of the ACM symposium on Virtual reality software and technology* (New York, NY, USA, 2005), ACM Press, pp. 167–176.