

# GRAPPLE

D3.4c Version: 1.0

## Implementation of an authoring platform for adaptive VR learning material

<b>Document Type</b>	Deliverable
<b>Editor(s):</b>	Frederic Kleinermann (VUB) and Olga De Troyer (VUB)
<b>Author(s):</b>	Frederic Kleinermann (VUB) and Olga De Troyer (VUB)
<b>Reviewer(s):</b>	Eelco Herder (LUH) and Jan Hidders (TUD)
<b>Work Package:</b>	WP3
<b>Due Date:</b>	31/08/2010
<b>Version:</b>	1.0
<b>Version Date:</b>	02/11/2010
<b>Total number of pages:</b>	73

**Abstract:** This document is the third deliverable of Work Package 3, task T3.4. It describes the final implementation of the authoring tool for Virtual Reality learning material. While deliverable 3.4a provides related work and gives a general overview of the design for the VR Authoring tool and deliverable 3.4b gives a description of an initial implementation, this deliverable provides a more in depth view on how the VR authoring tool has been implemented. It also provides a revised version of deliverable 3.4a, 3.4b and a short user guide.

**Keyword list:** Virtual Reality, authoring tool, adaptation, domain model, CAM model

**Disclaimer:**

All information included in this document is subject to change without notice.

The Members of the GRAPPLE Consortium make no warranty of any kind with regard to this document. Including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. The Members of the GRAPPLE Consortium shall not be held liable for errors contained herein or direct, indirect, special, incidental or consequential damages in connection with the furnishing, performance, or use of this material.

## Summary



## D3.4c: Implementation of an authoring platform for adaptive VR learning material

WP3 focuses on the design and development of a GRAPPLE authoring tool to support authors in writing adaptive courses. Deliverable 3.4 describes the development of an authoring tool that allows authors to specify how Virtual Reality (VR) material should be adapted during a course taken by a learner. The authoring and delivery of adaptive VR Learning material are the focus of Deliverables D3.4a, D3.4b, D3.4c, D4.2a, D4.2b and D4.2c.

This deliverable will describe the implementation of the VR authoring tool which can be considered as a part of the general Grapple Authoring Tool (GAT). It follows up on deliverable D3.4a and D3.4b and focuses on the implementation of the authoring platform for adaptive VR learning material.

The purpose of this deliverable is:

- To update and revise the material of the previous deliverable (D3.4a) and (D3.4b);
- To describe the implementation of the VR authoring tool component;
- To explain how it is integrated with the rest of the Grapple Authoring Tool (GAT);
- To explain the improvement made since D3.4b in terms of the VRCRTs and the VRCAM;
- To provide a short user-guide.

When familiar with D3.4a and D3.4b, please skip chapters 1 to 3. Chapter 4 and onwards are new to this deliverable.

## Authors

Person	Email	Partner code
Olga De Troyer	<a href="mailto:Olga.DeTroyer@vub.ac.be">Olga.DeTroyer@vub.ac.be</a>	VUB
Frederic Kleinermann	<a href="mailto:Frederic.kleinermann@vub.ac.be">Frederic.kleinermann@vub.ac.be</a>	VUB

## Table of Contents

<b>SUMMARY</b> .....	<b>1</b>
<b>AUTHORS</b> .....	<b>2</b>
<b>TABLE OF CONTENTS</b> .....	<b>2</b>
<b>TABLES AND FIGURES</b> .....	<b>4</b>
<b>LIST OF ACRONYMS, ABBREVIATIONS AND SYNONYMS</b> .....	<b>5</b>
<b>1 INTRODUCTION</b> .....	<b>7</b>
1.1 VIRTUAL ENVIRONMENTS .....	7
1.2 WHY ADAPTATION IN VIRTUAL REALITY? .....	8
1.3 VIRTUAL REALITY AND SIMULATIONS.....	8
1.4 CHALLENGES RELATED TO ADAPTATION FOR VES .....	8
<b>2 RELATED WORK</b> .....	<b>8</b>
2.1 E-LEARNING IN 3D ENVIRONMENTS.....	9
2.2 ADAPTATION AND 3D ENVIRONMENTS .....	12
2.3 DISCUSSION .....	17

<b><u>3</u></b>	<b><u>ADAPTATION FOR VR IN THE CONTEXT OF GRAPPLE</u></b>	<b><u>17</u></b>
3.1	ADAPTATION TYPES FOR VR	17
3.2	ADAPTATION STRATEGIES FOR VR	18
<b><u>4</u></b>	<b><u>REVISIONS</u></b>	<b><u>22</u></b>
4.1	REVISION BETWEEN D3.4A AND D3.4B	22
4.2	REVISION BETWEEN D3.4B AND D3.4C	24
<b><u>5</u></b>	<b><u>DESCRIPTION OF THE INTEGRATION OF THE VR AUTHORING TOOL WITH GAT</u></b>	<b><u>24</u></b>
5.1	INTEGRATION OF THE VR AUTHORING TOOL WITH THE OTHER COMPONENTS OF THE GAT TOOL	24
5.2	INTEGRATION OF DATA FROM THE VR AUTHORING TOOL INTO CAM	26
5.3	VRCAM	27
5.4	REPOSITORY FOR VR	27
5.5	SPECIFICATION OF THE VR DATA MODELS	27
<b><u>6</u></b>	<b><u>DESCRIPTION OF THE VR AUTHORING TOOL</u></b>	<b><u>28</u></b>
6.1	PREVIEWING VR RESOURCES	30
6.2	IMPORTING EXISTING VR RESOURCES	31
6.3	PREPARING VR RESOURCES	33
6.4	VIRTUAL REALITY ADAPTATION STATES	34
6.5	PREPARING THE VR LEARNING OBJECTS	36
6.6	VIRTUAL REALITY CONCEPT RELATIONSHIP TYPE (VRCRT)	37
6.7	VIRTUAL REALITY CONCEPTUAL ADAPTIVE MATERIAL (VRCAM)	40
6.8	IMPORTING VR RESOURCES INTO A CAM	44
<b><u>7</u></b>	<b><u>USER GUIDE</u></b>	<b><u>46</u></b>
7.1	STEP 1: IMPORT AND PREPARE THE VR RESOURCES NEEDED	47
7.2	STEP 2: ADD VRASs TO VR OBJECTS	48
7.3	STEP3: DEFINE VRCRTs (OPTIONAL)	49
7.4	STEP 4: CREATE A VRCAM TO SPECIFY THE ADAPTATION INSIDE THE VR RESOURCE	51
7.5	STEP 5: ADDING THE VRCAM TO THE CAM	56
7.6	STEP 6: ADDING A VR RESOURCE WITHOUT ADAPTATION TO THE CAM	58
<b><u>8</u></b>	<b><u>GENERAL DISCUSSION</u></b>	<b><u>59</u></b>
<b><u>9</u></b>	<b><u>CONCLUSION</u></b>	<b><u>60</u></b>
	<b><u>REFERENCES</u></b>	<b><u>60</u></b>
<b><u>10</u></b>	<b><u>APPENDIX</u></b>	<b><u>62</u></b>
10.1	XML SCHEMA FOR VIRTUAL REALITY CONCEPTUAL ADAPTATION MODEL (VRCAM)	62
10.2	XML SCHEMA FOR VIRTUAL REALITY CONCEPT RELATION TYPE (VRCRT)	65
10.3	XML SCHEMA FOR VIRTUAL REALITY ADAPTATIONSTATE (VRAS)	67
10.4	XML SCHEMA FOR VIRTUAL REALITY RAW MODEL (VRRAW MODEL)	69
10.5	XML SCHEMA FOR VIRTUAL REALITY LEARNING MATERIAL (VRLM)	70
10.6	XML SCHEMA FOR VIRTUAL REALITY LEARNING OBJECT (VRLO)	71
10.7	XML SCHEMA FOR VRRESOURCE FOR GALE AND THE VR PLUG-IN	72
	<b><u>DOCUMENT CONTROL</u></b>	<b><u>ERROR! BOOKMARK NOT DEFINED.</u></b>

## Tables and Figures

### List of Figures

Figure 1. Virtual Environment.....	8
Figure 2. Sloodle.....	9
Figure 3. Gestures of Avatars in Sloodle.....	10
Figure 4. <e-adventure> used for medical simulation.....	10
Figure 5. AliveX3D.....	11
Figure 6. Virtual U.....	12
Figure 7. Adaptive navigation technique.....	13
Figure 8. Santos and Osorio Software architecture.....	15
Figure 9. Virtual Conference.....	14
Figure 10. <e-adventure> Authoring tool.....	15
Figure 11. Humanoid character.....	16
Figure 12. E-learning adaptive environment.....	16
Figure 13. Overview of Adaptation types and strategies.....	22
Figure 14. Relation between GAT and VR Authoring tool.....	25
Figure 15. Architecture of VR Authoring Tool with the different data models.....	25
Figure 16. VR Authoring Tool invoked from GAT.....	26
Figure 17. VR Toolbox manager for the VR Authoring tool.....	29
Figure 18. New File.....	29
Figure 19. Open File.....	30
Figure 20. Previewing a VR resource.....	31
Figure 21. "New file" for importing VR material. It needs to be of type VRRAW.....	31
Figure 22. Copy and Paste X3D Code.....	32
Figure 23. Loader.....	33
Figure 24. Identifying VR objects.....	34
Figure 25. Adaptation State "Marking" with the VR Code and the instruction to turn it on/off.....	35
Figure 26. Adaptation State "Marking" with parameters.....	36
Figure 27. Customisation of Adaptation State and Preparation of Learning Object.....	37
Figure 28. Overview of the steps to follow to prepare a VR resource for its use in a VRCAM.....	37
Figure 29. Main interface for creating a VRCRT.....	38
Figure 30. Editor for creating a rule for a VRCRT.....	38
Figure 31. User Variable tab.....	39
Figure 32. Console for VRCRT.....	40
Figure 33: Main interface VRCAM.....	41
Figure 34. Course Entities.....	41
Figure 35. VRCRTs.....	42

Figure 36. Mapping VR objects to domain concepts ..... 42

Figure 37. Example VRCAM ..... 43

Figure 38. Previewing VRAS used in a VRCAM..... 43

Figure 39. Inspecting a VRCT in a VRCAM..... 44

Figure 40. Inserting a VRCAM into a CAM ..... 45

Figure 41. Previewer of VR Resources ..... 46

Figure 42. The loader tool ..... 47

Figure 43. Identifying VR objects ..... 48

Figure 44. Associate the VR Adaptation State *VRAS\_MarkPedagogy* to Earth..... 49

Figure 45. Creating a VRCRT: General Tab window ..... 50

Figure 46. VRCRTActivateBehaviourAfterMarking: GAL code ..... 50

Figure 47. VRCRTActivateBehaviourAfterMarking: User Variable tab..... 51

Figure 48. Mapping VR objects to Domain Concepts ..... 52

Figure 49. A start entity and an Adaptive Block have been dragged on the canvas..... 53

Figure 50. Adding VR object Author\_Earth with a VRAS for marking it ..... 53

Figure 51. List of available VRCRTs ..... 54

Figure 52. *markNow* VRCRT has been selected..... 54

Figure 53. DoubleClicking on a VRCRT shows the rule and allows adapting it ..... 54

Figure 54. Adding the VRCRT *annotateNow*..... 55

Figure 55. VRCRT for activating a behaviour ..... 55

Figure 56. VRCAM for step 3 ..... 55

Figure 57. Step 4 realised ..... 56

Figure 58. Step 5: Ending the course ..... 56

Figure 59. Overall view of the CAM in the GAT tool ..... 57

Figure 60. CAM in more details..... 57

Figure 61. VRCAM being inserted into a socket of the CAM related to the Solar system ..... 58

Figure 62. Inspecting a VR Learning Material using the previewer ..... 59

## List of Acronyms, Abbreviations and Synonyms

2D	<i>Two Dimensional</i>
3D	<i>Three Dimensional</i>
AdapTIVE	<i>Adaptive Three Dimensional Intelligent and Virtual Environment</i>
AS	<i>Adaptation Strategy</i> , also called: Conceptual Adaptation Model; Adaptation Strategy
AT	<i>Adaptation Types</i>
CAM	<i>Conceptual Adaptation Model</i> , also called: Adaptive Story Line; Adaptation Strategy
CRT	<i>Concept Relationship Type</i> ; also called: (Pedagogical) Relationship Type
DOM	<i>Document Object Model</i>
DM	<i>Domain Model</i> ; Domain Map also called: Subject Matter Graph



D3.4c: Implementation of an authoring platform for adaptive VR learning material

DTD	<i>Document Type Definition</i>
GAL	<i>GRAPPLE Adaptation Language</i>
GALE	<i>GRAPPLE Adaptive Learning Environment</i>
GRAPPLE	<i>Generic Responsive Adaptive Personalised Learning Environment</i>
GAT	<i>GRAPPLE Authoring Tools</i>
GUMF	<i>GRAPPLE User Model Framework</i>
IMS	<i>Innovation Adoption Learning</i>
IVA	<i>Interaction and Virtual Agent</i>
LMS	<i>Learning Management System</i>
SIM	<i>Simulation</i>
UI	<i>User Interface</i>
UM	<i>User Model</i>
VE	<i>Virtual Environment</i> also called Virtual World
VR	<i>Virtual Reality</i>
VRML	<i>Virtual Reality Modelling Language</i>
VRLM	<i>Virtual Reality Learning Material</i>
VRLO	<i>Virtual Reality Learning Object</i>
VRAS	<i>Virtual Reality Adaptation State</i>
VRCAM	<i>Virtual Reality Conceptual Adaptation Model</i>
VRCRT	<i>Virtual Reality Concept Relationships Types</i>

# 1 Introduction

This document gives an overview of the Virtual Reality (VR) Authoring tool for GRAPPLE. This report has the following structure:

- Chapter 1 general introduction.
- Chapter 2 related work.
- Chapter 3 general overview of adaptation for VR.
- Chapter 4 previous update of the design given in D3.4a and D3.4b.
- Chapter 5 description of the implementation of the VR Authoring tool and its integration with the other Grapple Authoring Tools (GAT).
- Chapter 6 overview of the tool.
- Chapter 7 the User Guide.
- Chapter 8 general discussion.
- Chapter 9 conclusion.

This chapter starts with a brief reminder of what VR is, followed by an explanation of adaptation for VR and what the difference is between VR and Simulation.

## 1.1 Virtual Environments

There are many definitions of Virtual Environments (VE) or Virtual Reality (VR) (Gutierrez et al., 2008). Usually, the restrictive approaches define VR as three-dimensional (3D), multisensory, immersive, real time and interactive simulations of a space that can be experienced by users via three-dimensional input and output devices. For the context of this research, VR is defined as a three-dimensional computer representation of a space in which users can move their viewpoints freely in real time.

A VE is made of different components and can be summarised as:

### (1) The scene and the objects

The scene corresponds to the world in which the objects are located. It contains lights, viewpoints and cameras. It has also some properties that apply to all the objects located within the virtual world. For instance, gravity can be a property of the world that applies to all its objects. The objects have a visual representation with colour and material properties. They have a size, position and an orientation.

### (2) Behaviours

The objects may have behaviours. For instance, they can move, rotate, change size and so on.

### (3) Interaction

The user must be able to interact with the virtual world and its objects. For instance, a user can pick up some objects or can drag an object. This may be achieved by means of a regular mouse and keyboard or through special hardware such as a 3D mouse or data gloves (Gutierrez et al., 2008).

### (4) Communication

Nowadays more and more VR applications are also collaborative environments in which remote users can interact with each other. To achieve this, network communications are important.

### (5) Sound

VR applications also involve sound. Some research has been done over the last ten years in order to simulate sound in VR application.



Figure 1. Virtual Environment

Figure 1 shows an example of a VE. In this VE, the user is represented by an avatar (Figure 1 uses a warrior) which is an object with behaviour. The platforms on which the avatar can stand, are objects without behaviour. The user interacts with the VE either through the keyboard or the mouse.

## 1.2 Why Adaptation in Virtual Reality?

Virtual reality brings 3D content that can help the user to learn or acquire a better understanding of a topic or a context. In the context of E-Learning, VR can help the learner to better understand the different concepts of the course by being immersed in a 3D virtual environment. The adaptation in VR can help the learner to focus by e.g. only showing him the VR objects that are relevant to him and hide other objects in the VE. As a VE can be rich and explored in different ways, the user can easily get lost. This should be avoided, especially in the context of E-learning, as this may cause the learner to give up. Therefore, it is important to have a VE that can adapt to the user.

To achieve this, it is important to first define what can be adapted within a VE, how it can be adapted and what the process of a VE is. Changes will be made in real-time according to the learner's background and knowledge.

## 1.3 Virtual Reality and Simulations

It can be said that VR and simulations are very closely related concepts. A VE can be seen as a space in which the user is immersed and in which he can see a simulation. However in the context of VR for E-learning, the user is not learning about how to make a VE, he is using the VE as a place where the learning takes place. While in the context of Simulations for E-learning, the learner is learning either about or how to control and use the model being simulated.

## 1.4 Challenges related to adaptation for VEs

Within a VE, content is composed of 3D objects placed in 3D environment. In VEs, users move seamlessly and this is different from following links like on the Web by clicking on them. As a consequence, it is difficult to apply well-known techniques used for adaptation of hypertext. For instance, if a 3D object is inserted (in the same way as paragraphs can be inserted in a page), there will be the risk of having a cluttered, incomprehensible and not navigable VE! Therefore, adaptation of 3D content requires special care to preserve a meaningful and usable 3D environment. Objects should normally not intersect each other and must be visible from various user positions, while users must have enough free space to move around in. In addition, the size of the different objects should be in harmony and behaviours and interactions should be suitable for the objects to which they relate. Therefore, providing an adaptation authoring tool for VR content is not straightforward.

## 2 Related work

This chapter will start by reviewing some work using 3D or VR for E-learning. This section is provided in order to illustrate that VR and 3D are indeed feasible in the context of E-learning. The second part of the chapter will describe related work that is also dealing with adaptation in a 3D learning environment.

## 2.1 E-learning in 3D environments

In this subsection, a number of 3D environments developed for E-learning will be reviewed. The purpose of this section is to show the possibilities of 3D environments in the context of E-learning and to illustrate that in the future more and more 3D learning material may become available. Some work is based on existing Virtual Reality (VR) platforms (more specifically VR for the web) and others on Game platforms. More and more, game platforms are used for purposes other than purely entertainment (see e.g., the Serious Games initiative<sup>1</sup>).

Barab et al. (2005) have used a commercial gaming environment to develop a multiuser, virtual learning environment to immerse children, aged 9-12, in educational tasks. It allows users at the participating elementary schools and after-school centres to travel through virtual spaces to perform educational activities, talk with other users and mentors and build virtual personae. 3D aspects are provided through the game platform and customisation is provided, but there is no real adaptation in the sense of adapting the objects populating the virtual world at real-time according to the student's background and knowledge.

Livingstone and Kemp (2008) introduce a learning environment approach integrated with SecondLife<sup>2</sup> (an online 3D virtual world) and Moodle<sup>3</sup> (a course management system). Their approach is called Sloodle. It mimics the structure of a Moodle course homepage with 3D objects (see figure 2). When the course designer repositions web content blocks, the corresponding objects are automatically repositioned. Sloodle does not provide any adaptation to the user's profile. But it is interesting to see the use of SecondLife as a learning environment and the use of virtual objects specific for E-learning. They have also developed gestures of avatars (see figure 3) specific for E-learning.

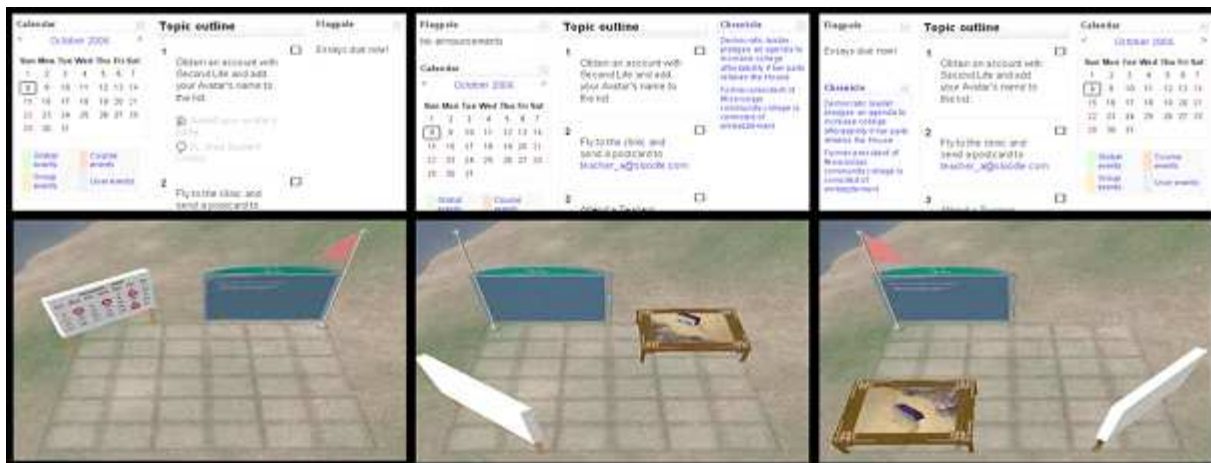


Figure 2. Sloodle

<sup>1</sup> <http://www.seriousgames.org>

<sup>2</sup> <http://secondlife.com/>

<sup>3</sup> <http://moodle.org/>



Figure 3. Gestures of Avatars in Sloodle

Alexiou et al. (2004) have presented a virtual laboratory, which is designed and implemented in the framework of the VirRAD European project. This laboratory represents a 3D simulation of a radio-pharmacy laboratory, where learners, represented by 3D avatars, can experiment on radio-pharmacy equipment by carrying out specific learning scenarios. This work also does not produce adaptations at real-time according to the student's background and knowledge.

Moreno-Ger et al. (2008a) describe a design and implementation methodology for medical simulations that avoids the high development costs of VR-like simulations (see figure 4). Their methodology takes into account that educational games and game-like simulations cannot be independent artefacts and therefore must be integrated with co-existing courses and training modules. For their work, they use <e-Adventure> (see 2.2) which is an educational game engine and which also has different plug-ins with the objective of integrating the educational games with online learning environments compliant to IMS Learning Design specification.

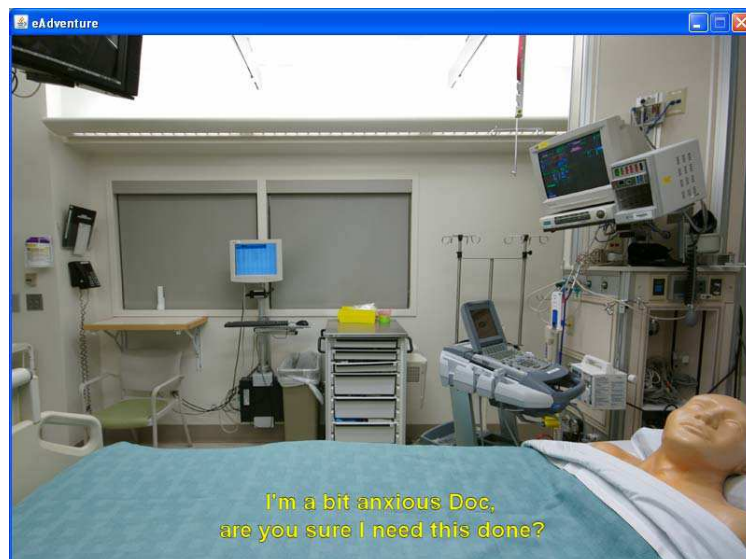


Figure 4. <e-adventure> used for medical simulation

Recently, Torrente et al. (2009) discuss how games can be mixed with E-learning. They explain the challenges associated with that and how to address those with the e-adventure platform. They describe their authoring tool and how they try to make life easier for the author. The work reported in this paper is different from the one reported in this deliverable as they do not take into account how to allow an author to define

adaptation inside virtual environments. Nevertheless, this paper is interesting as they try to make life easier for the author.

De Byl (2009) presents an Embedded Authentic Serious game-based Learning Experiences (EASLE) architecture which has been developed to assist in the definition of game based applications. De Byl makes the remark that contemporary game design architectures place great emphasis on storylines and character development in a top-down approach. This was not considered adequate and for this reason EASLE allows designer to specify game aspects by building up foundation layers where each successive layer refines the previous one and makes Gameplay more complete. The layers are Story, Scenes, Icons, Theme, Game rules, Gameplay mechanics and Genre. This work does not allow adaptations based on the user's background and knowledge but it does provide a methodology to develop game-based learning environments.

The Alive X3D project<sup>4</sup> is to bring the power of 3D gaming and simulation technologies to E-learning platforms and to educational experiences. It is based on the observation that most of the 3D environments for teaching mainly focus on the distribution of existing learning materials coupled with tools for communication. The Alive X3D project adopts Web 2.0 moving towards interoperable and reusable 3D learning objects, forming a Web2.0 – Web3D hybrid approach (see figure 5). Their work does not take into account any adaptation, but it shows how virtual environments can be used in the context of E-learning on the Web.



Figure 5. AliveX3D

KM Quest<sup>5</sup> is a Web-based simulation game aiming at developing core knowledge management skills with situated problems using collaborative learning. It is a team-based game in which 3 players are responsible for a fictitious company called Coltec. In that game, players are given an introduction to the basic concepts of knowledge management and training. The training consists of a limited form of the game with restricted choices and fixed events during the guidance and coaching and immediate feedback is given. 3D aspects are provided through the game platform. KM Quest also has no real adaptation in the sense of adapting the objects populating the virtual world at real-time according to the student's background and knowledge.

Virtual U<sup>6</sup> is a project started in 1997 aiming at creating a computer simulation of university management in game form. It is based on the idea of creating a "SimCity" for universities. It is now used across universities in America and demonstrates the use of games for E-learning (see figure 6).

<sup>4</sup> <http://www.alivex3d.org>

<sup>5</sup> <http://www.kmquest.net>

<sup>6</sup> <http://www.virtual-u.org/>



Figure 6. Virtual U

Like KM Quest, 3D is provided through the game platform, but there is no true adaptation.

## 2.2 Adaptation and 3D Environments

The previous section showed that different efforts exist in merging E-learning with 3D Virtual Environments. This section reviews related work in the context of adaptation and 3D virtual environments. Recall that the purpose of adaptation in E-learning is to personalise the course material to the profile of the learner. This section has a large overlap with the section “State of the Art and Related Work” in the deliverable D4.2b, which deals with the adaptive delivery of VR. This is because in most work on adaptation for 3D environments, there isn’t a strict distinction between the authoring and delivery. That kind of work is therefore relevant to both deliverables. To indicate the overlap between the two deliverables, the sections that are exactly the same are indicated by means of a vertical line in the left margin in both documents.

Brusilovsky et al. (2002) have developed an approach that supports different navigation techniques in the context of 3D E-Commerce. Their approach is based on Adaptive Hypermedia methods, which has been extended to Virtual Environments (here a 3D shop). It introduces:

*Direct Guidance.* This technique is based on the principle used in adaptive hypermedia where the user receives the best ‘next’ link according to his background. This principle has been extended to VR where the link is replaced by a viewpoint in the virtual world and where the best viewpoint, according to his background, is given to the user.

*Hiding.* This technique is based on the principle used in adaptive hypermedia where the navigation is restricted to a set of links. To realise this, different viewpoints for a specific object have been created. These viewpoints are within a scene sphere and as a result, the user cannot view any object out of this sphere. A new technique called Gaze Fixation has been defined where users can walk past the object with a system-generated orientation. It allows users to still control their position but not their orientation.

*Sorting.* This technique is based on the principle used in adaptive hypermedia where the links are sorted according to the user’s goal. A standard adaptive hypermedia application creates links related to a specific topic and arranges them according to their relevance. This principle has been extended to 3D environments by introducing *Point of Interest* navigation that adjusts the viewer’s motion speed logarithmically in relation to the distance from an object of interest.

*Annotation.* In adaptive hypermedia, a link can be displayed differently according to the user’s interest. For instance, the colour of a link can be changed or its visualisation can be changed. This principle has been extended to 3D environments in two ways: either by drawing an arrow pointing to the possible interesting object (see left side of figure 7) or by using the flashlight technique that highlights the object of interest (see right side of figure 7).



Figure 7. Adaptive navigation technique

Santos and Osorio (2004) have introduced another approach for adaptation in Virtual Environments. Their approach is called AdapTIVE (Adaptive Three-dimensional Intelligent and Virtual Environment) and is based on agents, called Interactive and Virtual Agents (IVAs), that assist the users and help them to interact with the environment. These agents help the user to discover the environment by providing more information about interesting products and by guiding him to his preferred area or products. Figure 8 displays the architecture of the proposed approach. Their approach has been applied to E-commerce and Distance Learning systems.

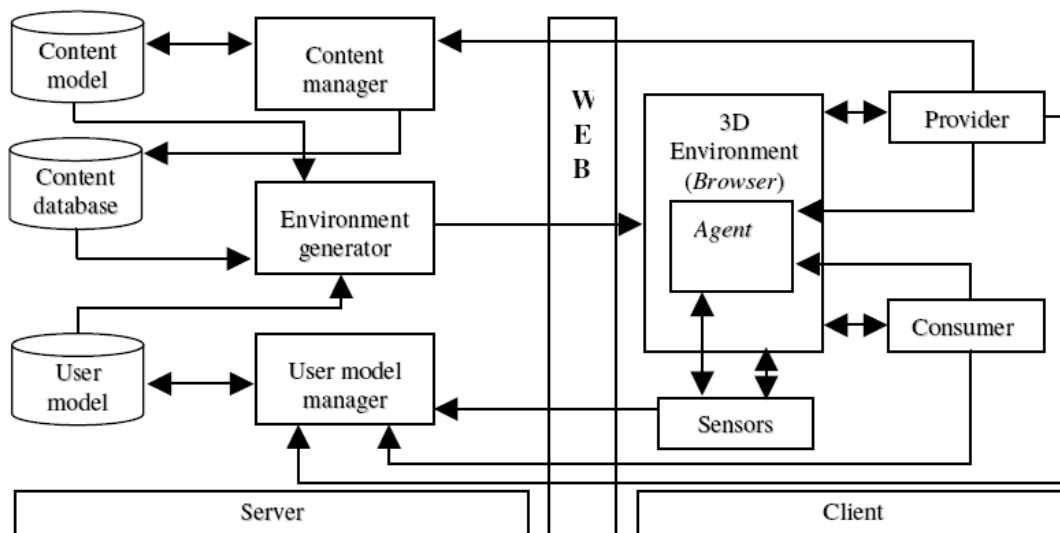


Figure 8. Santos and Osorio Software architecture

Daschelt et al. (2002) have developed an approach that provides adaptation to the user's device. Their approach suggests different alternatives with respect to the screen space usage for the same 3D interface element and information presented. 3D content is also considered in media adaptation. For instance, a

showcase where the seat capacity of a conference room can be adapted (see figure 9). The work presented in this paper is more about adapting the content to a large audience but not about personalisation.

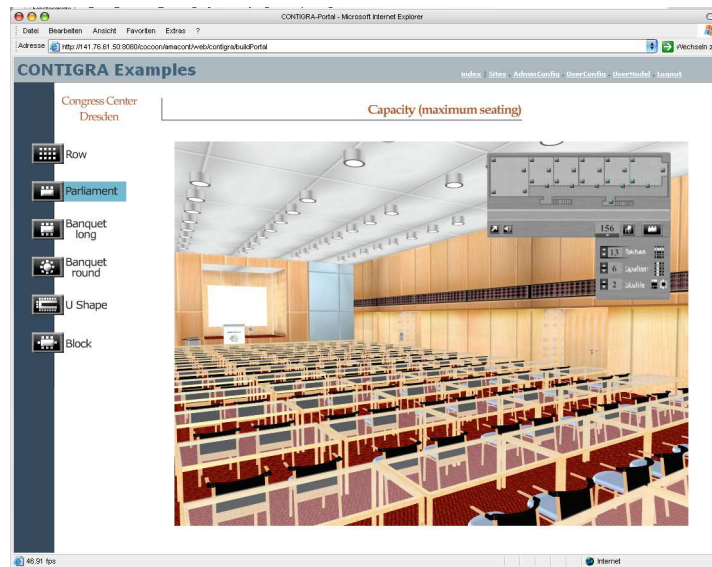


Figure 9. Virtual Conference

Moreno-Ger et al. (2008b) have developed an approach called <e-Adventure> which is an educational game engine designed to facilitate the creation of interactive educational content. It supports an adaptive model. The <e-Adventure> platform also includes a visual editor that facilitates the creation of the documents that describe the games (see figure 10). The platform can be integrated with a virtual Learning Environment (Moreno-Ger et al., 2008d). In their approach (Moreno-Ger et al., 2008c), the engine queries the Learning Management System for a set of properties (for instance the grade of a student) that are used to adapt the game. The games are defined so that different values of those properties will change the initial state of the gamer and as a consequence the game will be adapted. In this work, the content is adapted at the beginning of the game based on the student's profile. The presented work is very interesting in the context of GRAPPLE as it does provide an authoring tool allowing authors to create courses and it does provide some adaptations. However, the adaptations are only done between sessions or during an evaluation. Additionally, the content is mostly 2D, no true 3D.

In Blanco al (2009), they discuss the potential of combining learning environments with games and the challenges related to that. They present a general architecture that aims to tackle these challenges by abstracting the communication between the game and the learning environment. To achieve this, a Game Adaptation Layer (GAL) has been developed that monitors the student-game interaction through an API that the game core must implement. The GAL uses the collected interaction data to maintain a record of the activity of the student. Their work is similar to our work as we also do have some kind of abstraction represented in our architecture by *Update Component* and the *Monitor Component*. The *Update Component* can communicate to different Web VR players that can be interacted through javascript. The *Monitor Component* is responsible to monitor the students in real-time and to report that to GALE. Based on this, GALE can then instruct how to adapt the virtual world. The difference with our work is that we have a VR authoring tool which allows an author to specify what should be adapted within the virtual world and how and when (see deliverable 3.4b (Kleinermaann and De Troyer, 2009) ). Furthermore, they do not make a direct link with an adaptation engine such as AHA! or GALE.

In Westra et al (2009), they have a model for game adaptation that is guided by three main concerns: the trainee, the game objectives and the agents. More specifically, their approach uses the concept of learning agents to introduce adaptations. In order to coordinate the adaption of the agents, an organisational framework is used that specifies the limitations of the adaptation in each context. The work is interesting in the sense that an approach is provided to make adaptations to serious and complex games. However, it does not really provide any kind of authoring tool that can allow the author to decide what should be adapted and when.

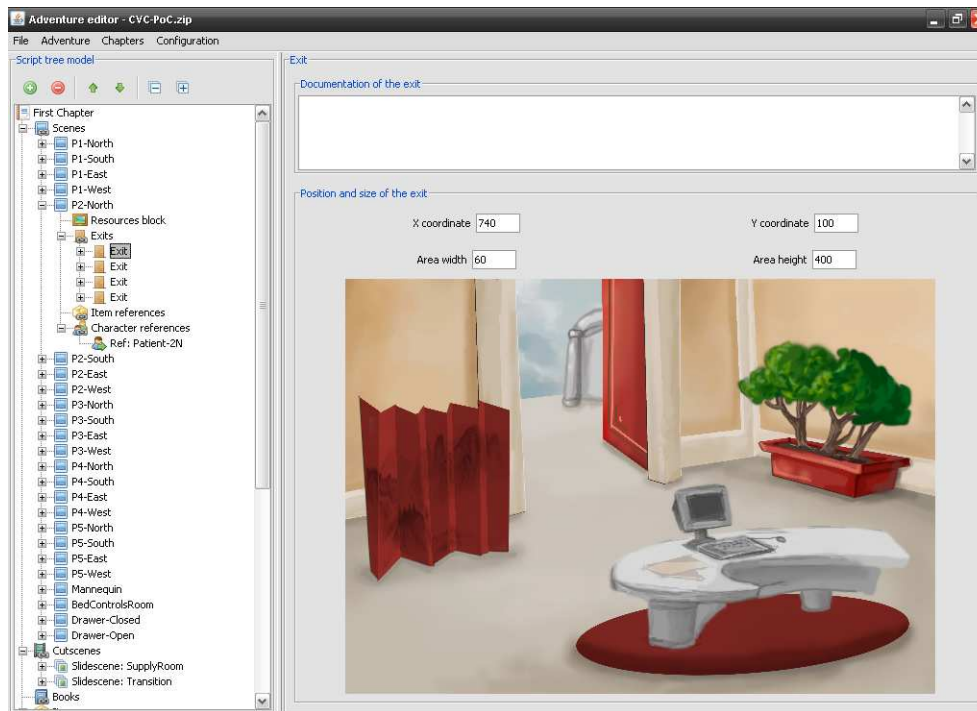


Figure 10. <e-adventure> Authoring tool

Celentano and Pittarello (2004) have developed another approach for adaptive navigation and interaction where a user's behaviour is monitored in order to exploit the acquired knowledge for anticipating the user's needs in future interactions. The approach uses sensors that tell when an object has been interacted with. These sensors collect usage data and compare them with previous patterns of interaction. These patterns represent sequences of activities that users perform in a specific situation during interactive execution of tasks and are encoded in Finite State Machines. Whenever the system detects that the user is entering a recurrent pattern of interaction, it may perform some activities of that pattern on behalf of the user. This approach focuses more on the navigation and interaction.

In 2000, Chittaro and Ranon (Chittaro and Ranon, 2000) has described how to introduce adaptation for e-commerce. This approach is called ADVIRT. A set of personalisation rules exploits a model of the customer to adapt features of the VR store, such as the display of products using the concept of shelves, displays and banners. The navigation and different layouts of the store have also been customised and personalised. The work reported in this paper focuses on E-commerce.

In 2002, Chittaro and Ranon (Chittaro and Ranon, 2002) have also introduced a software architecture for adaptive 3D web sites called Awe3D (Adaptive Web 3D) which can generate and deliver adaptive VRML. Awe3D has three modules; a Usage Data Sensing module, a usage Data Recorder module and an adaptive engine and 3D content creator. The Usage Data Sensing module is to monitor a user's interaction with the 3D Virtual Environment and sends the relevant events over the Internet. The Usage Data Recorder module is responsible for receiving the events sent by the Usage Data Sensing module and for recording them in the user model. The adaptive engine is responsible for performing the inferences needed to update the user model and given the current user model, to compute a set of adaptation choices for the requested adaptive content. The 3D content creator accepts the content request from the client.

In 2007, the same authors have explained in a paper entitled "Adaptive 3D Web site" (Chittaro and Ranon, 2007b) that adaptation can happen for navigation and interaction in order to help the users in finding and using information more efficiently. For navigation and interaction, direct guidance, hiding, sorting and annotation was proposed based on the work of Brusilovsky et al. (2002). Additionally an alternative approach for sorting and annotation was suggested, using virtual characters that act as navigation guides to show users the path to an object or to a list of objects of interest. It also uses annotations in order to provide additional information on navigation and interaction possibilities. An example of a humanoid character is shown in figure 11.

Based on their previous work (Chittaro and Ranon, 2004), Chittaro and Ranon have extended the E-learning platform EVE (Chittaro and Ranon, 2007a). Adaptive EVE was introduced that is tailored to the knowledge level of a student and his preferred style of learning. It also tries to re-create the feeling of real-life learning.

The student has to perform exercises and learn new materials adapted to the student's background. To achieve adaptivity in the context of EVE (Chittaro and Ranon, 2008), the AHA! engine is used which was originally developed for adaptive hypermedia applications. Figure 12 provides an overview of the architecture. Every concept in the domain model has been associated with related code in the Virtual Environment. When a student queries a specific concept by clicking on the object, the system will display the corresponding VE code depending on the current user's level of knowledge. This way the AHA! engine will use the user model and adaptation model to choose which 3D contents should be presented to the student. To demonstrate the work, a course on X3D has been developed, where the 3D content adapts to the student's background.



Figure 11. Humanoid character

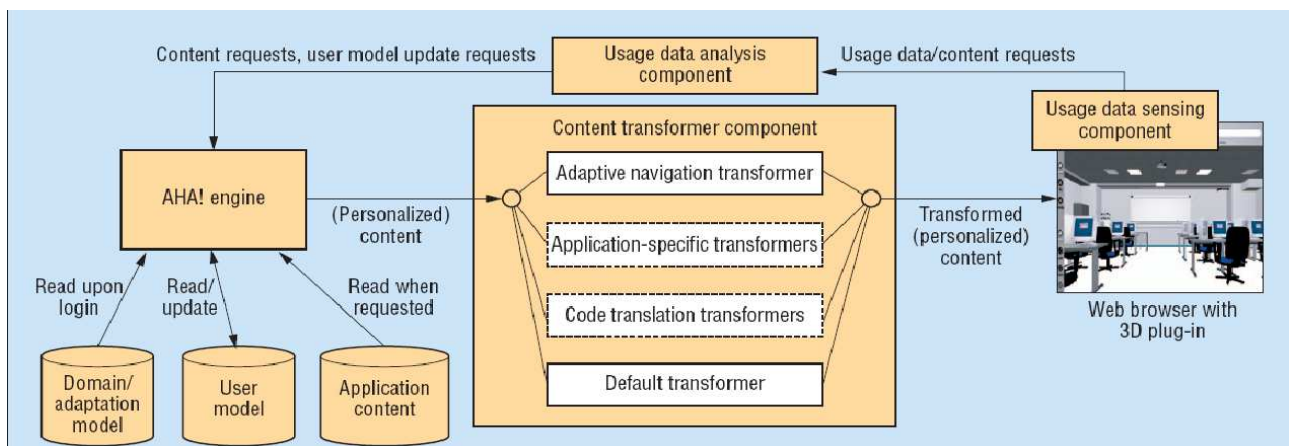


Figure 12.E-learning adaptive environment

This work is certainly close to the work to be developed for GRAPPLE. This work is not general enough so that any author can use an authoring tool for creating courses with a 3D adaptive virtual environment. It

certainly provides a lot of information on what to adapt and how to perform the adaptation. More information on X3D is given in Brutzman and Daly, 2008.

## 2.3 Discussion

This section has presented work related to the use of Virtual Reality and 3D material in the context of E-Learning. This section also presented some work done in the context of adaptation for Virtual Reality. Not all of the work on adaptation of Virtual Reality is oriented towards E-learning. Nevertheless, it provides a good basis for defining different types of adaptation that could be used in the context of Virtual Reality for E-learning. Most of the work on adaptation for Virtual Reality was done for one case (or one specific application) and is not general in the sense that it will easily be usable for other types of applications. As far as concerning authoring tools for VR adaptation, only one related work was found. The <e-adventure> platform includes a visual editor that allows authors to create a course and provides some means for adaptation. However, it does not provide ways to allow an author to define what can be adapted or how and when.

## 3 Adaptation for VR in the context of GRAPPLE

Based on the chapter about related work, this chapter will discuss possible adaptations for Virtual Reality (VR) in the context of GRAPPLE; to adapt VR learning material (3D objects as well as complete Virtual Environments) to the learner's profile. This chapter first discusses *Adaptation Types* for VR, which are a kind of elementary adaptations for a single object. Next it will discuss so-called *Adaptation Strategies* for VR that go beyond the adaptation of a single object.

### 3.1 Adaptation Types for VR

A VE can be adapted in many different ways. As described in the previous section, a VE contains a number of different components. In principle, adaptation can happen for each of these components. GRAPPLE will first focus on adaptation for the objects, the behaviours and the interaction. Next possible adaptations for the objects, behaviours and interaction will be described. These are called possible adaptations *Adaptation Types* (AT).

#### 3.1.1 Adaptation Types for Objects

Objects populating the VE have a visual appearance in terms of a geometry (shape) and material properties (colours and textures). To adapt a VE to the needs of the learner, it may be necessary to change the visual appearance of an object during the lifetime of the VE. This will be illustrated with some examples. To visually indicate that an object has not yet been studied it may be given a different colour or be smaller; when a learner is studying a complex object (like a planet), the visual appearance of the object could change according to the aspects being studied (size, temperature, geography, etc.) or become more detailed when more and more knowledge is acquired.

Therefore, the first category of adaptation types for objects concerns the adaptation of the visualisation of an object i.e. how to display it and how to hide it. To adapt the display, four adaptation types have been identified:

- *Semi-display*: this adaptation type is used to display the object in a semi-manner, e.g., by having a semi-transparent bounding box around it. In other words, semi-display consists of adding a box around the actual object and by making that box semi-transparent.
- *changeSize*: this adaptation type is used to change the visual appearance of an object by changing its size.
- *changeMaterialProperties*: this adaptation type is used to change the material properties (colour or texture) of an object.
- *changeVRRepresentation*: this adaptation type is used to change the visual representation of an object completely by replacing it with a different representation.

Two adaptation types concerned with hiding an object:

- *semi-hide*: this adaptation type allows the visual appearance of objects to be semi-hidden. This is similar to the semi-display but the purpose is to hide the object in a semi-manner.

- *hide*: this adaptation type allows to visually hide an object.

It is also necessary to be able to mark objects like in classic textual learning material. A reason for marking an object is for instance to draw the attention of the learner that the object has (or has not yet) been studied. In VR, marking an object can be done in different ways. Two different adaptation types have been distinguished for marking because they are essentially different in terms of VR:

- *spotlight*: by putting a spotlight on an object the object becomes more visible than other objects and can be considered as marked.
- *highlight*: marking is done by drawing a box around the object, only displaying the edges of the box.

Note that marking is considered different from changing the material properties (like the colour) of an object. The property of the object will not change by marking it.

In a VR learning environment, it may be useful to attach annotations to the VR objects. Annotations can be used to identify objects (e.g., display their name) or attach explanations to objects. It may be necessary to adapt the annotations to the profile of the learner. Therefore, the following adaptation types are identified:

- *displayAnnotation*: this adaptation type allows displaying an annotation associated with an object.
- *hideAnnotation*: this adaptation type allows hiding an annotation associated with an object.

### 3.1.2 Adaptation types for the scene

As explained in the previous section, the scene can have properties. There are physical properties, lighting, sound and the material properties of the scene itself. It may be useful to be able to adapt these properties as well. For example, lighting can be used to visualise that some parts of the VE have already been visited or are not yet accessible.

Due to lack of resources there will be no elaboration on possible adaptation types for the scene.

### 3.1.3 Adaptation of Behaviours

Behaviours are used to make the VE dynamic, i.e. to create environments where objects are active and show behaviour like planets that are rotating and comets that move through the universe. It may also be useful to consider adaptation of the behaviour, for instance by enabling or disabling behaviours in order to simplify or complicate the VE. Possible adaptation types for behaviours are:

- *enableBehaviour*: this adaptation type allows to enable a behaviour associated with an object.
- *disableBehaviour*: this adaptation type allows to disable a behaviour associated with an object.
- *changeBehaviour*: this adaptation type allows to change a behaviour by modifying the values of its parameters.

### 3.1.4 Adaptation of Interaction

By interaction we mean interaction between the end-user and an object. There are different ways to interact with an object, e.g., by clicking on an object, by touching an object or by closely passing an object. Interaction can also trigger some behaviour. For example, by interacting with an object the end-user can move it or start an animation. To fully adapt the VE to the needs of the learner, it may be necessary to adapt the interaction. For example, as long as the learner has not obtained a certain level of background knowledge he should not be able to manipulate a certain object. Adapting interaction comes down to *enabling* or *disabling* the possible types of interaction provided for an object. Possible interaction types considered for objects are: touching, clicking, and passing by. The adaptation types for interaction are:

- *enable*: this adaptation type allows to enable an interaction type for an object.
- *disable*: this adaptation type allows to disable an interaction type for an object.

## 3.2 Adaptation Strategies for VR

In this subsection, a number of VR *Adaptation Strategies* (AS) particularly useful in the context of E-learning will be discussed. An adaptation strategy goes beyond the adaptation of a single object. They can have an impact on several aspects of the VE or on a larger part of the VE. In principle, adaptation strategies can be defined by means of the adaptation types defined earlier. Adaptations strategies that are more like conditional adaptations are also included, for instance the 'displayAtMost' strategy will hide an object after it

has been displayed a given number of times. In general, it may be more convenient for a course author to use these VR adaptation strategies to define how to adapt the VE rather than the individual adaptation types.

A first group of strategies are those that will have an impact on how the learner can navigate through the VE. The following adaptation strategies for navigation have been distinguished:

- *restrictedNavigation*

This adaptation strategy allows an author to restrict the navigation in the VE by allowing it to happen only among a given list of objects. In other words, the learner will only be able to navigate from an object to another object from the list of objects. The navigation can also be restricted by either jumping or walking.

*Jumping* means going from one object to the next without viewing the part of the scene between the objects.

*Walking* means following a path in the scene from one object to the next.

The list of objects that can be visited can be specified in different ways, e.g., based on the user knowledge or based on the objects that are pre-requisite of each other.

*Possible Scenario:* This strategy can be used by the author of a course to force the learner to visit and explore only a pre-defined number of objects in a pre-defined way.

- *navigationWithRestrictedBehaviour*

This adaptation strategy allows the author of a course to restrict the possible behaviours of objects while the learner is navigating. The restricted behaviour can apply to all objects or to a specified list. The list of objects can also be specified in different ways, e.g., based on the user knowledge.

*Possible Scenario:* This strategy can be used to first allow the learner to explore the VE but without having the objects showing any behaviour; afterwards when he is familiar with the environment, behaviour can be enabled.

- *navigationWithRestrictedInteraction*

This adaptation strategy allows an author of a course to define that navigation through a list of objects is possible but without being able to fully interact with those objects. Again, the list of objects can be defined in different ways. It is also possible to have this restricted interaction for all objects in the VE.

*Possible Scenario:* This strategy can be used to first allow the learner to explore the VE but without allowing him to interact with the objects; afterwards when he is familiar with the environment, interaction can be enabled.

- *TourGuide*

The tour guide strategy can be used by the author to provide a tour guide to the learner to explain the context of the course. The tour guide will take the learner on a tour through the VE.

*Possible Scenario:* Like in real life, a tour guide can provide an easy and efficient way to learn about objects in a large and unknown VE.

The following adaptation strategies allow the author of a course to specify that the learner can navigate freely in the VE. Two types of free navigation strategies can be distinguished, namely *completelyFree* and *freeWithSuggestions*.

- *completelyFree*

This adaptation strategy will allow the learner to navigate freely in the VE.

*Possible Scenario:* This strategy can be used in an exploration-based learning style to let the learner explore the VE without any constraints in terms of navigation.

- *freeWithSuggestion*

This adaptation strategy will allow the learner to navigate freely in the VE but in addition, some objects will be "suggested". This suggesting is done by marking objects (using the adaptation type for marking an object, i.e. spotlight or highlight). The list of suggested objects can also be specified in different ways.

*Possible Scenario:* This strategy can be used to give the learner a lot of freedom but still provide some guidance.

A second group of adaptation strategies are those that allow adapting a group of objects:

- *filterObjects*  
This adaptation strategy allows the author of a course to filter the objects that should be available (visible) in the VE. The filter is possible by applying certain criteria on objects such as user-knowledge or user-background.  
*Possible Scenario:* The author can use this strategy to only show to the learner the objects that he is allowed to learn according to the pre-requisite rules.
- *markObjects*  
This adaptation strategy allows the author of a course to mark (e.g., highlight) a number of objects in the VE. The marking will be done for all objects satisfying certain criteria such as user-knowledge or user-background.  
*Possible Scenario:* This strategy can be used during the course to draw the attention of the learner to objects that still need to be visited.

A next group of adaptation strategies are strategies for displaying an object:

- *displayAtMost*  
This adaptation strategy allows an author to specify when an object should not be displayed anymore. This can be achieved using criteria like user-knowledge or by how many times it is allowed to display the object.  
*Possible Scenario:* The author can state that once the object has been displayed 10 times, it should not be displayed anymore. This will help learners to focus on the objects still to deal with.
- *displayAfter*  
This adaptation strategy allows an author to specify when an object should be displayed. This can be achieved using a criterion like user-knowledge or by means of a time criterion.  
*Possible Scenario:* The author can use this strategy for instance to keep the scene dynamic and interesting by displaying some objects only after a(n) (random) amount of time.

A next group of adaptation strategies are strategies for adapting the behaviour of an object.

- *behaviourAfter*  
This adaptation strategy allows an author to indicate when the behaviour of an object should be executed. This can be achieved by means of a criterion like user-knowledge or by means of a time criterion.  
*Possible Scenario:* The author can use this strategy to state that the behaviour of a specific object should only start when the knowledge of the learner is above a certain threshold.
- *behaviourAtMost*  
This adaptation strategy allows the author to specify when a behaviour should be disabled. This can be achieved by means of a criterion like user-knowledge or by how many times it is allowed to execute the behaviour.  
*Possible Scenario:* The author can use this strategy to specify that if a behaviour of a specific object has been executed 10 times, the learner should know it and it can be disabled to avoid that the learner will keep "playing" with it.
- *behaviourSpeed*  
This adaptation strategy allows the authors to specify the speed of a behaviour associated with an object.  
*Possible Scenario:* The author can use this strategy to make a behaviour (e.g., a simulation of a real-world behaviour) slower, depending on the learner's knowledge to allow him to observe the behaviour more carefully.

The last group of adaptation strategies contains strategies for the interaction with an object.

- *InteractionAfter*

This adaptation strategy allows the author to specify when the user can interact with the object. This can be achieved by means of a criterion like user-knowledge or by means of a time criterion.

*Possible Scenario:* The author can use this strategy to ensure that the end-user has enough knowledge about the object before he can interact with it.

- *InteractionAtMost*

This adaptation strategy allows the author to specify when the user cannot interact with the object anymore. This can be achieved by means of a criterion like user-knowledge or by how many times it is allowed to interact with the object.

*Possible Scenario:* The author can use this strategy to limit the interaction with an object. Again, it may be important in some situations to force the learner to explore new objects rather than sticking too long with objects the learner already knows well.

Figure 13 provides an overview of the adaptation types and adaptation strategies identified for VR.

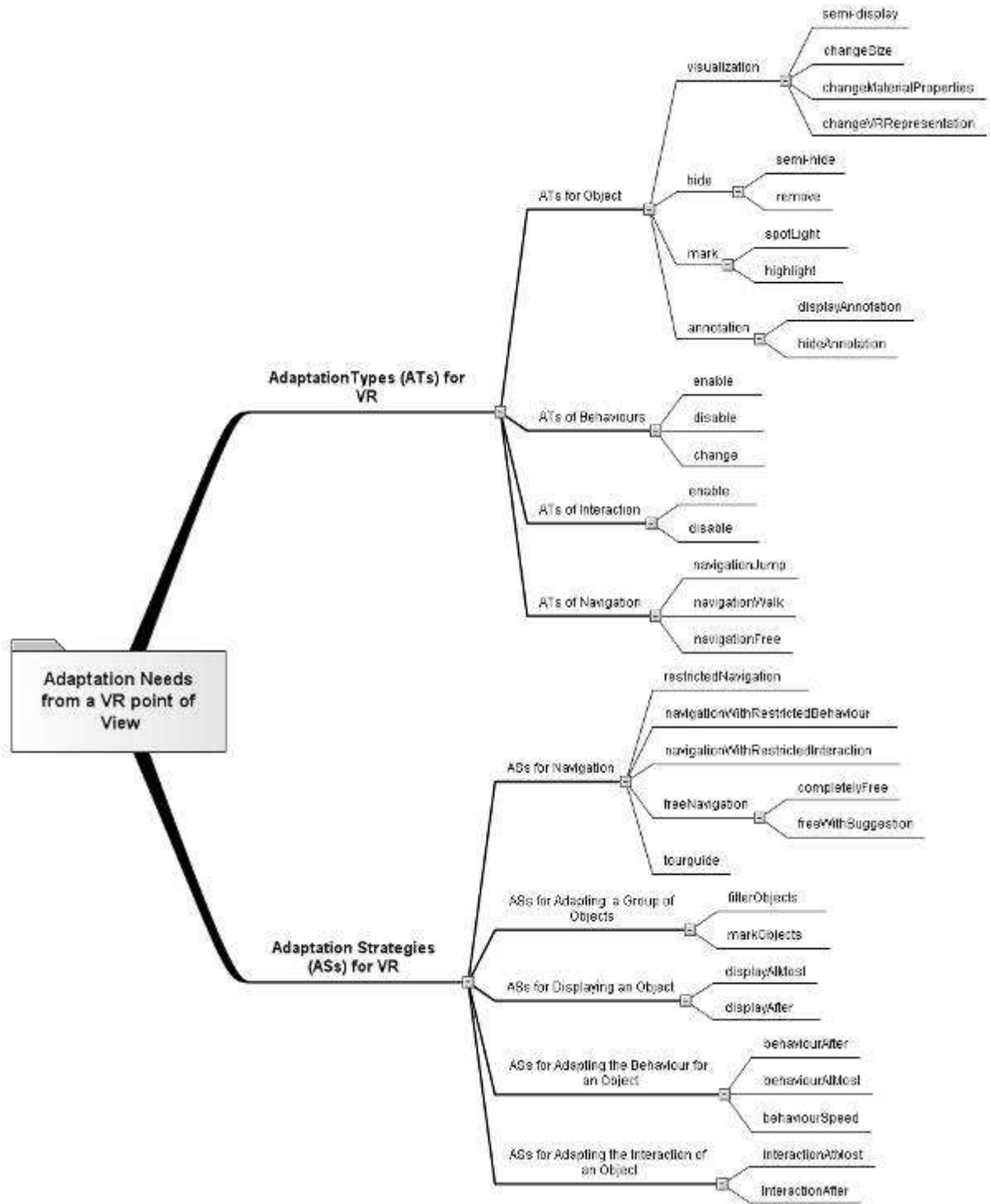


Figure 13. Overview of Adaptation types and strategies

## 4 Revisions

This chapter will introduce the revisions that have been made between deliverable 3.4a and deliverable 3.4b and between deliverable 3.4b and 3.4c.

### 4.1 Revision between D3.4a and D3.4b

There are three major changes between D3.4a and D3.4b. Firstly, the notion of *Virtual Reality Adaptation State* (VRAS) has been introduced in our design. This was needed to accommodate the limitations of the

current available VR players for the Web. Secondly, a visual way (graph) to specify the Virtual Reality Conceptual Adaptation Model (VRCAM) has been provided. This was done in order to be consistent with how the CAM was specified in the general authoring tool (GAT). Thirdly, the GAL code was directly introduced within the Virtual Reality Concept Relationships Types (VRCRT) when they were designed. This was also done for being consistent with the general GAT.

More details about these revisions are given in the following sections.

#### 4.1.1 Different Scenarios

In deliverable 3.4a (Kleineremann and De Troyer, 2008a), a number of scenarios on how VR resources can be included inside the CAM and having adaptation inside VR have been discussed.

The first scenario was to be able to include VR resources within a normal CAM. In that scenario, no adaptation happens on the VR resource. This scenario is still valid and the next chapters will show how that can be done.

The second scenario was to be able to specify adaptations that apply only to VR resources. This scenario is still valid and will be illustrated in the next chapters. To achieve this scenario the notion of VRCAM (which is similar to the notion of CAM but is dedicated to VR) has been introduced and the notion of VRCRT (which is also similar to the notion of CRT, but also dedicated to VR). Additionally the notion Virtual Reality Adaptation State (VRAS) has been introduced which will be explained in the next section and also in the next chapters.

The third scenario was a combination of the first and second scenario i.e. having adaptation within VR and being able to include these VR resources having adaptation within a normal CAM. This third scenario is still valid and the next chapters will show how that can be done.

#### 4.1.2 Virtual Reality Adaptation State (VRAS)

Deliverable 3.4a explained the first design of the VR Authoring tool. Since then, the concept of Virtual Reality Adaptation State (VRAS) has been introduced. The VRASs make it easier for an author to specify how an object needs to be adapted in a certain situation. A VRAS defines the effect of an adaptation for a VR object in a visual way. For example, suppose an author wants to light up a VR Object in a particular situation. To model this adaptation rule, the author (or a more experienced VR author) can first define an adaptation state (VRAS) where a spotlight is put on the VR object so that it lights up. By using Virtual Reality Concept Relationships Types (VRCRTs), the author can specify in the VRCAM when this VRAS should be used for the object.

Additionally, a VRAS actually follows the description of chapter 3, which describes what can be adapted within a VE made up of virtual objects with behaviours. For that reason different kinds of VRASs are supported such as spotlight, semi-hidden, marking, behaviour, and so on. The next chapters will describe how the VR authoring tool allows an experienced author to add a new VRAS.

From an implementation point of view, VRASs give the advantage of being able to treat VR objects as external resources similar to images (or videos) but with the difference that they are dedicated to VR only.

#### 4.1.3 Visual Graph for VRCAM

The goal was to have the VR Authoring tool integrated as much as possible with the other components of the Grapple Adaptive Tool (GAT) (Albert et al., 2009) (Hendrix et al., 2009) (Hendrix and Harrigan, 2009). Not only from the implementation point of view, but also from the way authors will use the tool. Since the design of a regular CAM was visual using a graph (see deliverable 3.3b (Hendrix and Harrigan, 2009), this design was also extended so that authors can create a CAM for a virtual world (VRCAM) in the same visual way as a regular CAM.

#### 4.1.4 VRCRT and GAL code

In Deliverable 3.4a (Kleineremann and De Troyer, 2009a), it was decided that the adaptation rule would be specified from a high-level point of view, i.e. away from GAL code and that the rule would be translated into GAL code later on by a tool. Due to the fact that the specification of a regular CRT in the version of GAT at the time of making D3.4b was done by having GAL code being explicitly entered, this design was adapted to allow entering GAL code explicitly in the VRCRTs. This was done for providing the author a consistent way of working. Although there is a prototype that helps the author to specify a VRCRT from a higher-level and away from GAL code, it was decided not to introduce it in this version of the VR authoring tool as GAL was still being refined at the time of writing the deliverable D3.4b.

## 4.2 Revision between D3.4b and D3.4c

After trying out some examples with the VR authoring tool, it was decided to make some improvements. They will be reviewed next.

### 4.2.1 VRCRTs

The part of the GRAPPLE authoring tool for creating VRCRTs has been rewritten and provides two ways of creating VRCRTs. First there is a direct way that allows the author to enter the rule using Grapple Adaptive Language (GAL) code. This was already possible with the first implementation (see deliverable 3.4b). However, as GAL is a very technical language and authors may not know how to specify the rules in GAL, it was decided to provide some sort of editor that will allow entering a rule in an easier way using some more-high level language and then have it translated to the dedicated language of the adaptation engine such as the GRAPPLE Adaptive Language (GAL). Some pre-defined VRCRTs have been defined that can be used for any course. These VRCRTs are based on deliverable 3.4a. The above changes will be explained in more details in section 6.6.

### 4.2.2 VRCAM

In deliverable 3.4b, it was decided to use the same library and functions developed for the regular CAM tool to develop the VRCAM tool. The main idea was to provide the same sort of environment to the author when creating a VRCAM as when creating a regular CAM. However, when trying to create VRCAM examples, number of problems had to be faced. The first one was that the tool was slow, especially when a large model needed to be created. From a usability point of view this is a serious drawback. The second problem that was encountered was that it was difficult for an author to specify the desired adaptations for a VE or to understand how the VE would be adapted based on the CAM. The main reason for this is the declarative approach used in CAM and the fact that the CRTs are actually hiding how and when something is adapted. Especially in the case of VR where adaptations need to be carefully specified in order to avoid VE's that are not usable anymore, this turned out to be a serious problem. VR authors are also used to think in terms of scenarios that don't match with the declarative approach used in CAM. As a result, it has been decided to change the VRCAM part of the tool. How this is done will be explained in more details in section 6.7.

### 4.2.3 Compiler to GAL

The implementation of D3.4b has been extended by developing a compiler to the GRAPPLE adaptive language (GAL). This was developed in order to allow the translation of VRCRT rules specified using the high level language into GAL. This way it is also possible to provide translations to other adaptive languages in the future.

## 5 Description of the integration of the VR Authoring tool with GAT

This chapter will describe the implementation of the VR Authoring Tool in terms of its integration with GAT (Albert et al., 2009) (Hendrix et al., 2009). It will explain how the integration has been done in terms of exchanging data and the data models being supported in the current version of the VR Authoring tool.

### 5.1 Integration of the VR Authoring tool with the other components of the GAT tool

The VR Authoring tool is realised as a module that can be launched from the main GAT shell (Albert et al., 2009) (Hendrix et al., 2009) (Hendrix and Harrigan, 2009), which is integrated into a Web browser like Internet Explorer<sup>1</sup> or Firefox<sup>2</sup>. The main part of the VR authoring tool is written in Flex 3.0 (Noble and Anderson, 2008) and ActionScript (Lott et al., 2006).

The tool supports a previewer which is realised using a freely available VR player (Vivaty player<sup>3</sup>) and it makes the connection through the Scene Authoring Interface (SAI) (Brutzman and Daly, 2008) and JavaScript (Flanagan, 1998). Note that also other X3D players supporting SAI can be used. The previewer allows an author to visualise an adaptation specified through the use of a Virtual Reality Adaptation State (VRAS). Figure 14 shows a high-level overview of how the VR Authoring Tool is called from the GAT shell and communicates data.

Note that although the VR Authoring tool can be directly called from the GAT shell, it can also operate as an independent module as it does not depend on the GAT shell to obtain and store information. Obtaining and storing information is done using the same web services as those used by GAT.

Figure 15 shows the different data models used in the VR Authoring Tool. All the files following those formats are stored through the available web services. The structure of these models will be explained in the next subsection.

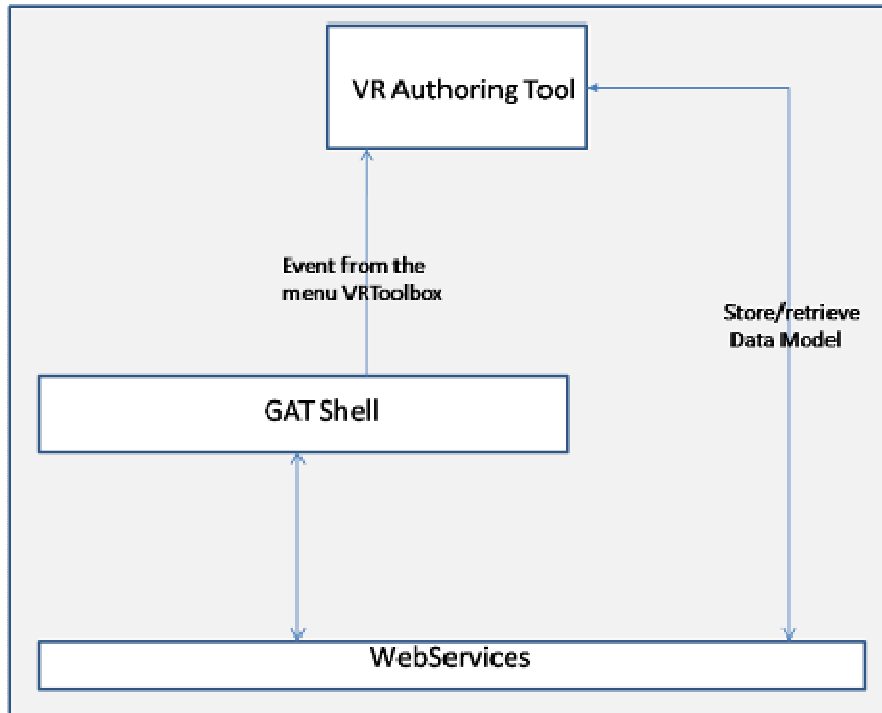


Figure 14. Relation between GAT and VR Authoring tool

- <sup>1</sup> [http://en.wikipedia.org/wiki/Internet\\_Explorer](http://en.wikipedia.org/wiki/Internet_Explorer)
- <sup>2</sup> [http://en.wikipedia.org/wiki/Mozilla\\_Firefox](http://en.wikipedia.org/wiki/Mozilla_Firefox)
- <sup>3</sup> <http://www.vivaty.com/downloads/player/>

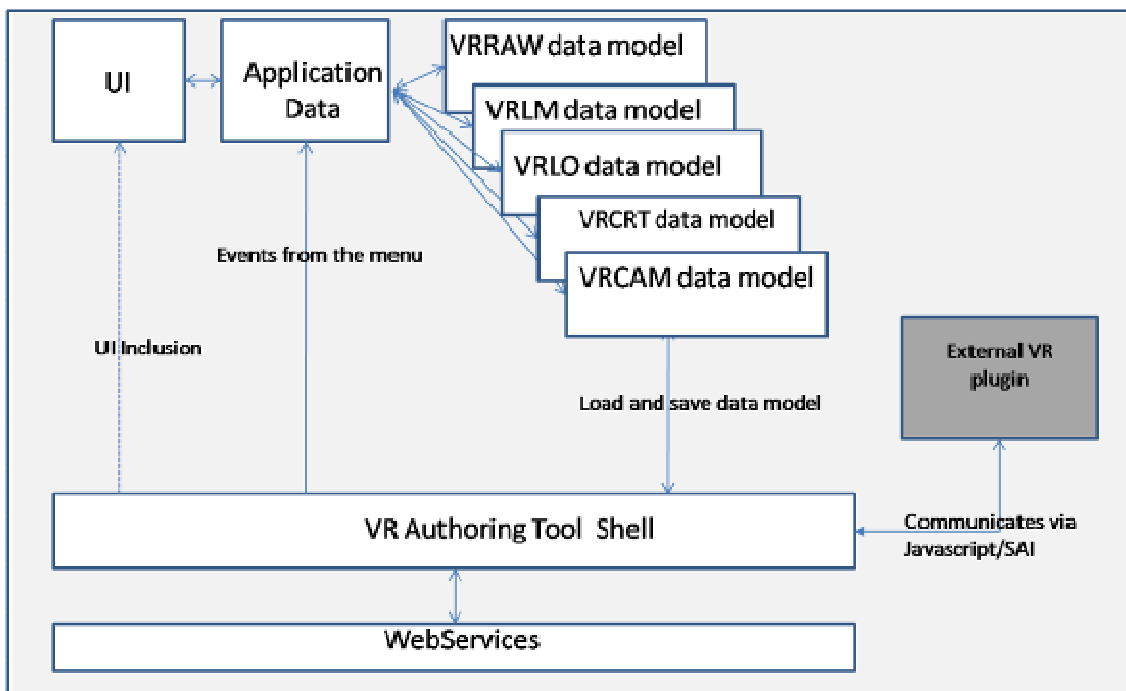


Figure 15. Architecture of a VR Authoring Tool with the different data models

In GAT, the VR Authoring tool is invoked by selecting the menu item VR Toolbox. Figure 16 shows how the VR Authoring tool appears once the author selected this option.

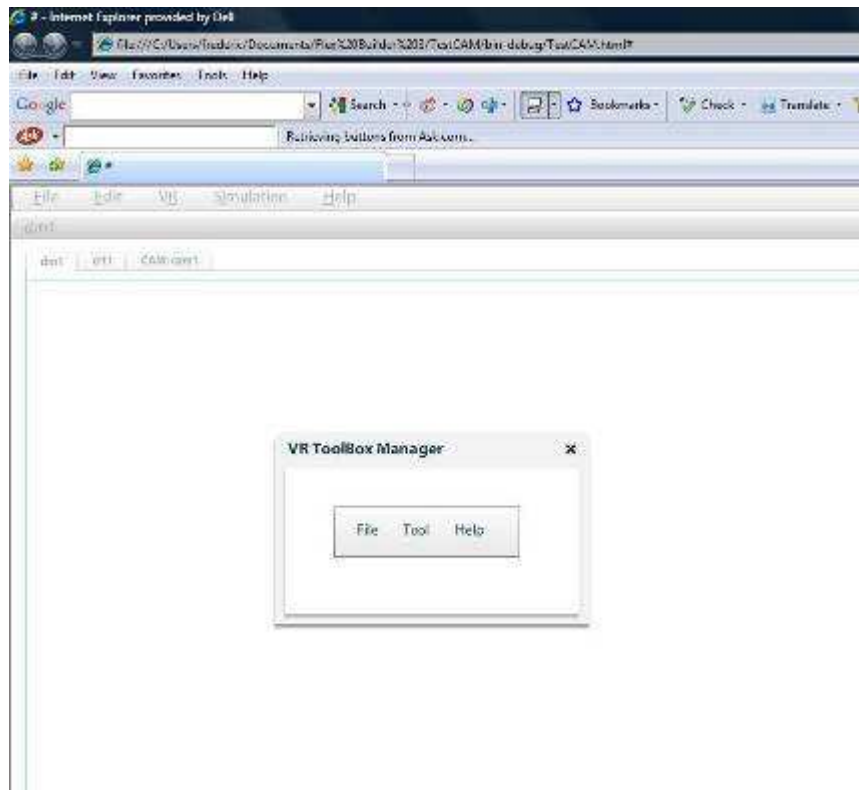


Figure 16. VR Authoring Tool invoked from GAT

The VR Authoring Tool itself is explained in chapter 6.

## 5.2 Integration of Data from the VR Authoring tool into CAM

As explained in deliverable 3.4a (Kleinermaun and De Troyer, 2009), a VR resource (or also called VR Learning Material) can either be included into a CAM as a *black box* (i.e. without adaptation) or with adaptation specified by means of a VRCAM. In both cases, we have to integrate the VR Learning Resources into the CAM. For this, we use the notion of *Entity* introduced in the CAM format (Hendrix et al., 2009) and also how normal CRTs are integrated in the CAM.

The term 'entity', in our case is used to refer to (1) a VR resource with no adaptations that may have been specified at the domain level or at the CAM level or (2) a VR resource (i.e. virtual world) with adaptation expressed with the VR authoring tool using a VRCAM. The applied approach entails adaptation within VR learning material by having VR objects with a unique ID (see also section 6.3) in a certain VRAS (VR Adaptation State). To express this in the VRCAM, the concept of Entity is also used. A VRAS can actually be seen as a VR resource, however, it is used especially for adaptation within a virtual world. Such a VR resource is described in XML and is composed as follows:

- a. idvw: represents the uuid of the VR resource (i.e. virtual world) containing the VR-object to which the adaptation applies.
- b. titlevw: provides the name of the virtual world
- c. idconcept : corresponds to the id of the related concept
- d. nameconcept : gives the name of a related concept
- e. idobj : corresponds to the id of the VRObjects that can be retrieved by the VR plug-in
- f. locationvr: provides the location of the file containing the virtual world
- g. idas: the id of a Virtual Reality Adaptation State (VRAS)
- h. tagid: the id of a node that can be retrieved by the VR plug-in to turn on/off a VRAS
- i. parameter: the X3D instruction to be used to turn on/off a VRAS

- j. value: the value to be used for the parameter
- k. type: the type of value i.e. integer, string, float, boolean

In deliverable 4.2b (Kleinermaun et al., 2010), it is explained how GALE can interact with the VR plug-in and communicates that VR resource to the VR plug-in which will then interpret it. Note that VRCRTs (like the normal CRTs) are expressed in XML and can therefore be integrated like normal CRTs in the CAM data model.

### 5.3 VRCAM

The VRCAM is like a regular CAM but dedicated to VR for creating adaptation within a virtual world. A VRCAM can be integrated into a CAM when one wants to mix VR Resources including adaptation and non-VR resources. However, VRCAM can also be used standalone. It has the same format as the regular CAM but in a more restricted way. It also uses the notion of socket where socket can have a concept id and location of VR Resources.

### 5.4 Repository for VR

A repository is used to store all the VR resources and can be accessed by the VR plug-in. A VR resource can be a VR resource in terms of learning content or a VR resource that is used for GALE to communicate to the VR plug-in what to change in the displayed scene.

### 5.5 Specification of the VR data models

In order to integrate the different components of GAT, it was decided that all the models should have a certain format. The following section explains this.

#### 5.5.1 Common header

The following XML code outlines the common structure of GRAPPLE authoring tool models. As outlined in deliverable 3.3b (Hendrix and Harrigan, 2009), each model has a common header part, where Meta-data of the model is located. Meta-data include title, common description, creation and update time, author, authorisation, and the UUID of the model. Below is a short outline of the common parts of the XML format.

```
<model>
  <header> //the header is shared between all models
    <modeluuid>660e8400-e29b-41d4-a716-446655440000</modeluuid>
      //the modelUuid is a unique identifier for the model.
    <modeltype>DM</modeltype>
      //Types can be DM, CRT, CAM or their VR and Simulation equivalent
    <authoruuid>660e8400-e29b- a716-41d4-446655440000</authoruuid>
      // the author UUID is the unique identifier of the author, who created the model
    <authorisation>readwrite</authorisation>
      // authorisation for all other author; the original author always has all permissions.
    <creationtime>2002-10-10T17:00:00Z</creationtime> //the timestamp of creation of the model
    <updatetime>2002-11-10T17:00:00Z</updatetime> //the timestamp the model was last updated
    <title>sun-example-dm</title> //the title of the model
    <description>sun-example-dm</description> //the description of the model
  </header>
  <body>//the body contains the actual content of the model. For VR it can VRAdaptation States, VRCRT and VRCAM

  </body>
</model>
```

The different VR models also have this structure. However, each of the VR models itself will be between the tags `<body>` and `</body>`.

### 5.5.2 Data Model for the VRCAM

Virtual Reality Conceptual Adaptation Model (VRCAM) represents the adaptation model to be used to create adaptation within a virtual world. Each time a VRCAM is created, it is stored in a VRCAM data model that follows the general data model format.

The visual editor of VRCAM uses the same library as the one used for the regular CAM so that the author does not have to learn a new visual language. It also stores the VRCAM into a format similar to the one used for the CAM. Please see appendix 10.1 for the XML specification.

### 5.5.3 Data Model for the VRCRT

Like CRTs are used to specify adaptation based on relationships between concepts, VRCRTs are used to specify when an adaptation inside VR should happen. Like a CRT, they use the concepts of socket, user model variables, as well as GAL code. They are stored in an XML format. The XML specification is given in appendix 10.2.

### 5.5.4 Data Model for VRAS

The tool stores Virtual Reality Adaptation State (VRAS). They are used to store the adaptation state a VR object can be in. They also store information on how to turn that state *on* and *off* and how it can be customised. More details will be given in the next chapter. VRASs are stored in XML format and have a general data model format (see section 5.5.1). They have a header and a body. Their XML specification is given in the appendix 10.3.

### 5.5.5 Data Model for VRRaw

To create adaptation in a virtual world, the author has to import this VR resource (which is in X3D format in our case) so that it can be pre-processed to prepare it for adaptation. VR resources are also being stored into the general data model format explained in section 5.5.1. This data model is called Virtual Reality Raw data model (VRRaw) since it is not yet processed. The XML specification is given in the appendix 10.4.

### 5.5.6 Data Model for VRLM

To identify the VR objects that can be subjected to adaptation, the author should upload VRRaw resources and then he can define which VR objects can be subject to adaptation. This means that in large virtual worlds there can be a lot of VR objects. Adaptation may only be needed to some of them. Such a VR resource (with VR objects identified) will be stored into a data model called Virtual Reality Learning Material (VRLM). The XML specification is given in the appendix 10.5.

### 5.5.7 Data Model for VRLO

In the used approach, the author can upload a VR resource expressed as VRLM data model (with possible VR objects for adaptation being identified) to define VRASs for these VR objects. The result of this is then stored in a data model called Virtual Reality Learning Object (VRLO). The XML specification is given in the appendix 10.6. Chapter 6 and chapter 7 will explain how an author creates such VRLOs. Note that the VR resources of type VRLO will be used by the author to create a VRCAM.

## 6 Description of the VR Authoring Tool

The VR Authoring tool component is invoked by the user from the general GAT shell by selecting the VR menu. Then, a VR Toolbox manager appears as a popup window (see figure 17). The VR Authoring tool component itself has three menus that are the *File menu*, the *Tool Menu* and the *Help Menu* (see figure 17).



Figure 17. VR Toolbox manager for the VR Authoring tool

- The *File* menu has five menu items; *New*, *Open*, *Import X3D*, *Insert VRCAM* and *Exit*.
  - The *New* menu item results in a menu that allows the author to create Virtual Reality Adaptation States (VRAS), Virtual Reality Concept Relation Type (VRCRT), Virtual Reality learning Material (VRLM), Virtual Reality Learning Objects (VRLO) and Virtual Reality Conceptual Adaptation Model (VRCAM).
  - The *Open* menu item results in a menu that allows opening the above model types i.e. VRAS, VRCRT, VRLM, VRLO and VRCAM.
  - The *import X3D* menu item allows the author to import new VR Materials in X3D format. They are stored as *VRRaw Data Model*.
  - The *Insert VRCAM* menu item allows author to include VR resources into a regular CAM. A VR resource can be one with adaptation or without adaptation.
- The *Tool* menu (currently) has only one item called *Preview VR Resources* which calls a previewer for VR resources.
- The *Help* item provides some help to the author.

All of these menu items will be explained in the next subsections. If the author selects *File->Exit* or the *black cross* (upper right corner), he returns to the main GAT shell.

Next each component of the VR Authoring tool will be reviewed.

Before starting to explain in more details the different components of the VR Authoring tool, it is important to realise that when the author wants to create a new file using *File->New*, a window (see figure 18) is displayed first, in which the name of the file and a description can be entered. The author also needs to select the type of data model needed. According to the type selected, an appropriate window will be displayed once the button *Create* is pressed. When the author uses *File->Open*, a window (see figure 19) is displayed in which the author can select a file. Once a file has been selected, an appropriate window according to the type of the selected file will be displayed.

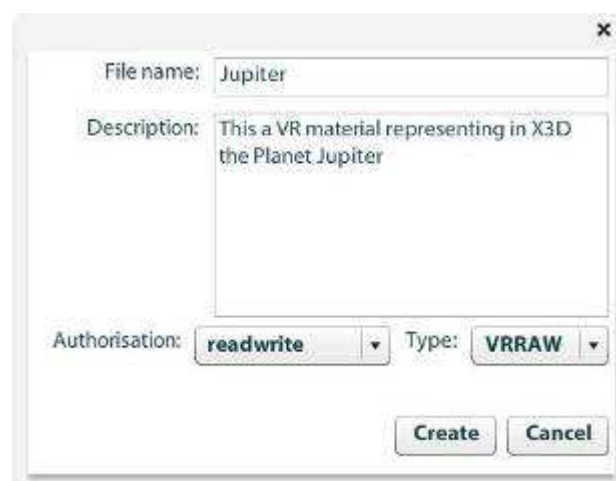


Figure 18. New File

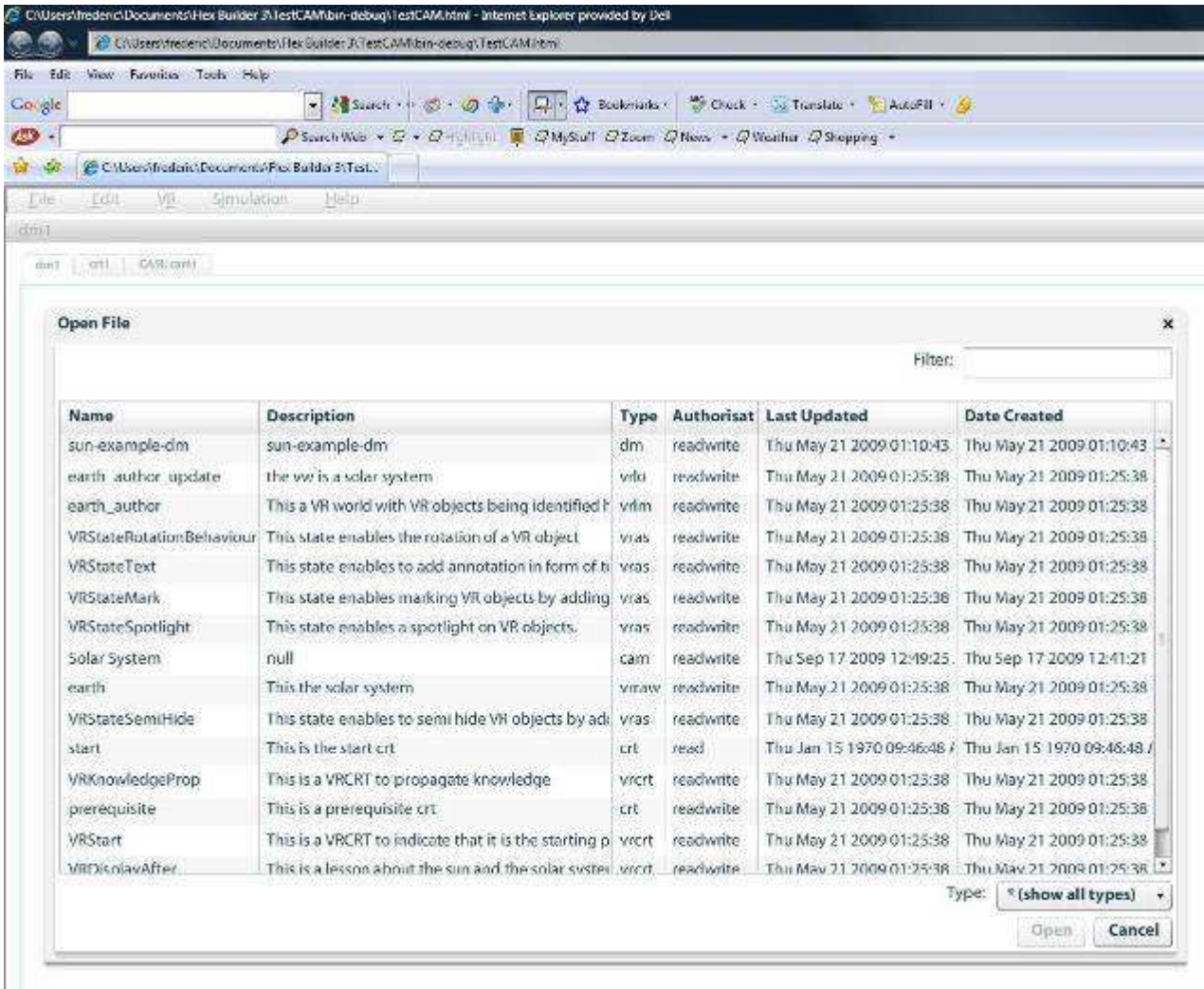


Figure 19. Open File

## 6.1 Previewing VR resources

The VR authoring tool allows the author to preview VR resources. An author can use this previewer component to see what certain VR resource looks like. If the resource is approved it can be decided to use the link to this resource (url) for adding the VR resource to a concept (by using the domain component of the GAT tool or by inserting the VR resource in a socket of the CAM using the CAM component of the GAT). The previewer is launched using *Tool->Preview VR Resources*. Figure 20 shows the previewer.

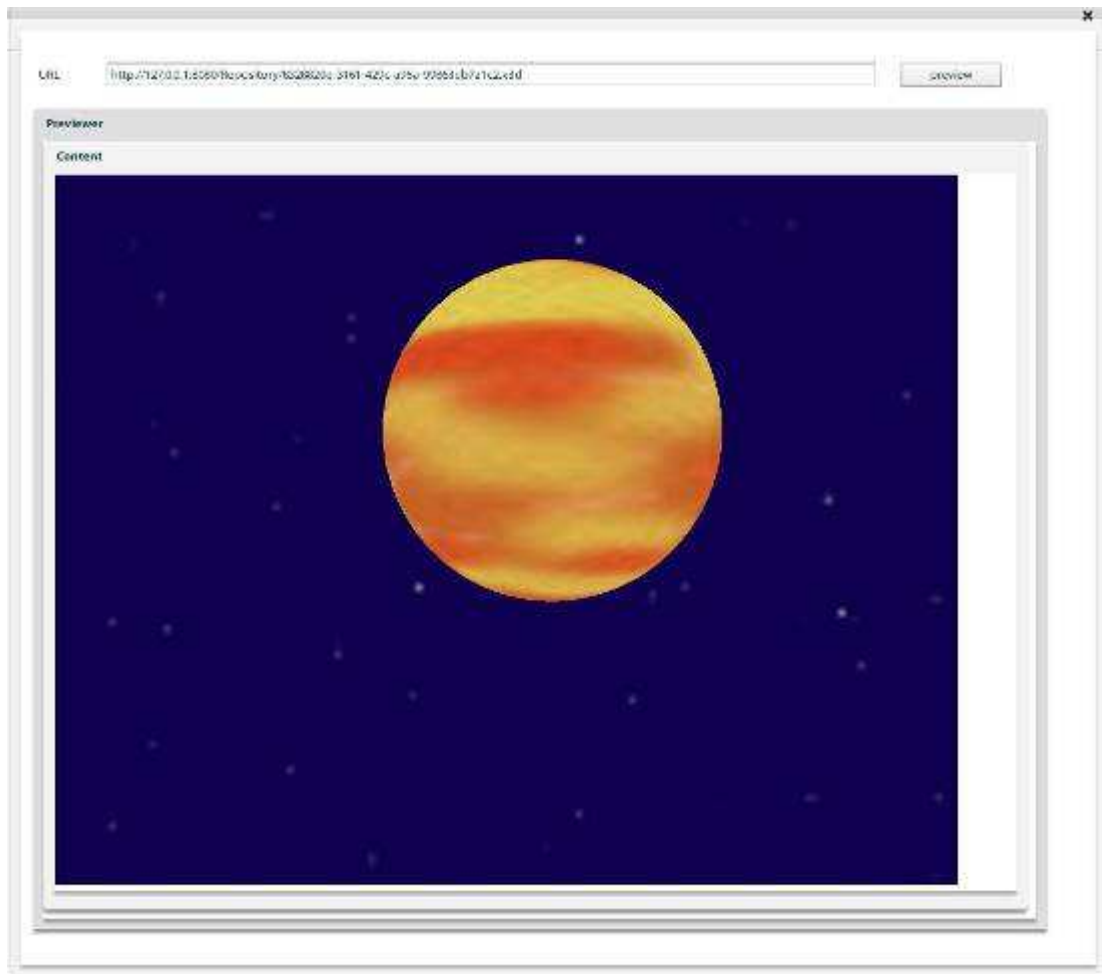


Figure 20. Previewing a VR resource

## 6.2 Importing Existing VR Resources

To define adaptations for a VR resource, the VR resource must be prepared for this. The first step in this approach is to provide the possibility to the author to import a VR resource (created by e.g. available VR content authoring tools (Murdock, 2010), (Chopra, 2009)). VR resources can be imported if they are specified in the X3D format. To import a VR resource, the author should use *File->New* and select VRRAW as type (see figure 21). The tool will then send an event by calling the appropriate UI (see figure 22).

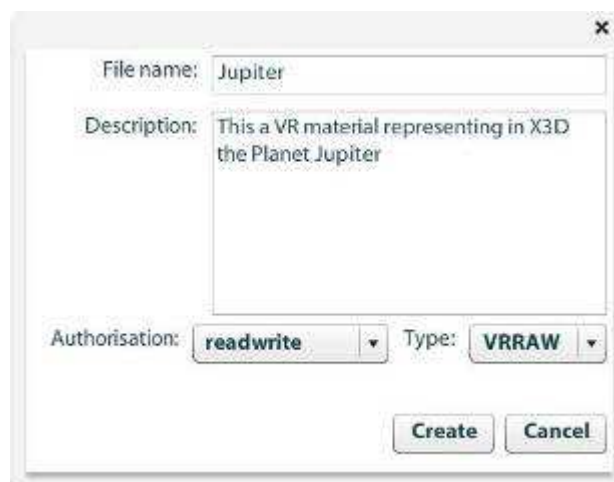


Figure 21. "New file" for importing VR material. It needs to be a VRRAW type

Given the fact that Adobe Flex does not allow to load files directly due to security reasons, the author has to copy/paste the X3d code into the window presented in order to import it (see figure 22). This is possible as the X3D format is an XML format.

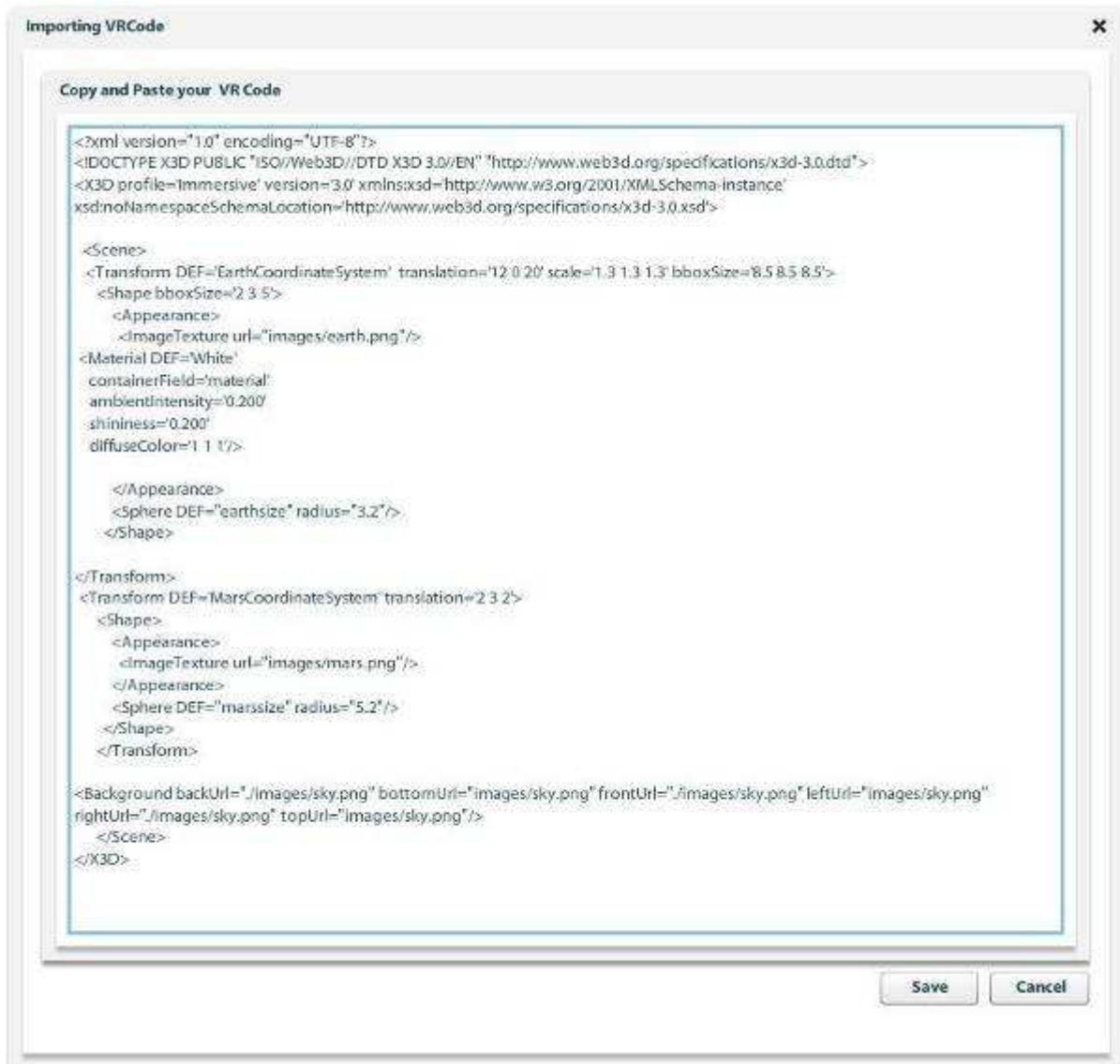


Figure 22. Copy and Paste X3D Code

However, as this is not very convenient, a tool has been developed with Adobe AIR technology, which allows an author to import the content of the file and store it in the database. The tool allows importing an X3D file, but also other files for the VR tool (see figure 23).

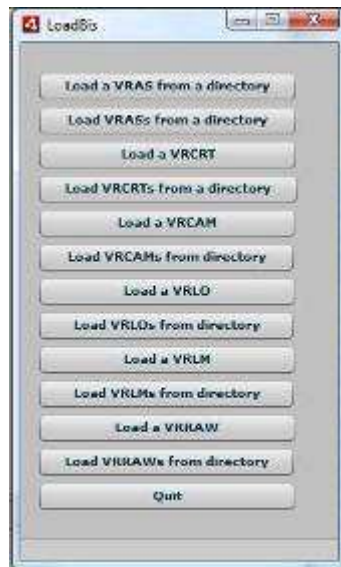


Figure 23. Loader

### 6.3 Preparing VR Resources

In order to be able to adapt VR objects in a virtual world (a VR resource), it should be possible to identify these VR objects, e.g. by means of an identifier (ID). VR resource in X3D format can be created with different VR content authoring tools (Murdock, 2010), (Chopra, 2009) or found on the Internet. However, they may not always have IDs for the VR objects they include. If IDs are available, they may also not have meaningful names, at least not from the viewpoint of the course author. These IDs are important for an author as he needs to use them to map VR objects to the domain concepts of the course so that they can be used. VRCAM is designed. These IDs will also be used by the VR plug-in to know which VR objects should be adapted. The VR Authoring tool allows an author to identify VR objects by entering his own IDs. The tool will also make sure that the author does not enter the same ID for two different VR objects. Note that not all the VR objects of a virtual world need to be identified but only those that will be used for the adaptation.

It was decided to do the identifying through a GUI. As an X3D structure is a scene graph based structure (tree structure) (Brutzman and Daly, 2008), a visualisation of the scene graph is provided as a tree so that the author can select the node that he wants to give an ID. This tool provides two versions of the tree. The first one is the original scene graph represented as a tree (see left side of figure 24) and the second one is the same scene graph but with changed/added the IDs of nodes (see right side of figure 24). The middle part of the GUI (see figure 24) contains two text boxes. The top one displays the node ID selected by the author when he clicks on a particular node of the left tree. The bottom text box is an input box and allows the author to enter the ID that he wants to assign to the node. In Figure 24, the author has selected the node "EarthCoordinateSystem" and changed that name to "Earth". Note that each node for which the ID has been changed, a prefix "Author\_" will be added. This helps the author to see which one has been changed.

This VR Authoring tool component can be used in two modes, represented by two tabs in the GUI (see figure 24). The first tab *Identify VR objects* allows the author to identify VR objects through the GUI as described. The second tab *Console* allows an author to see the actual XML file.

Once the author is happy with the IDs that have been provided, the results can be stored by pressing the button *Save the X3D*. This will invoke the web services needed to store the file. It will have the extension ".vrlm" as it is stored as VRLM data model. To call this component from the VR authoring tool component, the author selects *Tool->Identify VR Learning Material*.

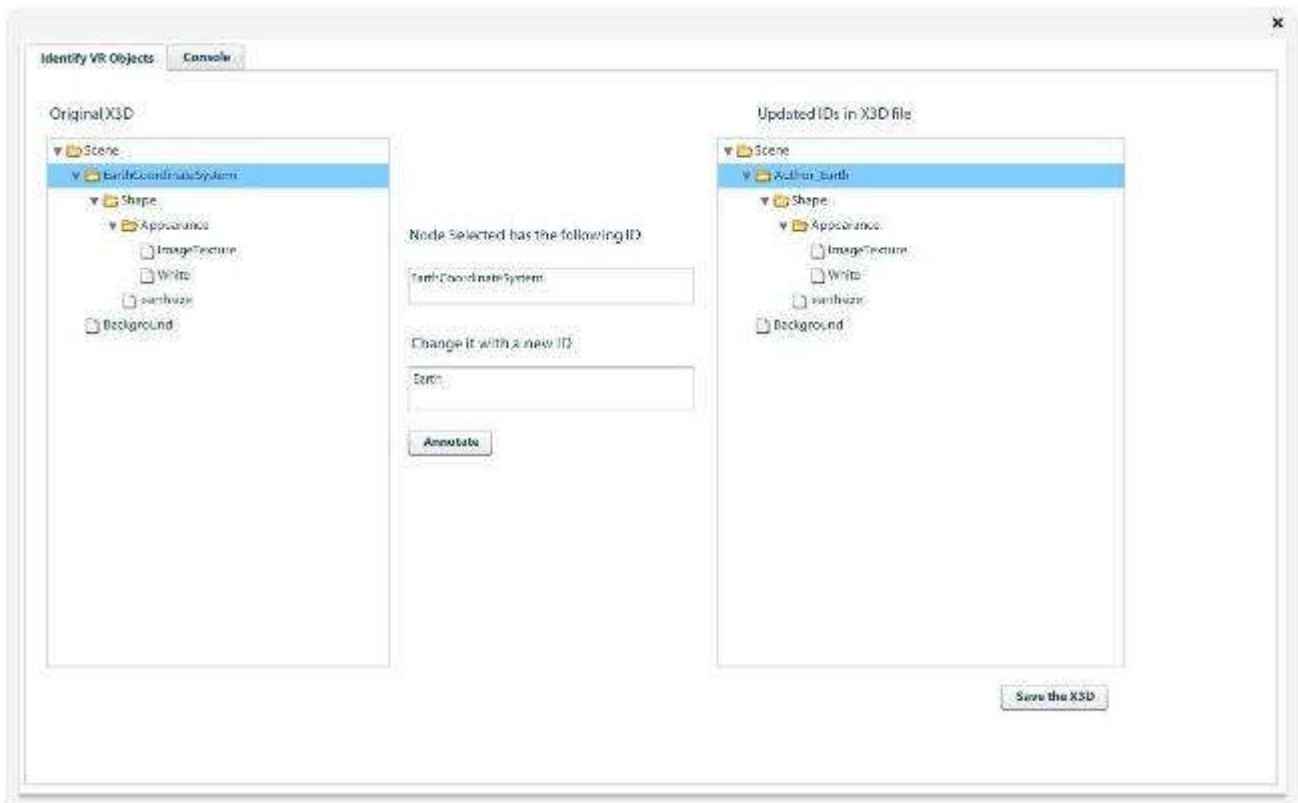


Figure 24. Identifying VR objects

## 6.4 Virtual Reality Adaptation States

Chapter 3 described what features of a VR object can be adapted. The VR authoring tool allows an author to specify these specifications through so-called Virtual Reality Adaptation States (VRAS). A VRAS allows an author to define the effect of an adaptation. For instance, the VRAS named *Marking* will “draw” a box around an object (actual the box becomes visible).

### 6.4.1 Defining a VRAS

For the actual adaptation, it is important that the VR plug-in knows how to trigger a VRAS. This is done by means of a TAG name and a trigger used to invoke or revoke the VRAS. Two values need to be specified for that trigger, i.e. one for turning *on* the VRAS and one for turning it off. This is illustrated with an example.

Suppose an author wants to introduce a new VRAS that makes a box around a VR object visible. For this the author selects *File->New* in the VR Authoring tool and then selects type *VRAS*. Next the VR code for the VRAS is entered, a X3D code for a box with some material properties. Then a TAG is provided that represents the node in the scene graph and which refers to the VRAS. This TAG will be used (later) by the VR plug-in to retrieve the node. In this example, the author enters the TAG *EnableVRStateMark* (see figure 25).

The trigger is specified as well as its values, for turning *on/off* the VRAS. Note that this trigger should appear in the X3D code as an X3D instruction. In the example, the author uses an X3D instruction as a trigger, called *scale* to turn the VRAS *on* and *off*. The value *1 1 1* is used for this parameter to turn the state *on* and the value *0.050.050.05* to turn it off. The type of these values also needs to be given; in this case the type is float.

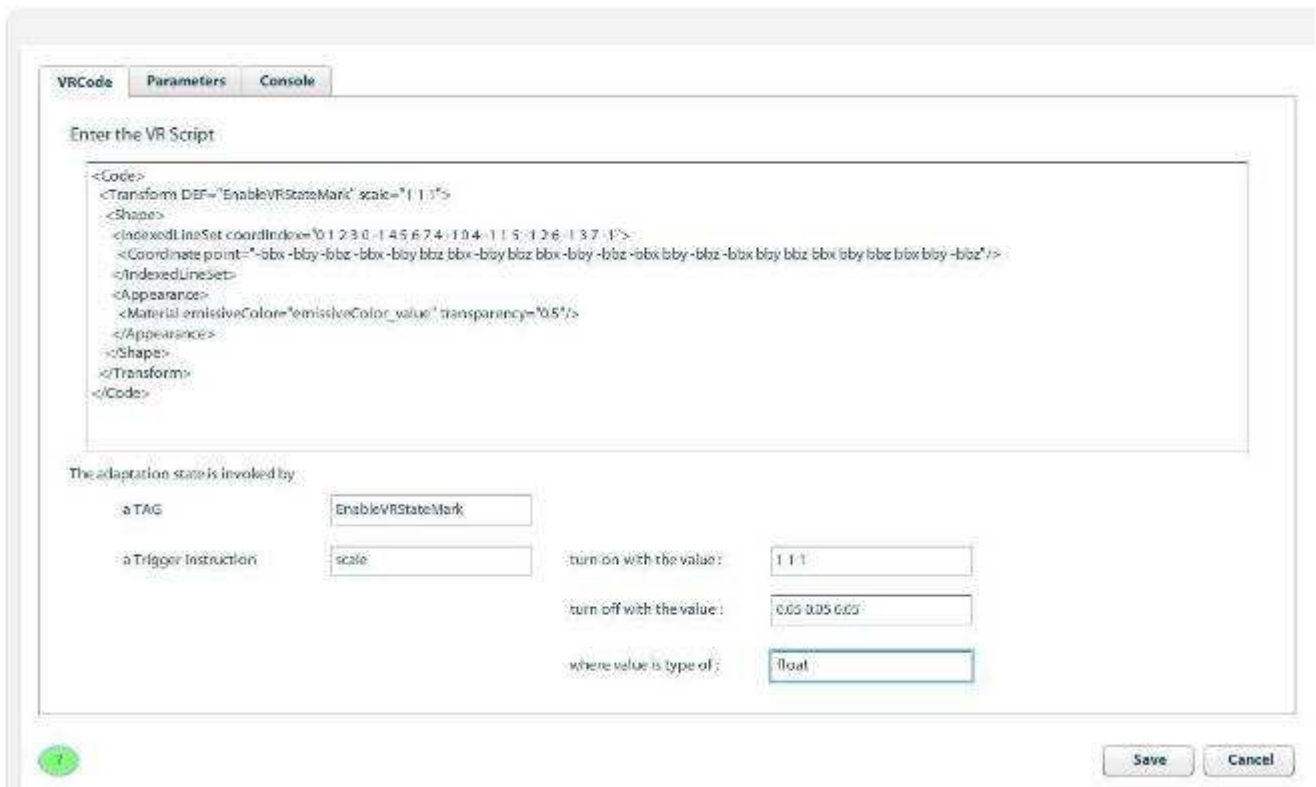


Figure 25. Adaptation State "Marking" with the VR Code and the instruction to turn it on/off

### 6.4.2 Allowing to customise a VRAS

Sometimes, it can be useful to reuse a VRAS for different VR objects. This is illustrated with the VRAS *EnableVRStateMark*. Since that VRAS adds a box around the object, it would be nice if the author could specify the size and colour of the box himself. The VRAS can be used for different VR objects and in different situations. The creator of that VRAS can add four additional parameters that can customise the VRAS when used; the size of the box along the X-axis, the Y-axis, and the Z-axis, as well as the colour. For each parameter, the default value has to be specified, as well as the type of parameter. Note that these parameters should refer to parameters in the X3D code of the VRAS. In figure 26, there are the parameters bbx, bby, bbz. These parameters can be changed by an author when using the VRAS to customise the VRAS. For instance, in this example, the size of the box can be changed. A description for each parameter can also be provided, which will help authors to understand how the VRAS can be customised. Figure 26 illustrates this.

The VRAS can be saved using the save button, with the ".VRAS". format

Note that the tab console can be used to enter the VRAS directly in XML. At any time, a VRAS can be changed or updated by selecting *File->Open* in the VR Authoring tool and selecting the appropriate VRAS file.

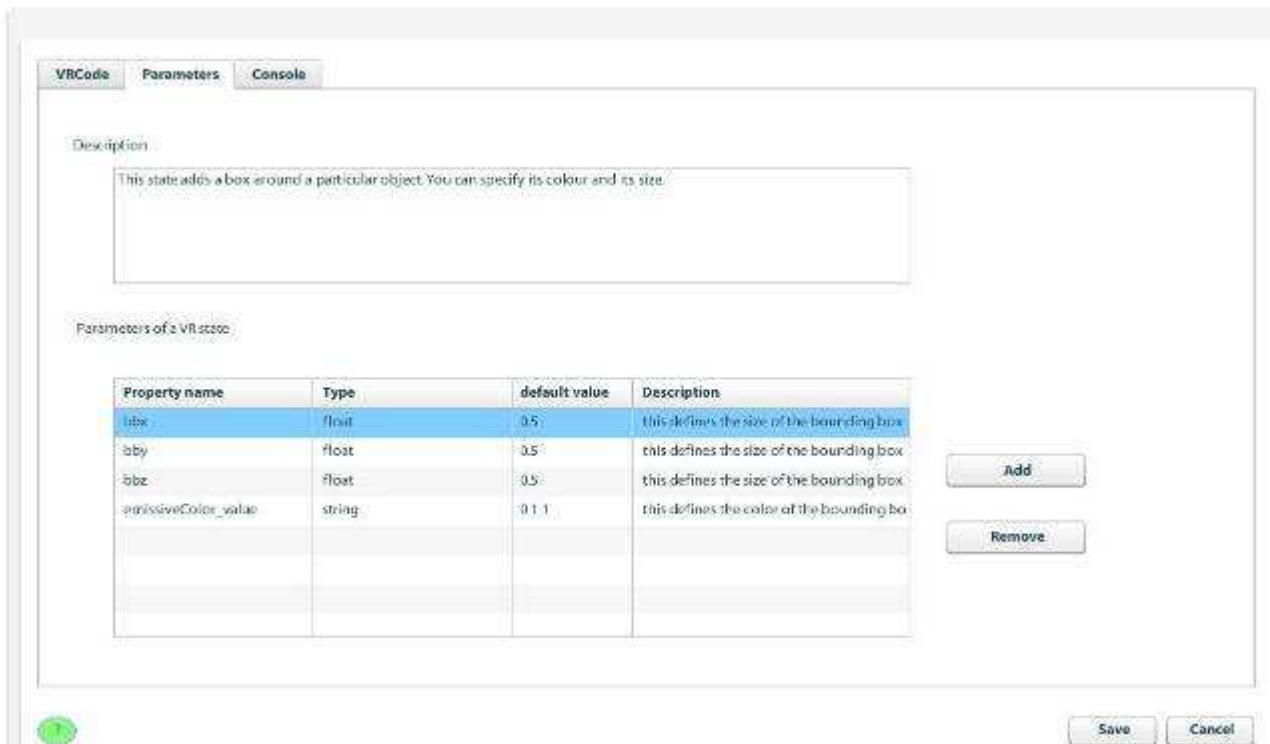


Figure 26. Adaptation State "Marking" with parameters

## 6.5 Preparing the VR Learning Objects

The three previous subsections explained how to import VR material, how to identify virtual objects in VR resources and the creation of VRASs. These are activities that should be done by someone who has some knowledge about X3D. The idea is to have a library of such VR resources and VRASs that is available to course authors so that they it can be used making adaptive E-learning courses using VR resources but without the need to have knowledge about X3D.

Before an author can specify an adaptive VR course (by creating the VRCAM), first the VR resource needs to be prepared. The author selects the VR resource for his course, adds a number of possible VRASs for the VR objects identified, and eventually customises a VRAS so that it fits the course. For instance, an author may want to use the VRAS *VRStateMark*, but would like to have the edges of the box in blue rather than red and the size of the box smaller since the VR object for which the VRAS is used, is smaller. The VR Authoring tool component allows this by providing a component for customisation.

Figure 27 shows the interface after the author has selected a VR resource that is a virtual world with only one identified VR object being the planet Earth. The VRAS *VRStateMark* for this VR object (using its ID "Author\_EarthCoordinateSystem") is selected and the the colour parameter customised. The adaptation can be previewed by pressing the button *Preview Adaptation State*. Once the author is satisfied, the VR Learning Object can be saved. This will be stored as a data model with the extension ".VRLO".

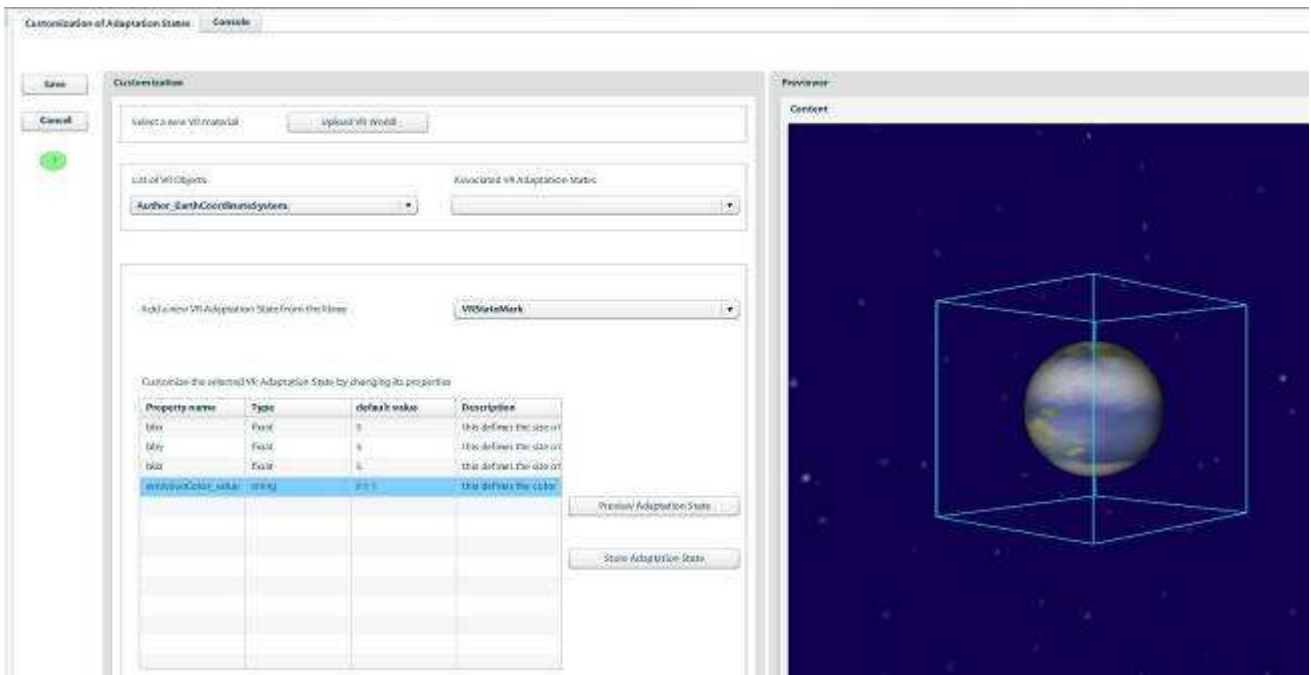


Figure 27. Customisation of Adaptation State and Preparation of Learning Object

Note that the author can use the “console” tab to enter the VRLO directly in XML. At anytime, the author can change or update a VRLO by selecting *File->Open* in the VR Authoring tool and selecting the appropriate VRLO file.

Figure 28 shows an overview of the steps to take when preparing VR material for adaptation.

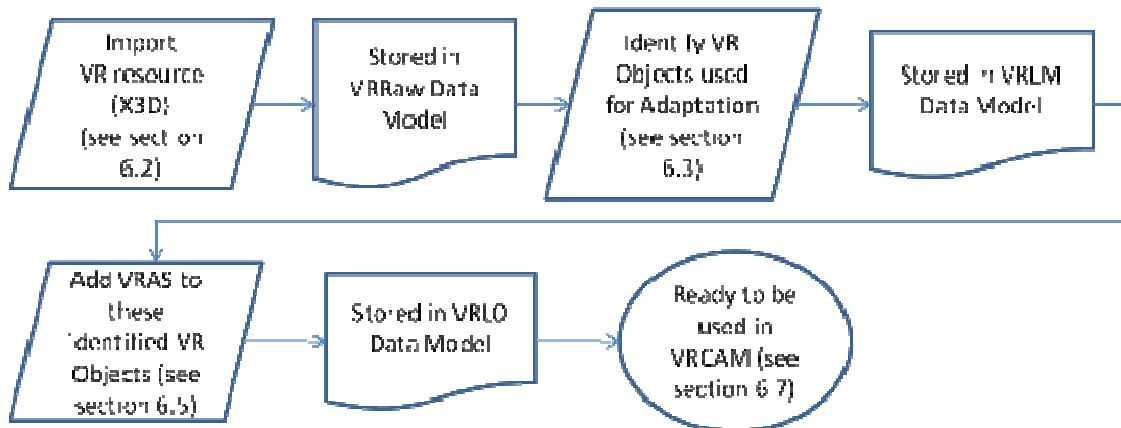


Figure 28. Overview of the steps to follow to prepare a VR resource for its use in a VRCAM.

## 6.6 Virtual Reality Concept Relationship Type (VRCRT)

The VR authoring tool component allows the authors to create their own Virtual Reality Concept Relationship Types (VRCRTs). VRCRTs follow the same principles as the regular Concept Relationship Types (CRTs) but they are dedicated to VR material. The tool has also been designed so that the way VRCRTs are created is similar to the way regular CRTs are created. This way an author does not have to learn a new method when using VR. VRCRT uses the concept of *socket* like CRTs (Steiner and Nussbaumer, 2009).

To create a new VRCRT, the author selects *File->New* and then selects the type VRCRT. A window will pop up in which there will be three tabs (see figure 29). Next each of them will be described.

### 6.6.1 The Main Tab

This tab is split into two parts. The first part allows entering general information about the VRCRT, such as the name of the VRCRT and a description. Two types of VRCRTs are possible: *Directional* and *Undirectional*. The author can also assign a colour to the CRT to make it more meaningful when used, for instance the colour blue for a CRT related to beginners, the colour red for a CRT related to novices, and the colour green for a CRT related to advanced learners.

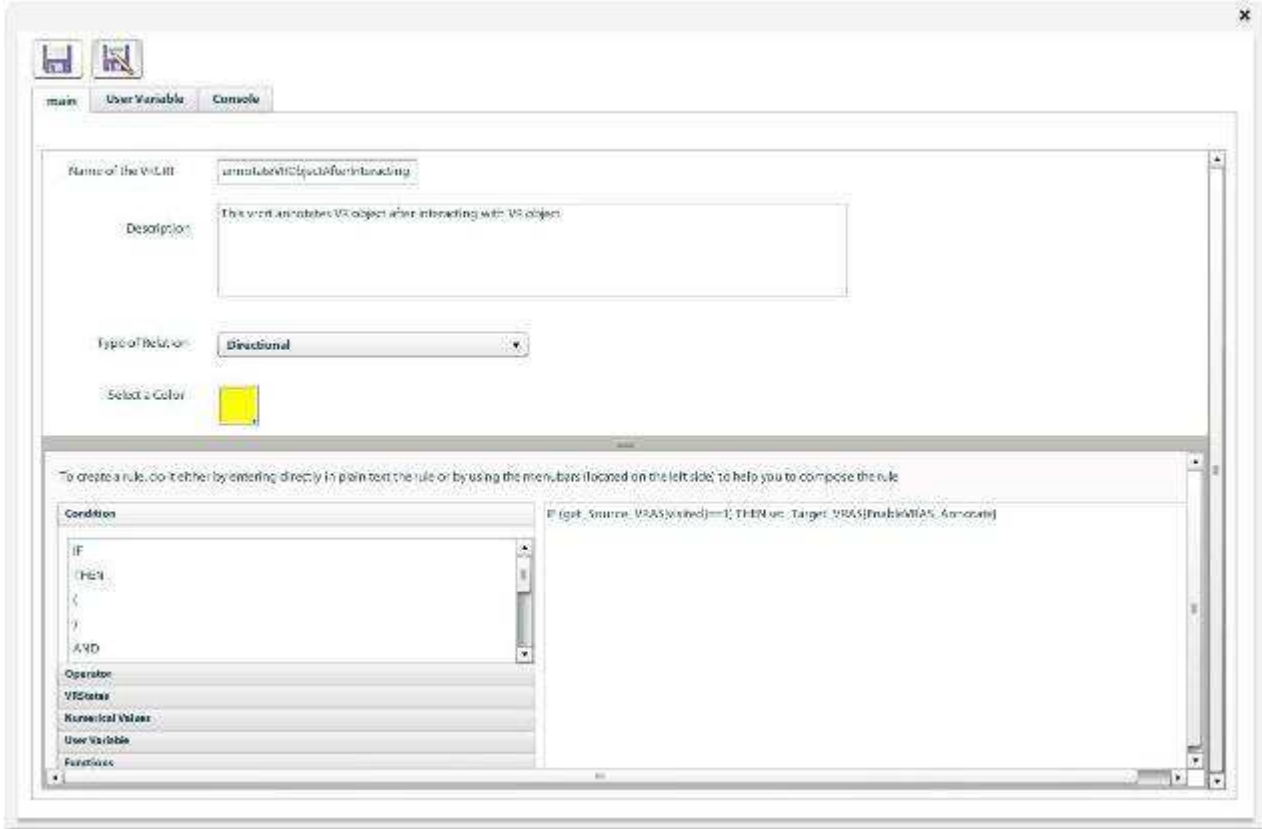


Figure 29. Main interface for creating a VRCRT

The second part of this tab is dedicated to the creation of the rule defining the semantics of the VRCRT. The author can either use the right panel to enter the rule directly, which assumes that the author knows the language to use. The author can also use the left panel that helps an author to compose the rule of the VRCRT (see figure 30).



Figure 30. Editor for creating a rule for a VRCRT

### 6.6.2 The User Variable Tab

The User Variable tab (see figure 31) allows specifying variables from the user model that will be used in this VRCRT. For instance, an example of a user variable is *knowledge* or *visited*. A user model variable has a

value and a type. It can also have a location telling where the information can be obtained. This will be explained in chapter 7.

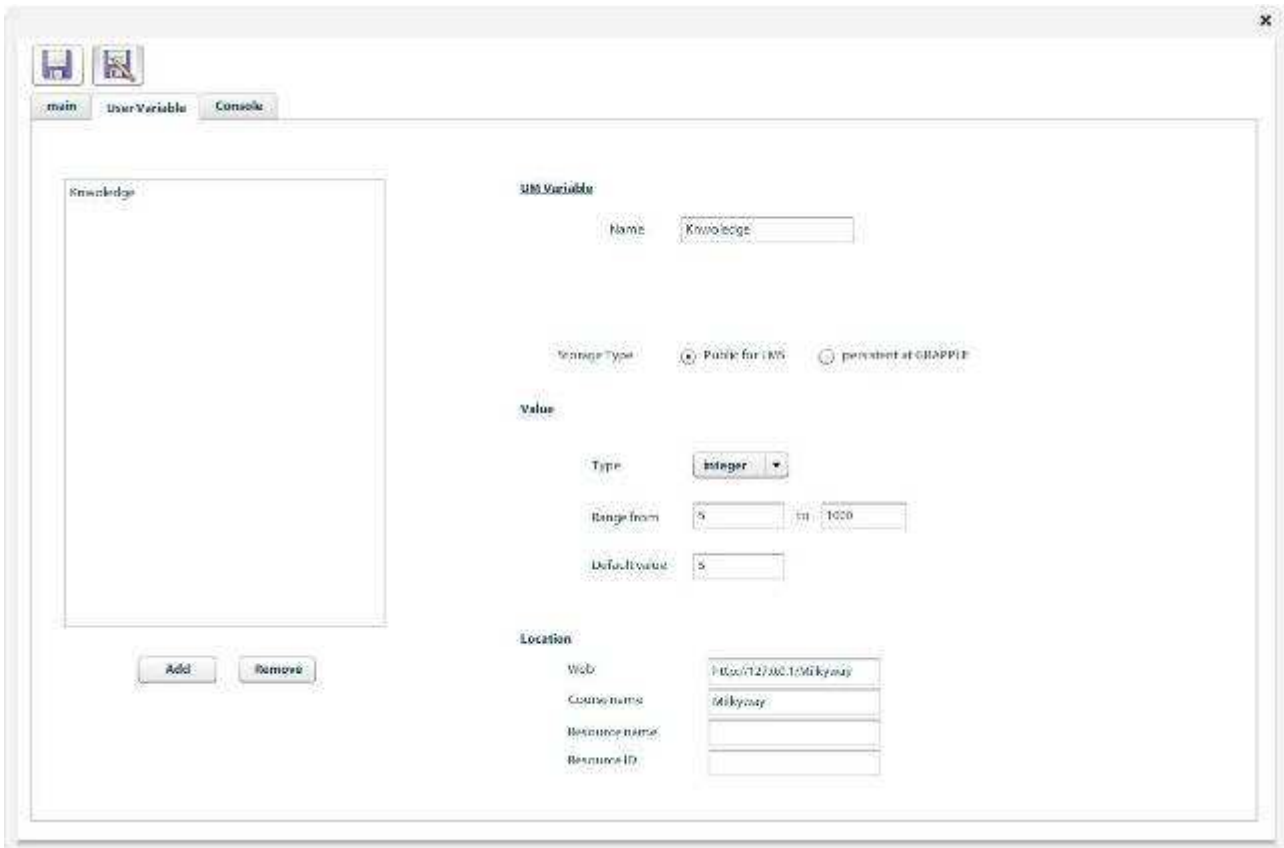


Figure 31. User Variable tab

### 6.6.3 The Console Tab

The console tab shows the VRCRT in XML format (see figure 32). The author can directly change the VRCRT by modifying its XML.

When the user saves the VRCRT, it will be saved as a data model of type ".vrcrt". Note also that the author can always read and change a VRCRT. In order to do this *File->Open* has to be selected in the VR Authoring tool and "*name of the file*" with the type ".VRCRT" has to be selected.

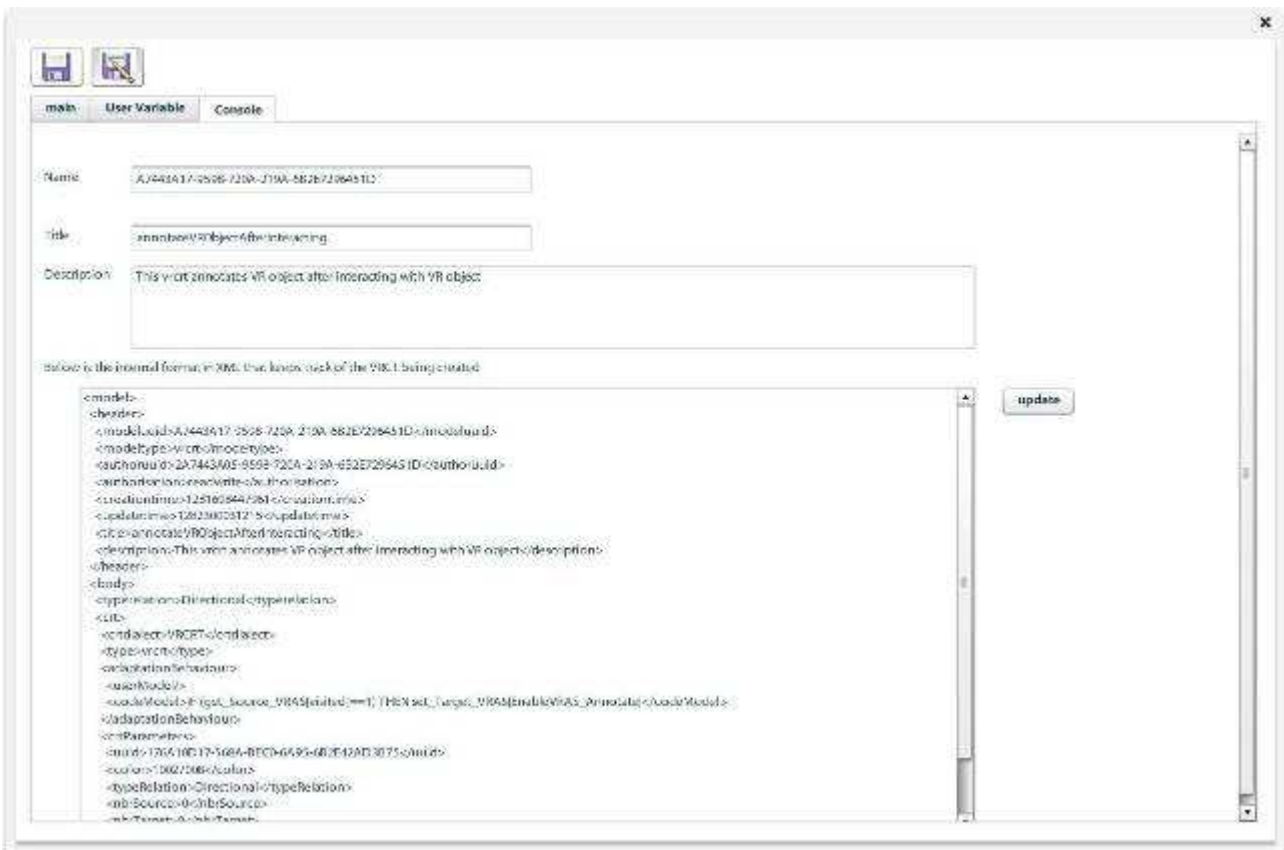


Figure 32. Console for VRCRT

## 6.7 Virtual Reality Conceptual Adaptive Material (VRCAM)

A Virtual Reality Conceptual Adaptive Model (VRCAM) allows the author to specify the adaptations required inside a VR resource.

Suppose that the author creates a CAM that uses a *prerequisite* CRT between the concept *planet* and the concept *earth*. However now he also would like to specify some adaptations within the virtual world . For instance, he wants to create a VR course that first highlights the planet Earth by marking it with a box around it and annotating it with its name in order to attract the attention of the learner. Once the learner has interacted with it by clicking on it, Earth should rotate around its axis. After that, when the learner has interacted with it again by clicking on it, Earth should start to move along its orbit around the Sun. After another interaction, the user will be taken to planet Mars which will be highlighted. The course then finishes by putting a red spotlight on Mars.

To create such a course, the author will launch the VR Authoring tool, select *File->New*, and create a new VRCAM by giving it a name, a description and selecting the type VRCAM (see figure 18). This will result in the following window (see figure 33).

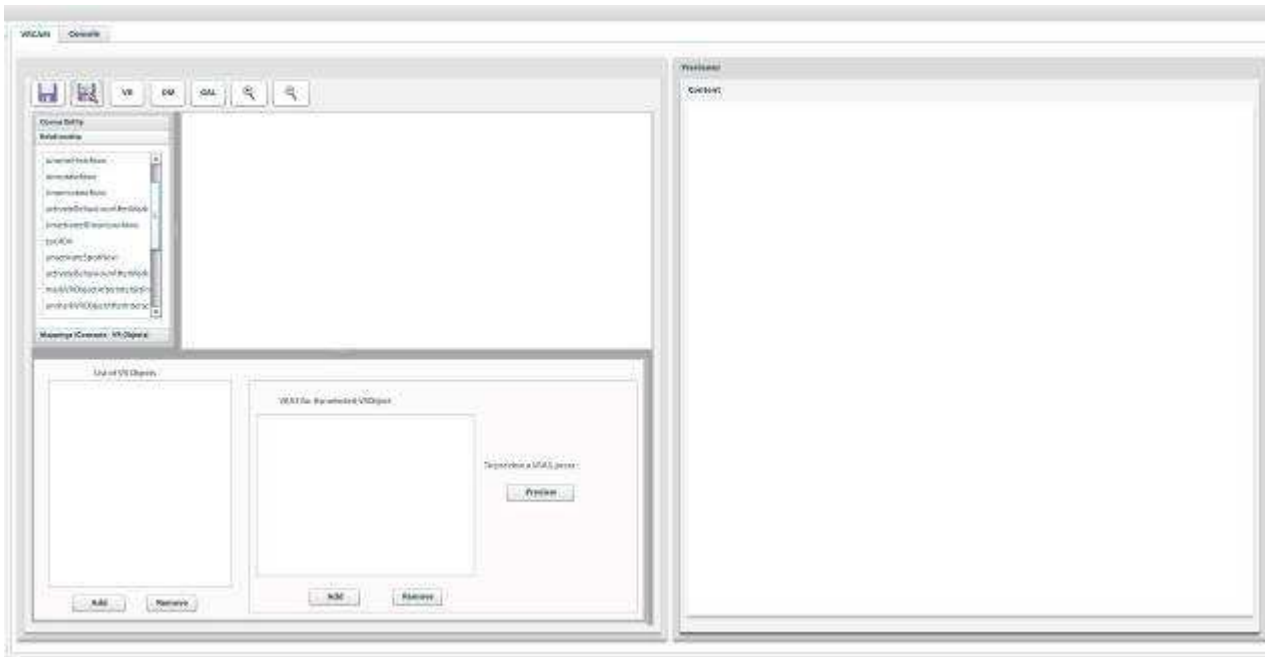


Figure 33: Main interface VRCAM

The author will see two tabs. The first tab is the main one and the second one is the console used to see the VRCAM in XML format.

The window for the first tab is split into two main panels. The left panel is used to design the course, i.e. the VRCAM. The right panel is used to preview the virtual world and to see how an adaptation will be visualised within the virtual world.

The left panel has three subpanels called *Course Entity*, *Relationships* and *Mappings*. First the roles of each of these subpanels will be explained.

The first subpanel *Course Entity* contains the generic building blocks that can be used to compose a VR course; a *start*, an *end* and an *adaptive* block (see figure 34).

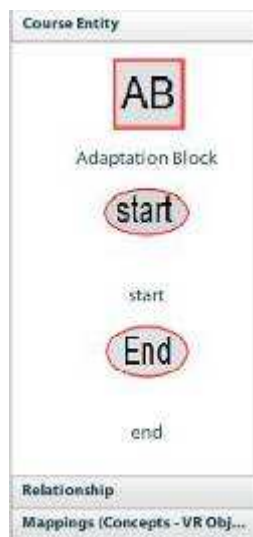


Figure 34. Course Entities

A *Start* entity is used to define how the course will start and also to give a number of variables a value if necessary, like suitability or knowledge. An *End* entity is used to specify when the course finishes. It can be used for setting or updating values in the user profile.

An *Adaptation Block* entity specifies an adaptation. It specifies the VR Objects that should be adapted and the adaptations to be used for these VR Objects. For instance, it can have the VR Object representing the planet Earth in the adaptation state (VRAS) that marks it.

Note that Start and End can be considered as special kinds of Adaptation Blocks.

The second subpanel *Relationships* is the one that deals with the VRCRTs (see figure 35). It provides a list of VRCRTs that have been created using the VRCRTs editor and which are available to the author for creating his course. These VRCRTs will be used to link the different Adaptation Blocks.

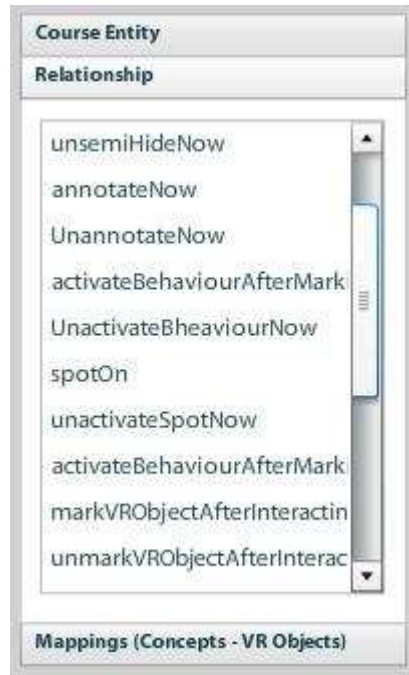


Figure 35. VRCRTs

The last subpanel is the *Mappings*. Using this panel, the author can specify the mapping between the VR Objects and the concepts of the domain model. For instance (see figure 36), the VR object called *Author\_Earth* is mapped to the domain concept *Earth* meaning that the VR object Author\_Earth represents the domain concept Earth.



Figure 36. Mapping VR objects to domain concepts

To create a VRCAM, the author will first select a domain model and a VR resource (i.e. a virtual world with VR objects being VR resources that are of type ".VRLO").

The author will now compose the scenario of the adaptations by dragging entities on the canvas and connecting them with VRCRTs. Starting with an entity Start, followed by Adaptation Block entities and ending with an End entity. Each Adaptation Block represents an adaptation step. For an Adaptation Block, the author needs to specify the VR objects involved in the adaptation step and the VRASs used. Figure 37 shows a complete VR course based on the scenario defined above.

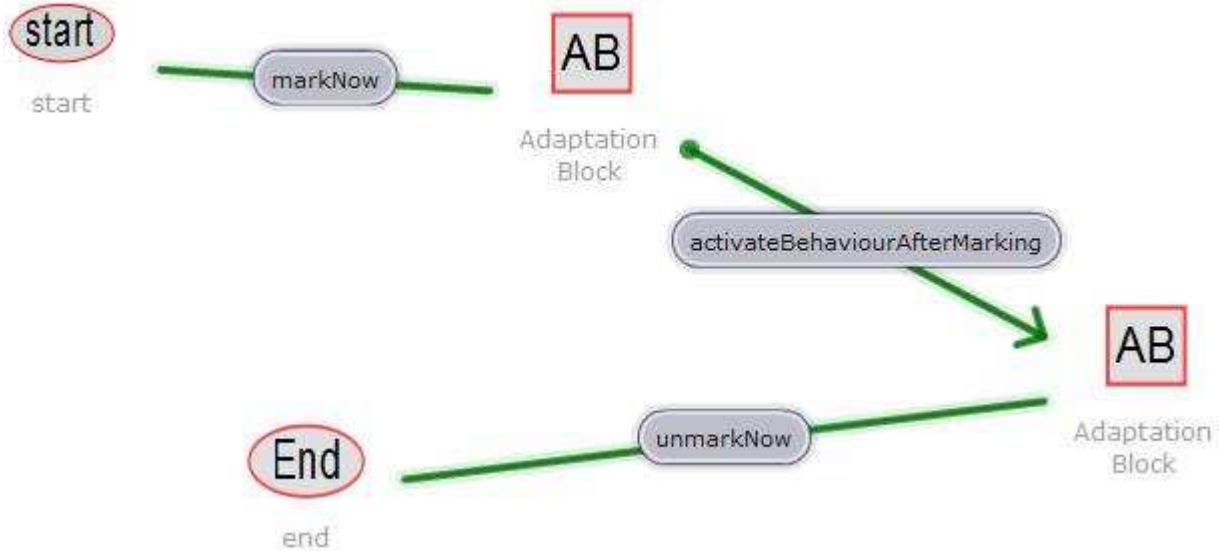


Figure 37. Example VRCAM

Figure 38 shows that when the author selects the first Adaptation Block, he can add the VR object he wants to be marked, in this case *Author\_Earth* and he then can select the VRAS that should be used for marking. In this case only one VRAS is used but there could be several. The author at any time can preview the VRAS (see figure 38 below, earth being marked by a blue square around it).

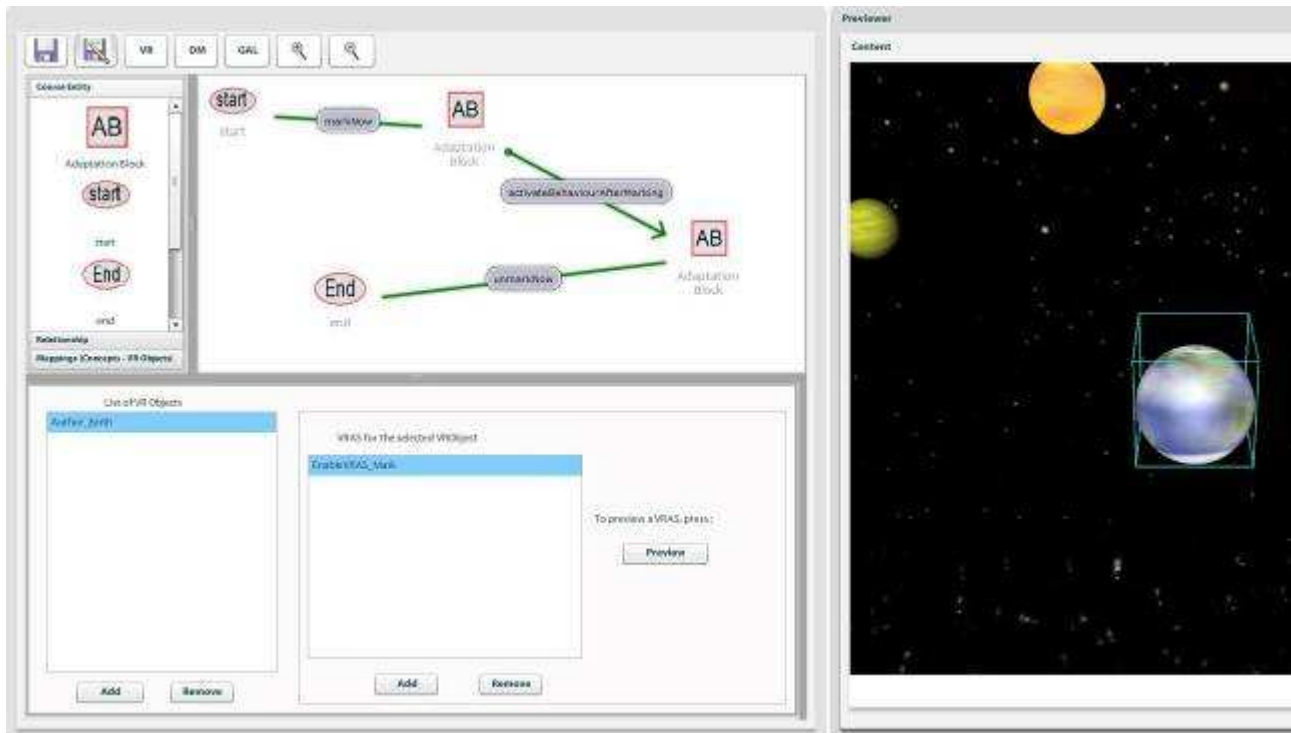


Figure 38. Previewing VRAS used in a VRCAM

By clicking on a VRCRT like *activateBehaviourAfterMarking*, the author can see the rule associated with this VRCRT (see figure 39). At anytime, the rule can be changed to fit the requirements. Note that the VRCRTs being dragged on the canvas become instances from the original defined VRCRTs, which can be considered as some sort of template.

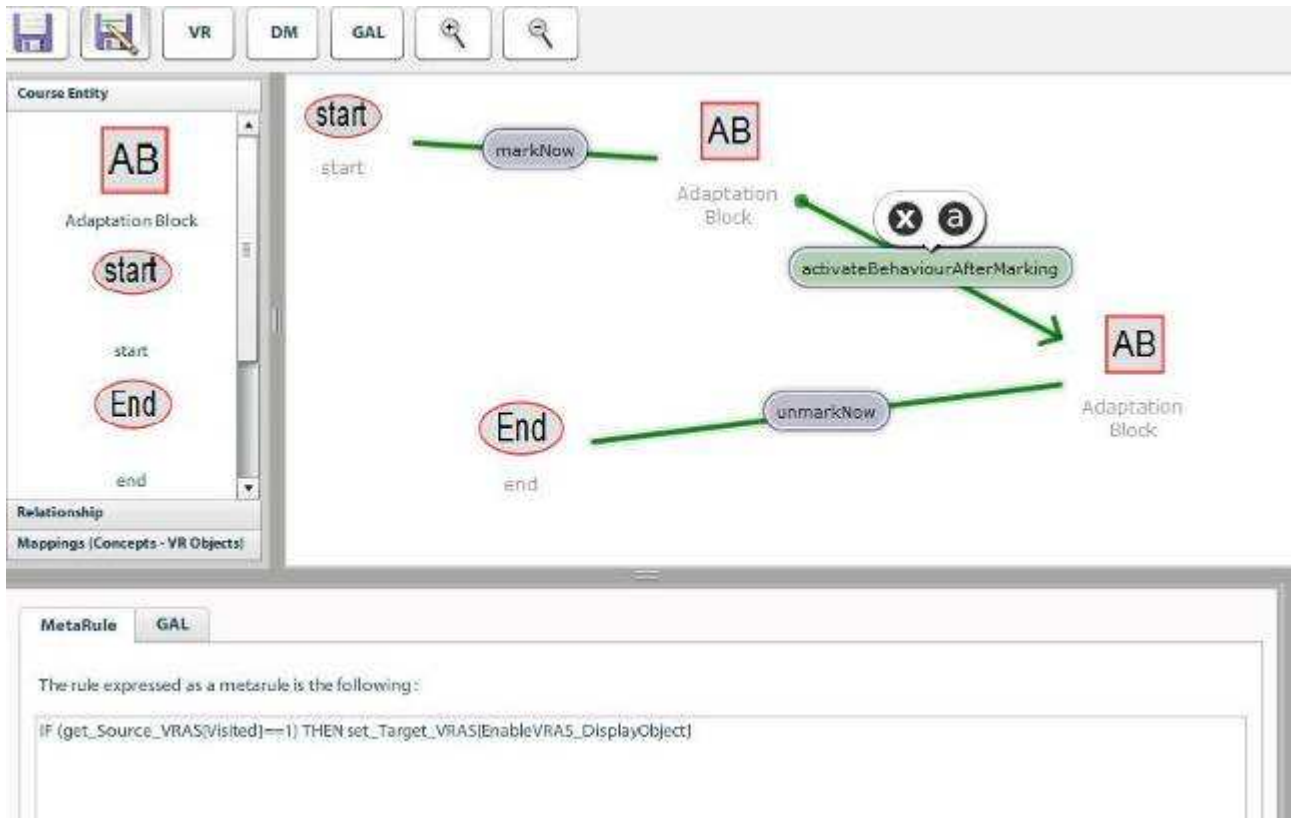


Figure 39. Inspecting a VRCRT in a VRCAM

## 6.8 Importing VR resources into a CAM

There are different ways for importing VR resources into a CAM.

### 6.8.1 VR resources attached to concepts using the domain tool component from the GAT tool.

Using the domain component from the GAT tool (see deliverable 3.1b (Dicerto and Oneto, 2009)), an author can attach resources (by means of URLs) to concepts. Such a resource can also be a VR resource. They can be inserted into a CAM once the user uses the concepts of a domain. Note that the author can always preview these VR resources by invoking the VR authoring tool and using the previewer (*Tool->Preview VR Resources*).

### 6.8.2 Adding VR resources directly into a CAM

The CAM tool component from the GAT tool allows an author to insert resources (having URL location) to sockets (see deliverable 3.3b (Hendrix and Harrigan, 2009)). This way the author can also add VR resources. In this case the author can always preview the VR resource using the previewer of the VR authoring tool (*Tool->Preview VR Resources*).

### 6.8.3 Inserting a VRCAM to a CAM

The VRCAM describes adaptation within a VR resource. It is possible to integrate such a VRCAM into a regular CAM. Special care has to be taken when inserting a VRCAM as it needs to be inserted completely into the CAM.

In order to do this the user needs to go to *File->Insert VRCAM*. This will result in a window (see figure 40) that will ask the user to choose the socket of the CAM in which the VRCAM needs to be inserted and also to choose the VRCAM to insert.

### 6.8.4 X3D previewer

To allow the author to see what a VR resource looks like, the author can preview it by going to the VR Authoring tool and selecting *Tool->Preview VR Resources*. A window will be displayed with the VR player embedded within the window and with a text box in which the URL location of the VR resource can be entered. By pressing the “Preview” button, the author can preview the VR resource (see figure 41).

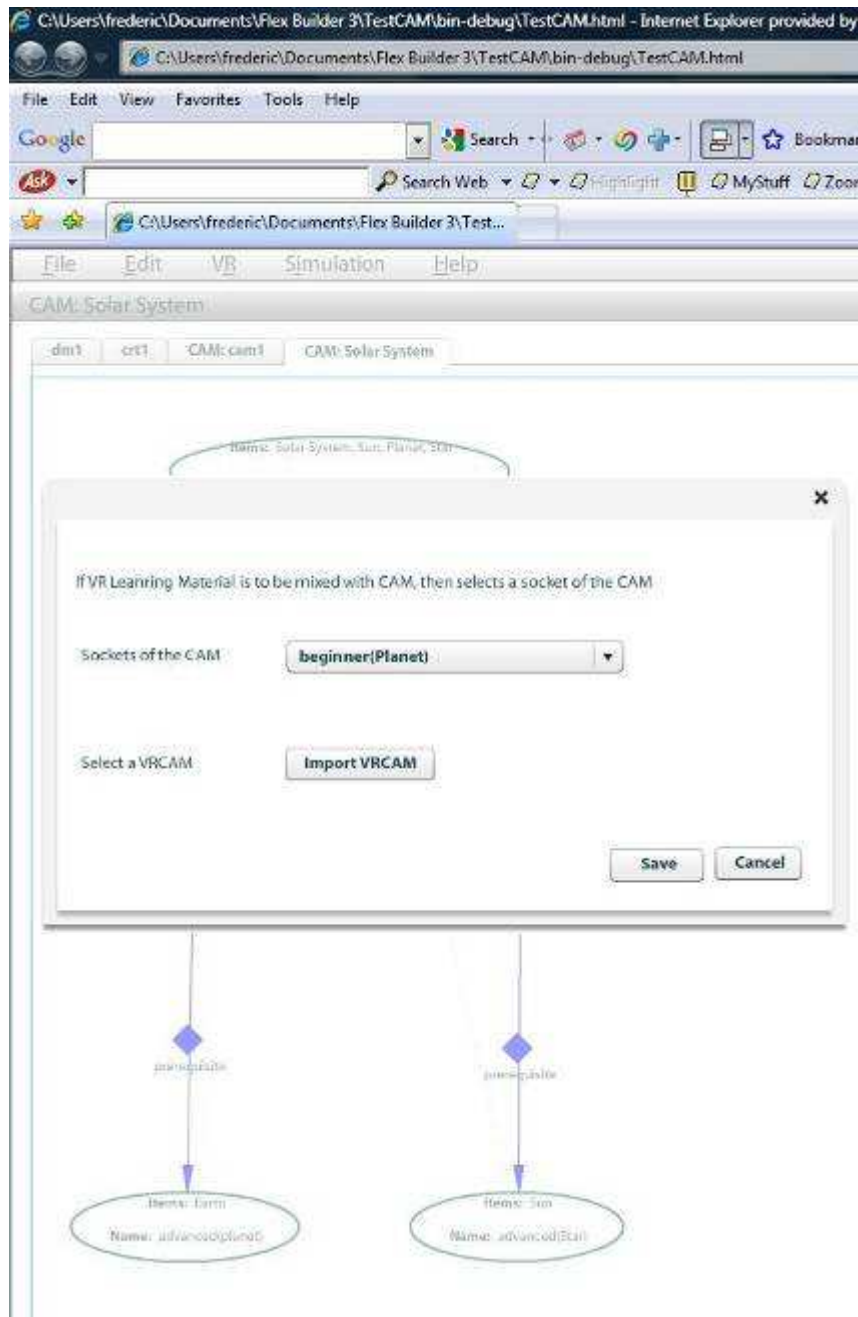


Figure 40. Inserting a VRCAM into a CAM

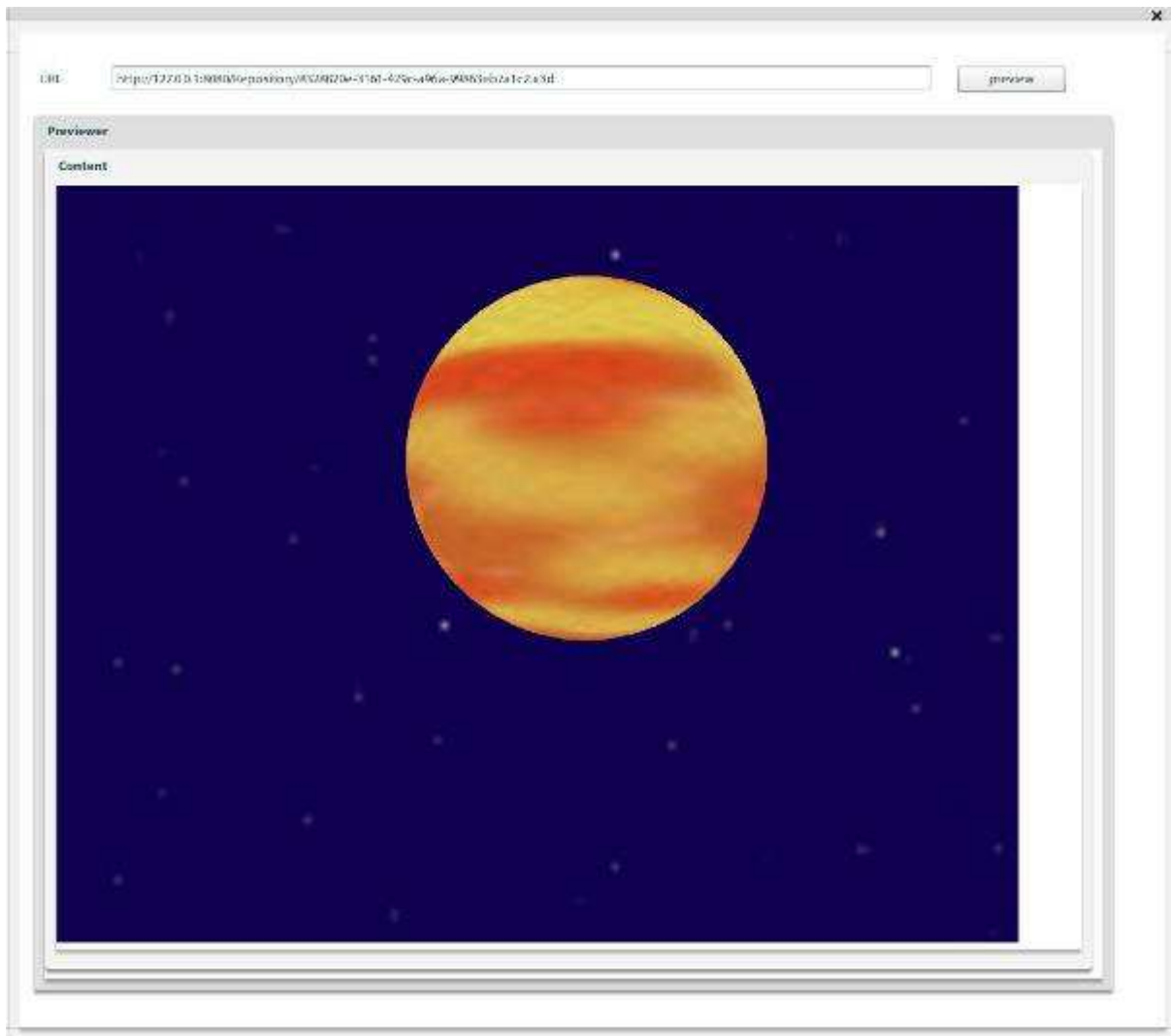


Figure 41. Previewer of VR Resources

## 7 User Guide

Chapter 6 described the VR authoring tool. This chapter will now provide an example of how the tool can be used for creating an adaptive course with some adaptive VR resources. The example course is about the solar system.

Suppose that the author first wants to give the student a general introduction on the solar system (this will be specified by a regular CAM). He also wants to illustrate a planet by showing the planet *Earth* in 3D. For this VR resource no adaptation is needed. Up until this point only regular (non-VR) adaptations will take place. At some point in the course, the author wants to show a virtual world with the planets *Earth* and *Mars* and the star *Sun* in order to provide the learner a notion of distance between *Earth*, *Mars* and *Sun*. To guide the student about what he still needs to study, the author decides to start with marking *Earth* by adding a blue box around it and annotate it by its name. After some interactions by the learner, *Earth* will first start to rotate around its axis and afterwards around its orbit. Once the knowledge about *Earth* is above a certain threshold, *Earth* will be unmarked, the learner will be taken towards *Mars* and *Mars* will now be marked. The course will end by highlighting *Mars*.

How this scenario can be realised using the VR Authoring tool will be explained next. The author should start by preparing the VR resources needed for his course. To do so, two steps need to be performed: first, import and prepare the VR resources needed (step 1, explained in section 7.1). Next, associate VR Adaptation States (VRASs) to the VR objects to be adapted in the course (step 2, explained in section 7.2). After these preparation steps VRCRTs (step 3 – section 7.3) can be defined optionally; also predefined VRCRTs can be

used (which will usually be case). VRCRTs are needed to create the VRCAM, which allows defining the adaptivity required for this VR resource (step 4 – section 7.4). Step 5 (section 7.5) and step 6 (section 7.6) deals with integrating the VR resource into a regular (hypertext-based) adaptive course.

## 7.1 Step 1: Import and prepare the VR resources needed

The first step that the author has to take is to prepare the VR resources needed in the course. It is assumed that no VR resources are imported into GRAPPLE yet. It is also assumed that the author has knowledge of X3D or that help can be asked of someone with X3D knowledge.

Using the above described scenario, the author needs a VR resource in a virtual world containing VR objects representing the planet *Earth* and *Mars* and a VR object representing the star *Sun*. To create this virtual world, a VR content authoring tool like (Murdock, 2010), (Chopra, 2009) can be used (creating the actual VR resources is outside the scope of the project) or found on the Web. The only requirement is that the VR resource used must be in X3D format. Suppose that such VR material is available

To import a VR resource, the author launches the VR Authoring tool and selects *File -> Import X3D*. This will result in the “New File” window (see figure 18) asking the author to enter a file name under which this resource should be saved, along with a description. VRRaw should be the data model type. The author can now upload the X3D code using the loader tool (see figure 42) and save the information. This generates the file with the given name and with type *VRRaw*.

Next the author needs to identify the VR objects in this VR resource for which adaptation is wanted. In order to do this select *File->New* to create a new file of type *VRLM* (VR learning material). This results in the “New File” window asking for the name of the file to be created and a description of the content of the file. The data model type should now be *VRLM*. By pressing the *Create* button a new window, the “Open File” window (like the one in figure 19), will be shown asking to select the file containing the required VR resource. In order to identify VR objects in this VR resource the *VRRaw* file saved in the previous step should be selected. A new window will now be displayed (see figure 43) in which the author can identify the VR objects needed, in this case *Earth*, *Mars* and *Sun*. All of the nodes representing the required VR objects need to be selected and given a meaningful name. Figure 43 shows the interface after the VR object representing planet Earth has been identified by the author. *Author\_EarthCoordinateSystem* (see the tree in Figure 43) can be seen as the ID for Earth. IDs that have been specified by the author receive the prefix “Author\_”. The same should be done for *Mars* and *Sun*.



Figure 42. The loader tool

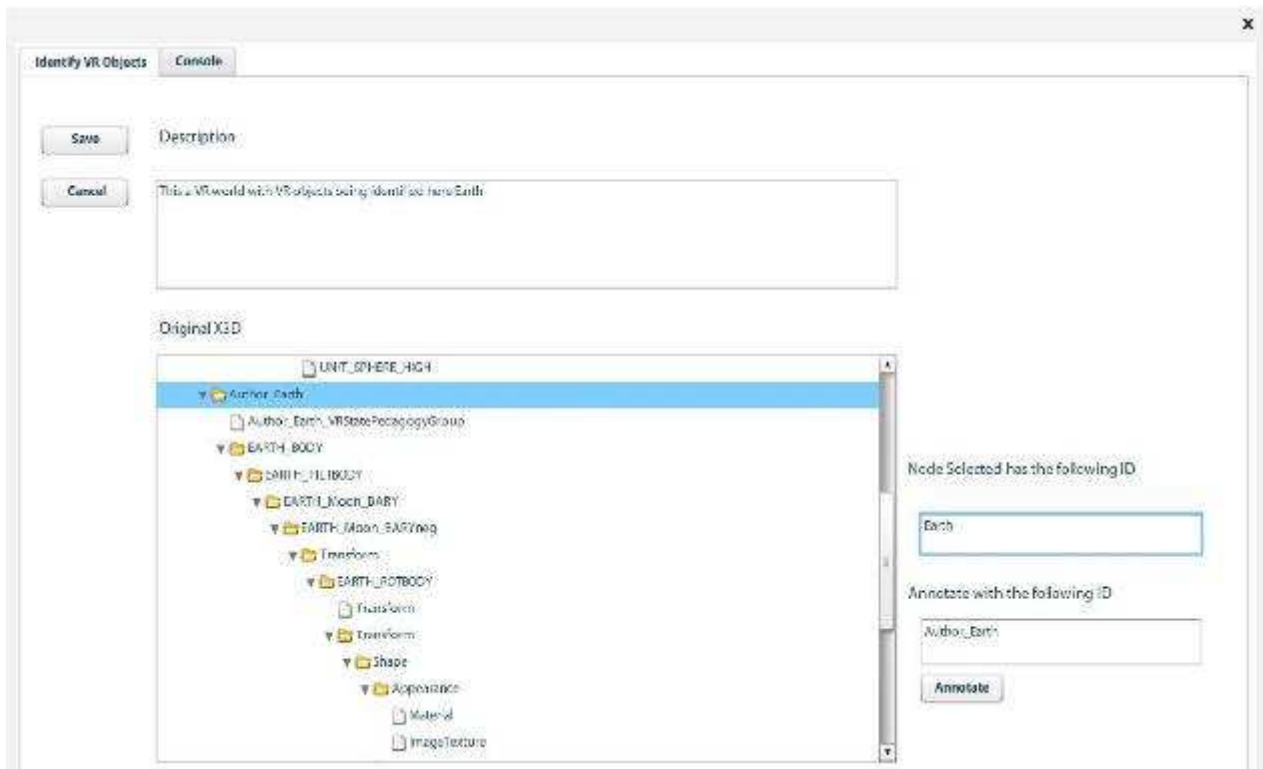


Figure 43. Identifying VR objects

## 7.2 Step 2: Add VRASs to VR Objects

To be able to specify some adaptations for a VR resource, the author needs to associate Virtual Reality Adaptation States (VRASs) to the different VR objects identified in the previous step. A VR object with an associated VRAS is called a VR Learning Object (VRLO). A number of VRASs are predefined but the author can define others. In this scenario, the author wants to be able to mark the VR objects Earth and Sun. An existing VRAS, called *VRAS\_MarkPedagogy* that puts a blue box around an object, will be used.

The author needs to select *File->New* from the VR Authoring tool to use the existing VRAS. This will result in the “New File” window asking the author to enter a name and description for the VR Learning Object to be created. The data model type should be VRLO. Next a new window will be displayed (see figure 44) where the author should select a VR resource of type VRLM (i.e. the VR resource in which the required VR objects have been identified (see previous step)) by pressing the *Upload VR World* button.

The author should now select the VR resource that has been prepared in the previous step (i.e. the virtual world with *Earth, Mars* and *Sun* as identified VR objects). These VR objects as well as the available VRASs will then be shown to the author by means of combo-box lists (see left side of figure 44). Using these combo-boxes, the author can associate a VRAS to a VR object. For instance, the VRAS *VRASMarkPedagogy* can be associated with the VR object *Author\_EarthCoordinateSystem*. When selecting a VRAS, the list of parameters associated with this VRAS will be shown (see left side of figure 44 – middle part) and their values can be customised. For instance, the author could change the default colour blue for the lines of the box into the red colour. Note that the effect of the customisation can always be previewed in the previewer (see right side of figure 44). Once satisfied with the customisation, it can be saved (by pressing the “Save Adaptation State” button). A VR object should be associated with all VRASs needed in the course and all VR objects that require adaptations need to be associated with some VRASs. Once this is done, the VR resource should be stored with data model type “VRLO”.

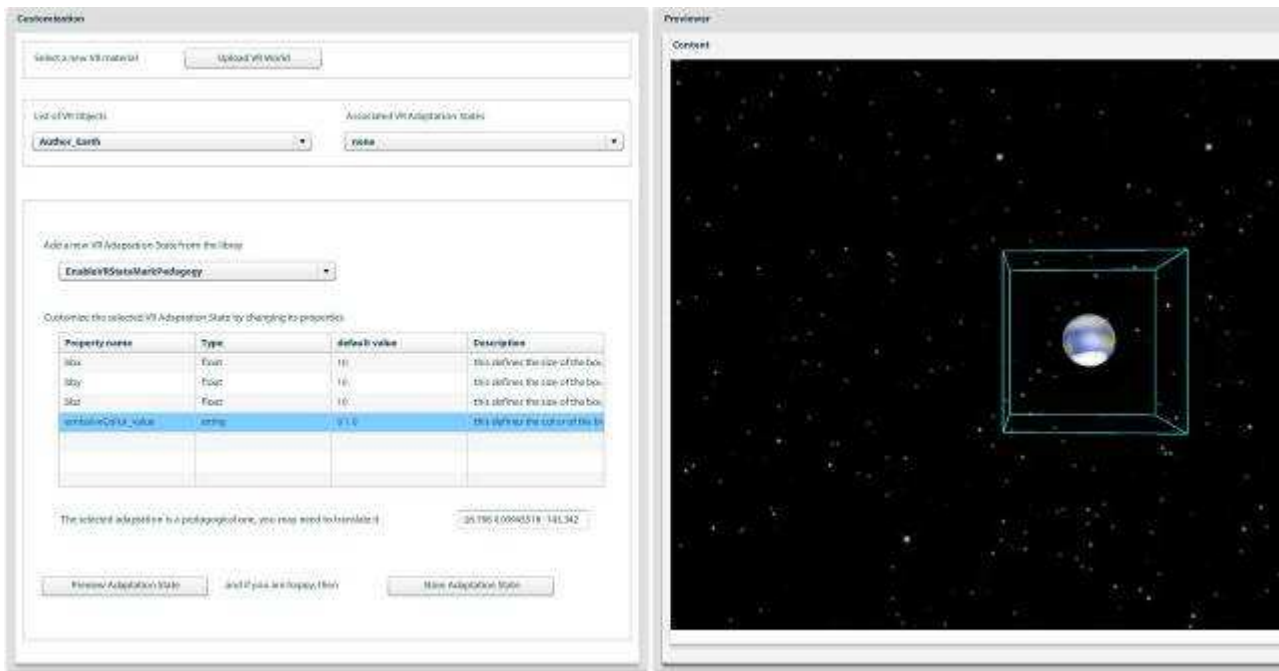


Figure 44. Associate the VR Adaptation State *VRAS\_MarkPedagogy* to Earth

### 7.3 Step3: Define VRCRTs (optional)

To specify VR adaptations (step 4), the author will need to use some VRCRTs. For instance, for the given scenario, a VRCRT is needed to activate some behaviour (i.e. rotate) for a VR object (i.e. Earth) after the learner has interacted with this VR object and the VR object has been marked. The creation of such a VRCRT, called *VRCRTActivateBehaviourAftermarking* will be illustrated.

To create a VRCRT, the author should select *File->New*. This will result in the “New File” window in which the author can enter the name of the VRCRT to be created, a description and the type VRCRT should be selected. By pressing the *Create* button, a new window will be displayed (see figure 45).

The author will see the window of the Main tab (see figure 45). The name and the description given in the previous window are already filled in. In this case, the author should select *directional* for the VRCRT type in order to be able to specify an if-then rule (similar to a pre-requisite relationship). Next a colour for the VRCRT can be selected. The colour illustrates the level of expertise that learner should have. In this case, yellow is chosen to illustrate that it is for a beginner as shown in Figure 45. The author can now compose the adaptation rule associated with the VRCRT by either entering the rule in plain GAL code (see right side of figure 46) or using an editor (on the left side of figure 46).

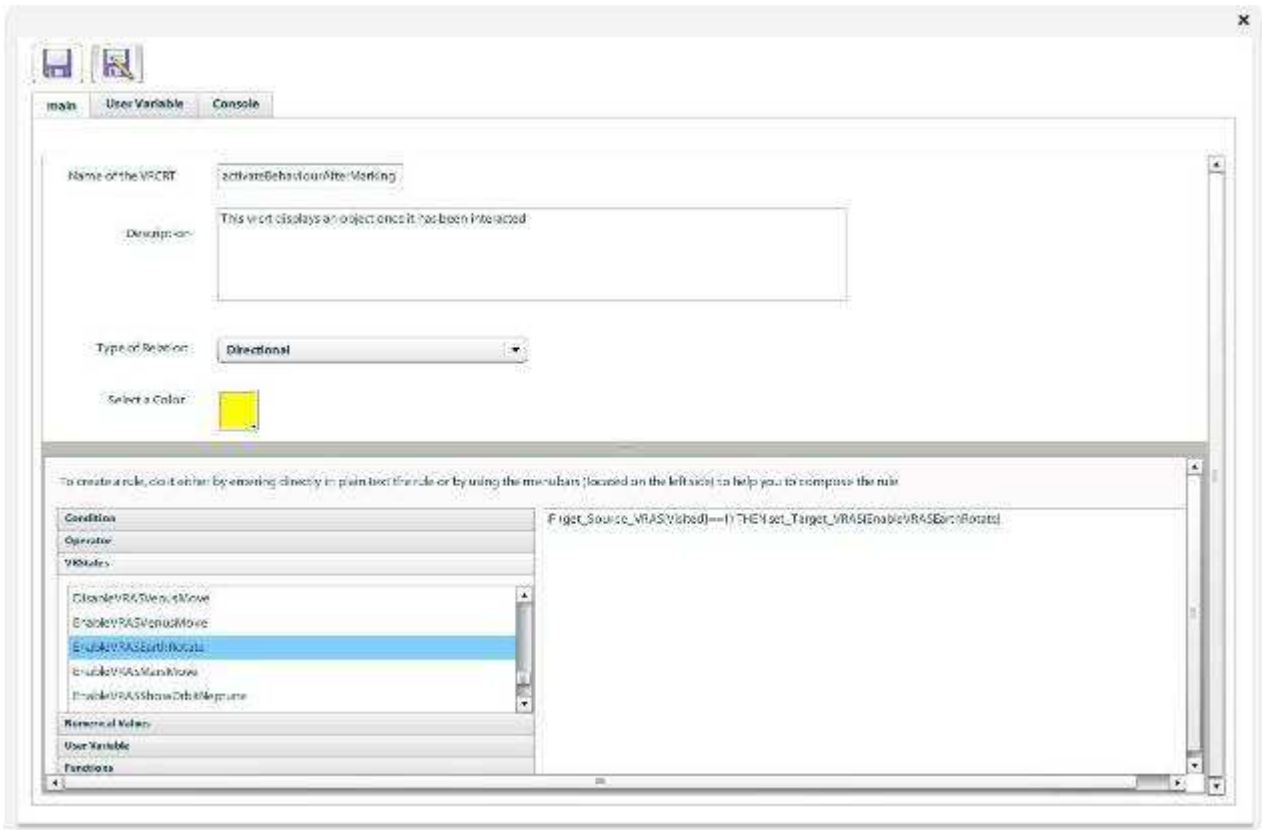


Figure 45. Creating a VRCRT: General Tab window

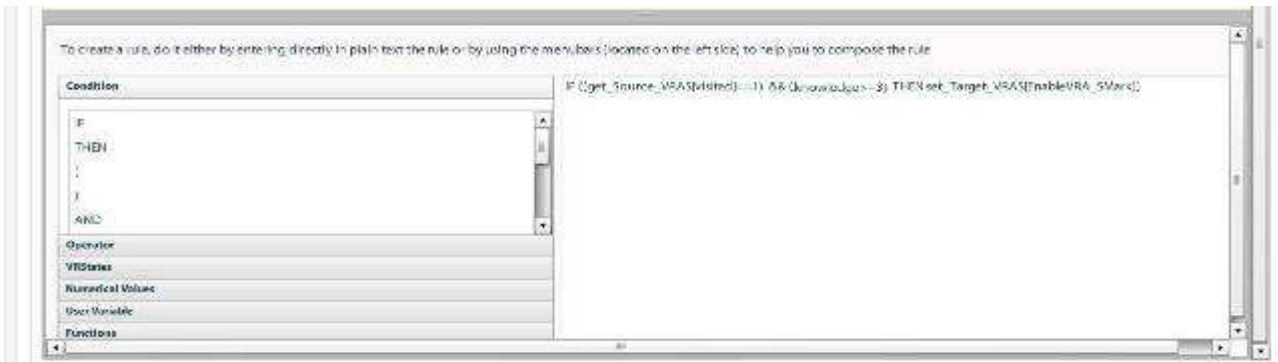


Figure 46. VRCRTActivateBehaviourAfterMarking: GAL code

The author also needs to specify the user variable(s) that are used in the GAL code. These user variables can be specified using the User Variable tab (see figure 47). Note that in this case the user model variable comes from a LMS external to GRAPPLE which is why the location of the user model variable should be given (see Location section in figure 47).

This concludes the specification of VRCRTActivateBehaviourAfterMarking.

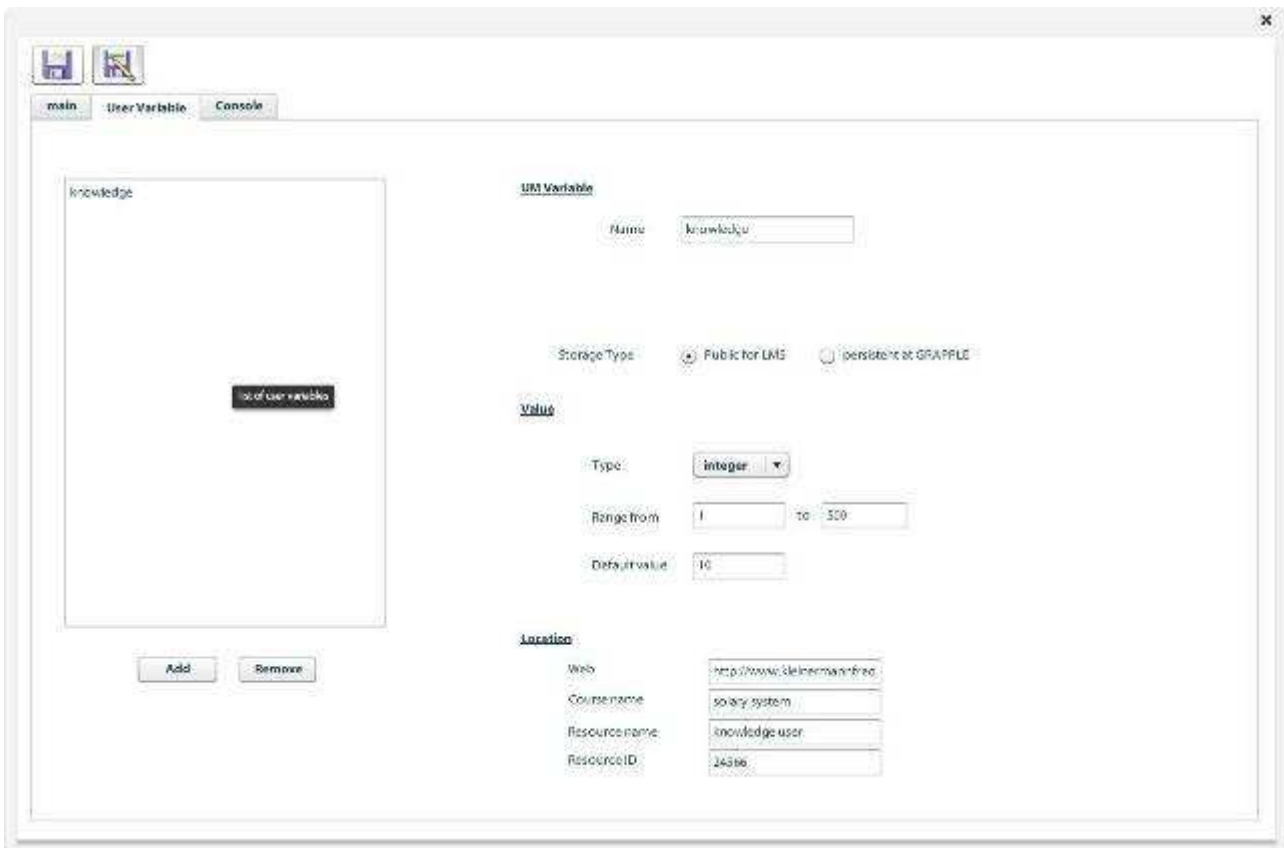


Figure 47. VRCRTActivateBehaviourAfterMarking: User Variable tab

#### 7.4 Step 4: Create a VRCAM to specify the adaptation within the VR Resource

The author is now ready to create a VRCAM to specify the required adaptation within the VR resource (being the virtual world containing Earth, Mars and Sun). Select File->New and the “New File” window will display in which the author should enter the name and description of the VRCAM to be created. The VRCAM type should be selected. Once the the *Create* button is pressed a new window will be displayed (see figure 48).

Note that script-based approach is used to specify a VRCAM. This is comparable to a movie script as will be illustrated with the following example. The adaptations steps that the author would like to have for this course:

- Step 1:  
At the start of the course, the VR object *Earth* is marked with a blue edge cube and annotated by the label “Earth”.
- Step 2:  
After the second interaction of the learner with *Earth*, the VR object *Earth* should start to rotate around its axis.
- Step 3:  
Again after the second interaction of the learner with *Earth*, the VR object *Earth* should move around its orbit.
- Step 4: Consisting of:
  - (4a) When the learner interacts once more with VR object *Earth*, the VR object *Earth* will be unmarked.
  - (4b) The learner is moved to the VR object Mars.
  - (4c) The VR object Mars is marked.
- Step 5:  
The course ends by highlighting the VR object *Mars*.

The author should specify this as will be shown next. Assume the existence of a Domain Model for the solar system.

#### Stage 1: Mapping between VR objects and Domain Concepts

The author will first need to specify the link between the VR objects in the considered VR resource and the concepts in the domain model of this course. Start with selecting the VR resource and obtain a list showing all identified VR objects that can be used for adaptation, in this example *Author\_Earth*, *Author\_Mars* and *Author\_Sun*. Next the domain model for this course (in this case the solar system) should be uploaded and the mappings between VR objects and Domain Concepts need to be defined by clicking on *Mapping*. The domain concepts for the different VR objects can now be added. For instance, figure 48 shows that the domain concept for the VR object *Author\_Earth* is *Earth*.



Figure 48. Mapping VR objects to Domain Concepts

## Stage 2: Compose the VRCAM

The VRCAM can now be defined. The VRCAM is a script composed of a number of *course entities*. The script of the VRCAM starts with a *Start entity*, has a number of *Adaptation Blocks* and ends with an *End entity*. To specify this VRCAM, the author can select the entity types from the menu and drag them on the canvas. In figure 49 a Start and an Adaptation Block have been dropped on the canvas. Note that Start and End can be considered as a special kind of Adaptation Block. For each Adaptation Block (which corresponds with an adaptation step), the author should specify the required adaptations. This is done by selecting the VR objects that need to be adapted and the VRASs to be used. For instance, for the Start entity, the VR object *Author\_Earth* should be marked. To do this, the author needs to add the VR object *Author\_Earth* to this Adaptive Block and to select the VRAS for marking a VR-object (see figure 50).

Next, the author needs to specify **when** the actions specified in the Adaptive Block need to be executed. To specify this, the author uses a VRCRT. The needed VRCRT is selected from the list of available VRCRTs (see figure 51). Selecting a VRCRT will display its visual representation on the canvas (see figure 52). The author then needs to connect it with the source course entity (i.e. a Start entity or Adaptive Block) and the target course entity (i.e. an End entity or Adaptive Block).

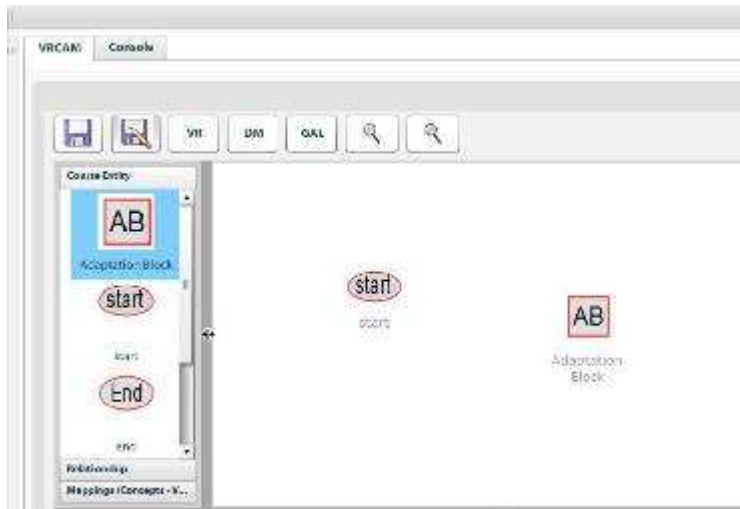


Figure 49. A start entity and an Adaptive Block have been dragged on the canvas

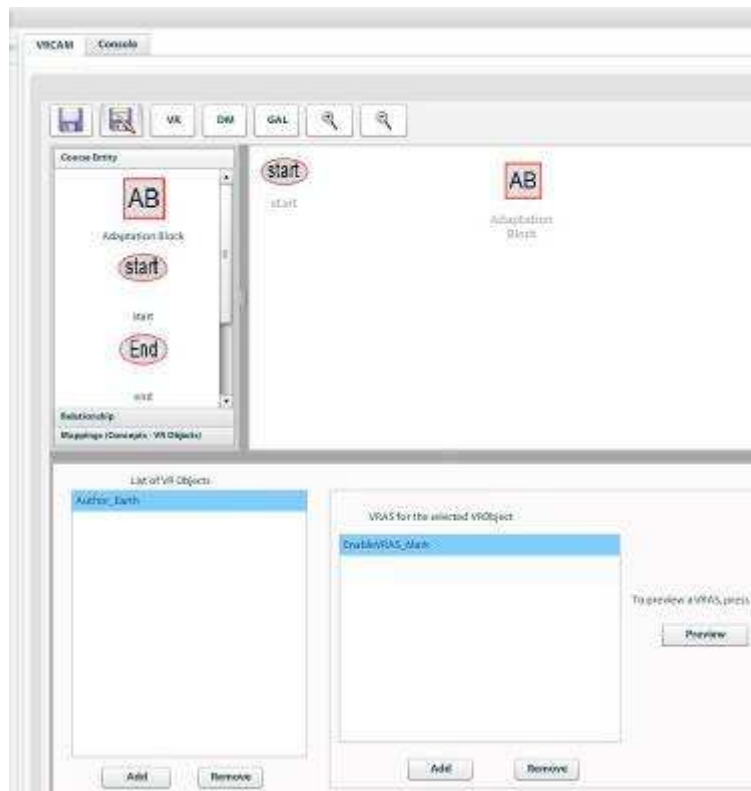


Figure 50. Adding VR object Author\_Earth with a VRAS for marking it

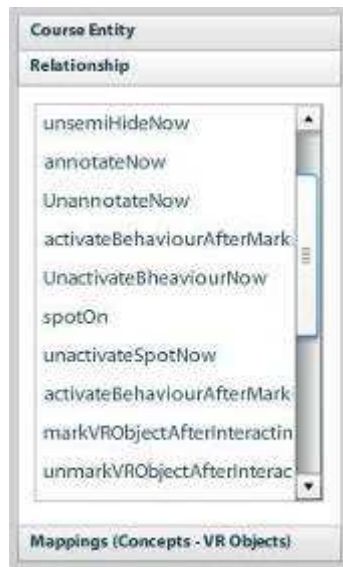


Figure 51. List of available VRCRTs

Figure 52 shows a VRCRT *markNow* being selected. By double clicking on the VRCRT the author can see the rule of the VRCRT and adapt the rule if necessary as shown in Figure 53. Note that once a VRCRT is dragged onto the canvas it becomes an independent instance of that VRCRT and it can be modified as such. In other words, the original VRCRT will not be changed; it only acts as a template.

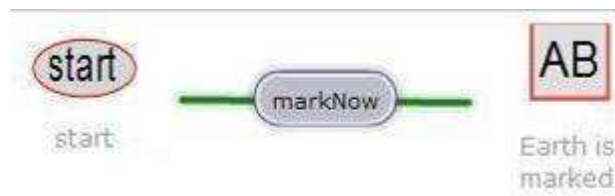


Figure 52. *markNow* VRCRT has been selected

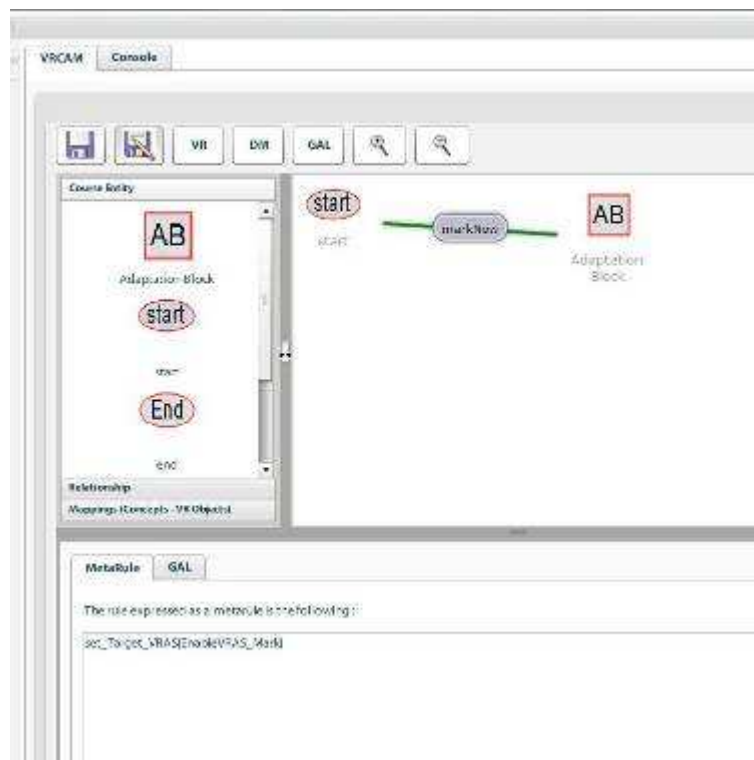


Figure 53. DoubleClicking on a VRCRT shows the rule and allows adapting it

Figure 54 shows another Adaptation Block to specify the adaptation where Earth will have an annotation. The VRCRT *annotateNow* is used to tell when to annotate.

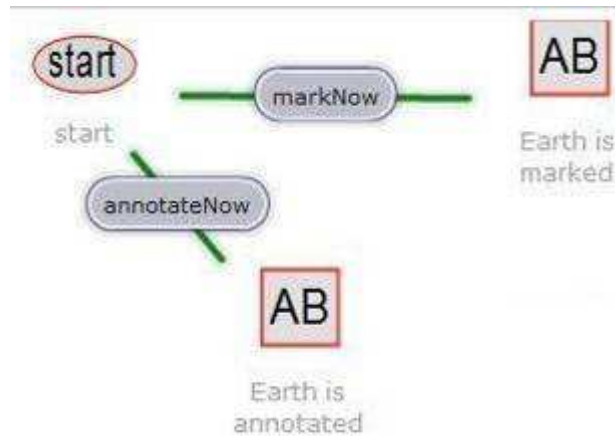


Figure 54. Adding the VRCRT *annotateNow*

To achieve the second step of the course script (i.e. once the user interacts a second time with the VR object Earth, then this VR object should start to rotate around its axis), the author drags a second Adaptation Block on the canvas. For this Adaptation Block specify that the VR object *Author\_Earth* should be in a VRAS that makes it rotating around its axis. As VRCRT select one that triggers a behaviour once the user has interacted with the VR object (*activateBehaviourAfterMarking* - see figure 55).



Figure 55. VRCRT for activating a behaviour

The author will then specify step 3 of the script, i.e. when the user starts to interact a second time, the VR object *Earth* will move around its orbit. In order to achieve this, the author drags a new Adaptation Block on the canvas; adds the VR object *Author\_Earth* to this Adaptation Block and selects the appropriate VRAS. To specify when to move to this state, use the same VRCRT as for step 2 as shown in Figure 56.

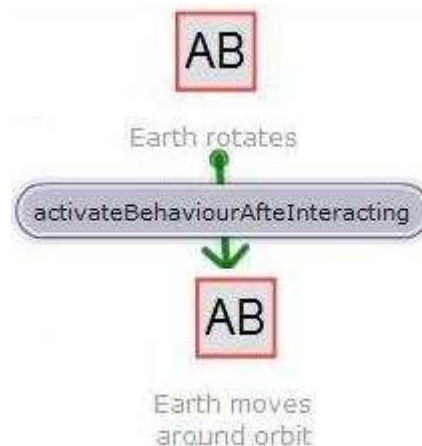


Figure 56. VRCAM for step 3

Step 4., being composed of (4a) (when the learner interacts once more with VR object *Earth*, the VR object *Earth* will be unmarked), (4b) (the learner is moved to the VR object Mars) and (4c) (the VR objects Mars is marked) is realised in the same way. This has been specified by the part of the VRCAM shown in figure 57.

Finally, step 5 can be specified, i.e. the course should end by putting a red spotlight on the VR object *Mars*. The author should drag the special course entity *End* on the canvas and use the unidirectional VRCRT *spotOn* to trigger the VRAS that highlights a VR object; the VR Object *Mars* is added to the course entity *End*. Figure 58 shows the complete VRCAM.

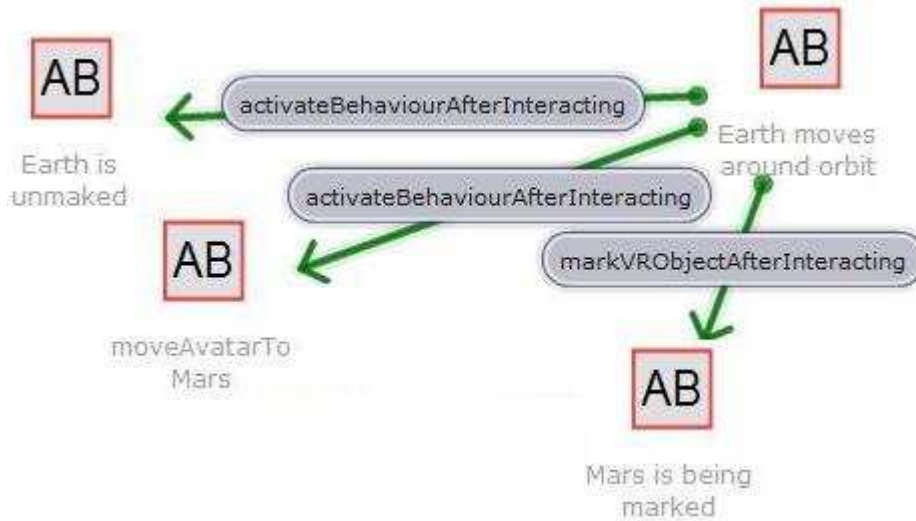


Figure 57. Step 4 realised

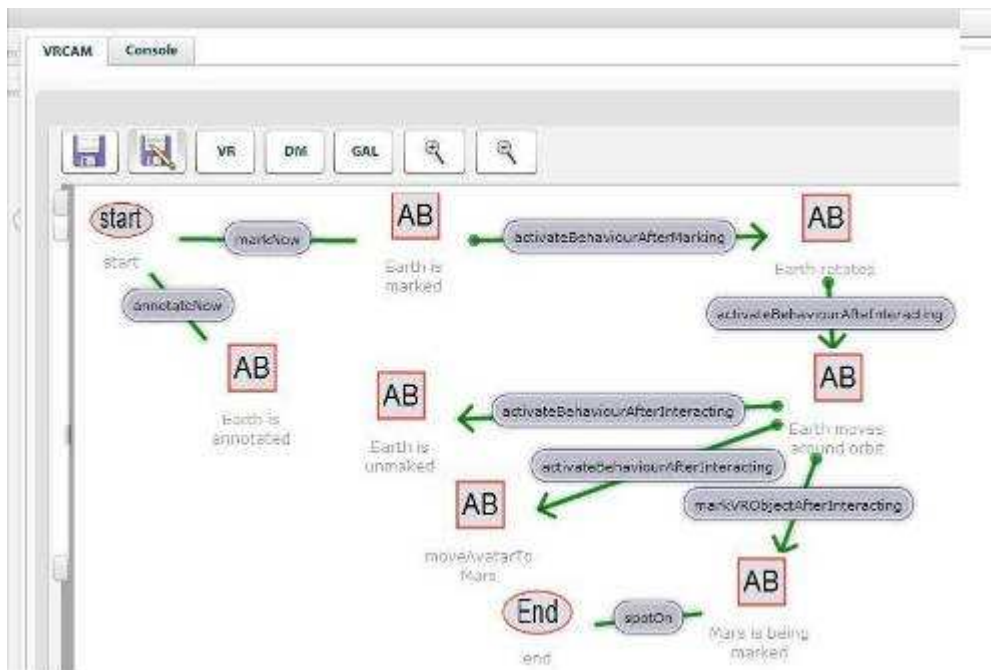


Figure 58. Step 5: Ending the course

### 7.5 Step 5: Adding the VRCAM to the CAM

If the author wants to integrate the adaptive VR resource into a regular course, the created VRCAM should be incorporated into the regular CAM (modelling the regular course). It is assumed the author has already created this CAM (using GAT). Figure 59 shows this CAM. Figure 60 is a zoom of that CAM.

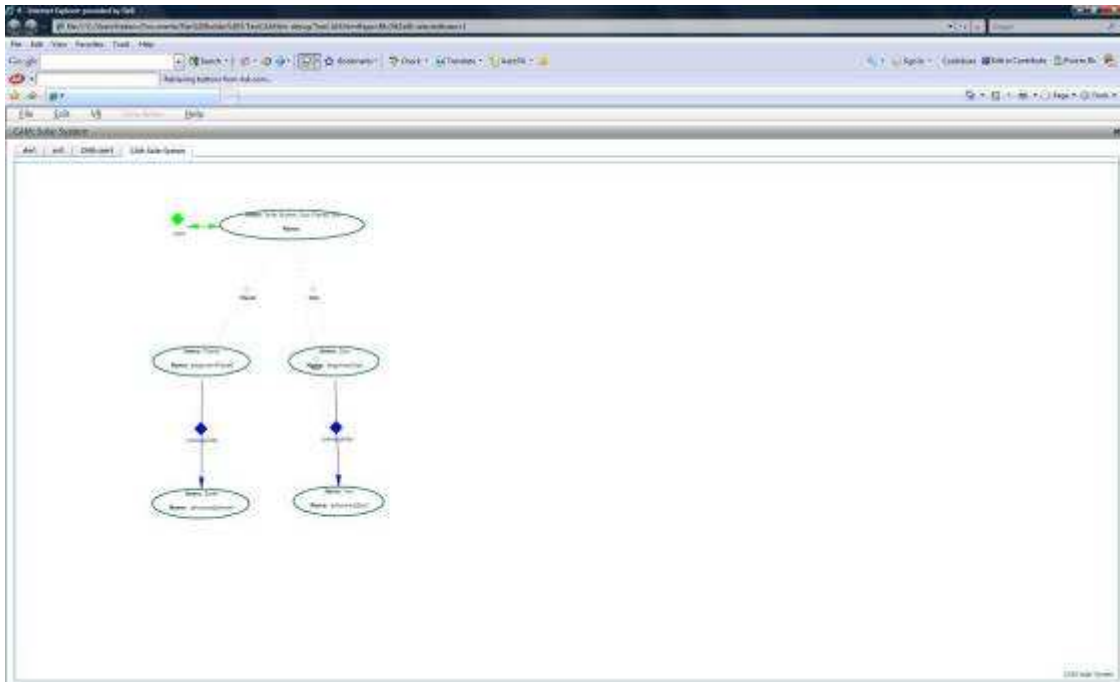


Figure 59. Overall view of the CAM in the GAT tool

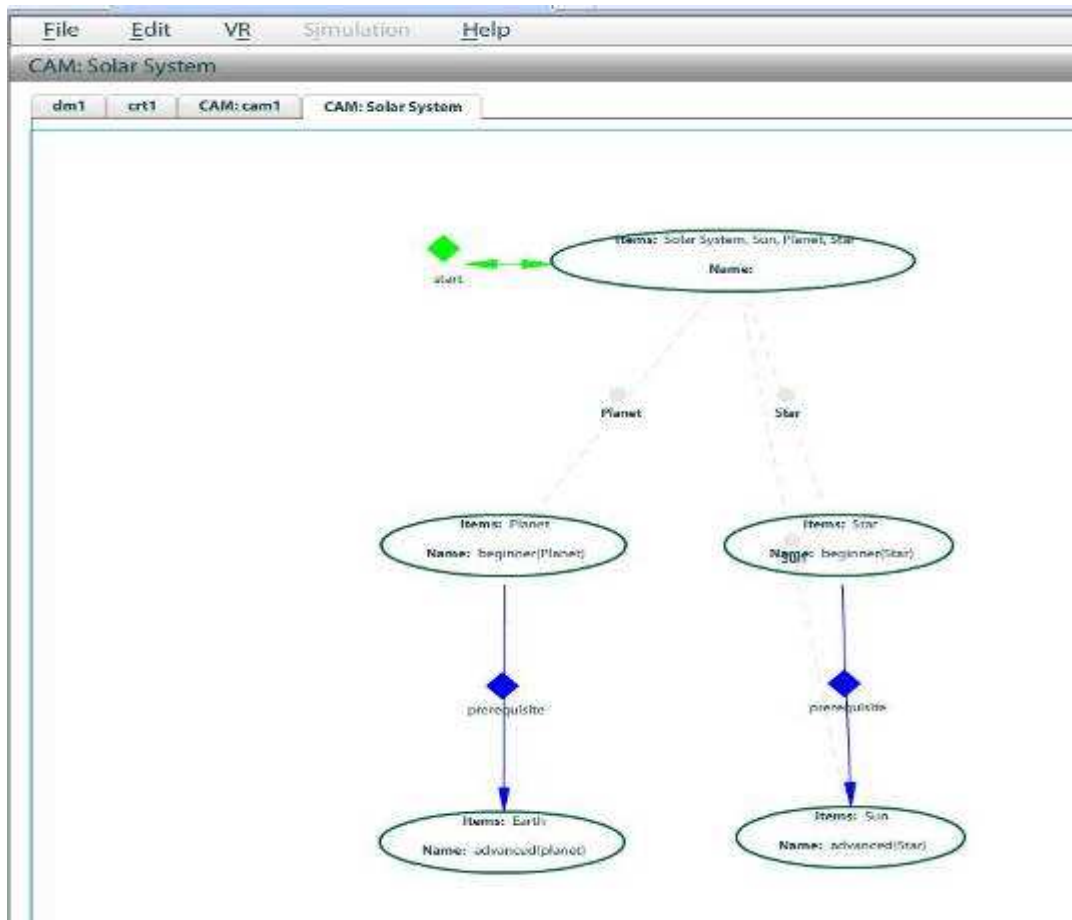


Figure 60. CAM in more details

The CAM shows that the course should start by introducing the Solar System. It shows that the concept *Planet* must be studied before the concept *Earth* (expressed by the CRT *prerequisite*). The same goes for the concept *Star* and the concept *Sun*.

According to this scenario, the VRCAM should become active when the Sun concept has been studied. That is why it should be included in the CAM socket called *advanced[Star]* for Sun.

To add the VRCAM to the CAM, the author should use the VR Authoring tool and select *File->Insert VRCAM*. This results in the window shown in figure 61. In that window, the author selects the socket from the CAM model in which the VRCAM should appear, in this scenario the CAM socket is called *advanced[Star]* for Sun. The VRCAM to be used should be specified by selecting the *Imported VRCAM* button. The VRCAM will be stored in the existing CAM data model when the author presses the *Save* button.

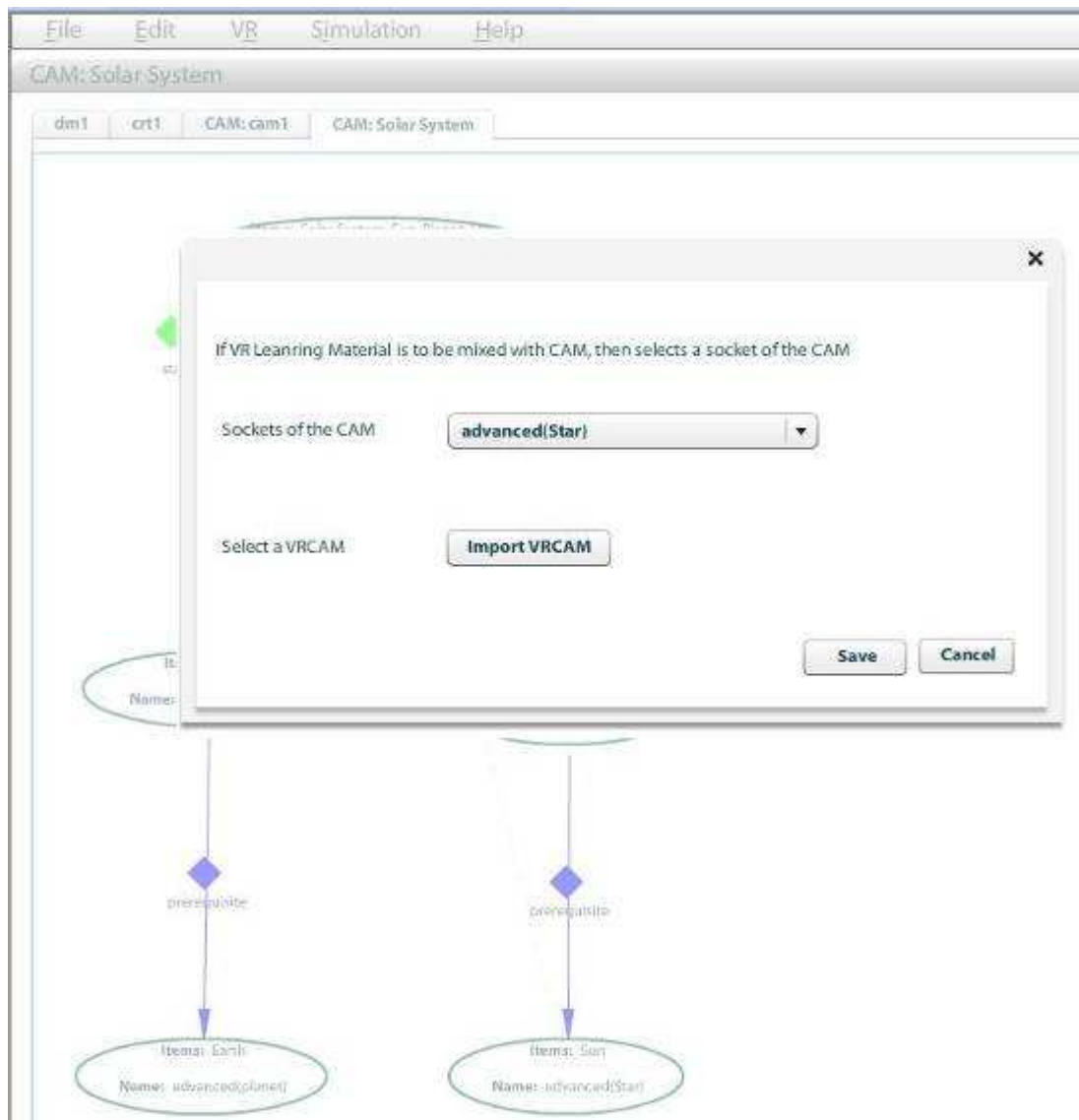


Figure 61. VRCAM being inserted into a socket of the CAM related to the Solar system

## 7.6 Step 6: Adding a VR resource without adaptation to the CAM

In the given scenario, the author also wants to insert a VR resource (without adaptation) in order to show an example of a planet (earth in 3D) once the author visits the concept planet. This can be done by inserting the VR resource containing a 3D representation of earth into the socket with *beginner[planet]* using the CAM component of the GAT. To check which VR resource to use, invoke the Previewer (see figure 62) from the VR authoring tool by selecting *Tool->Preview VR Resources*. In order to visualise VR resources, copy the link of the desired VR resource, and paste it into the appropriate CAM socket using the CAM component of the GAT tool.

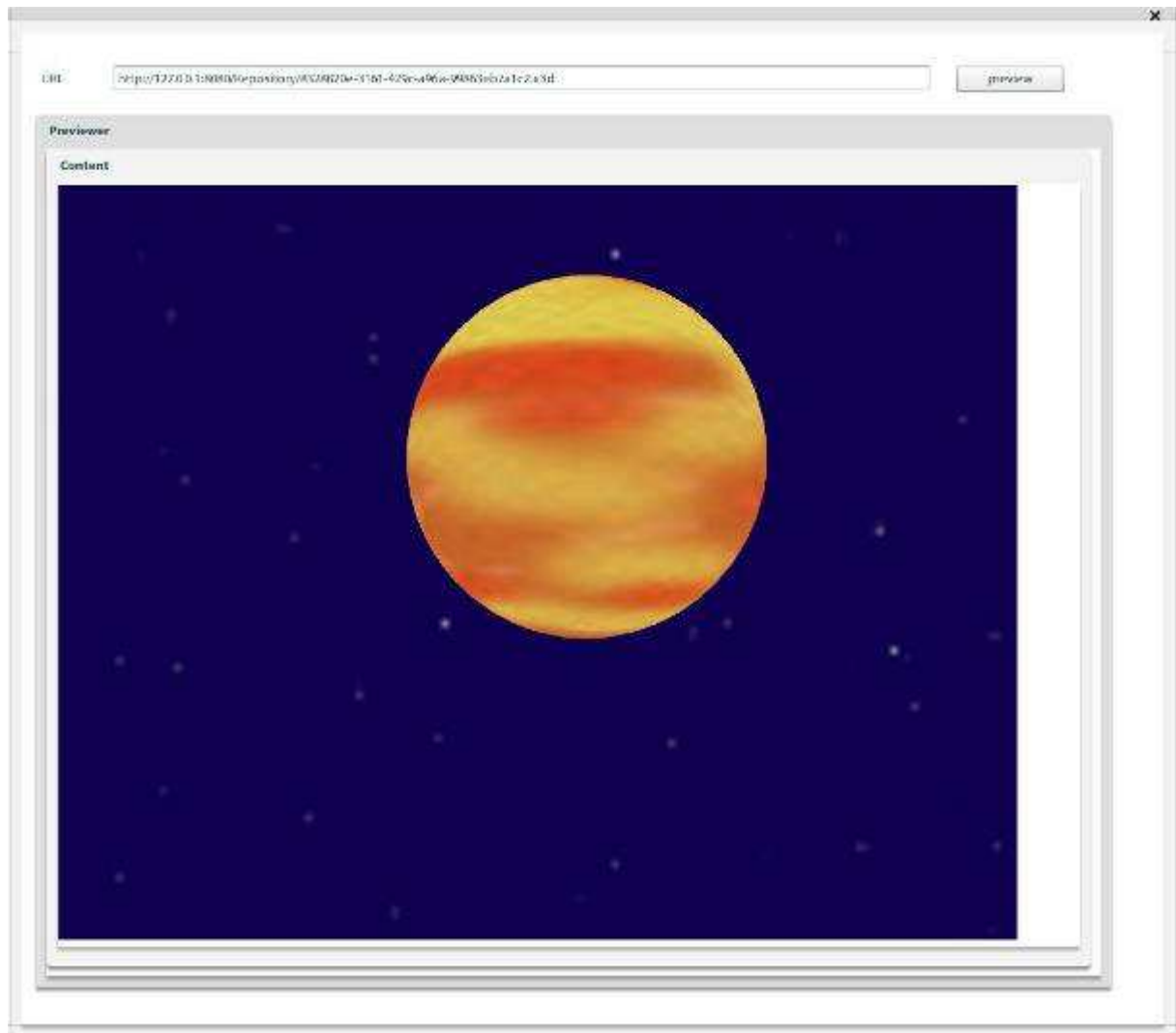


Figure 62. Inspecting a VR Learning Material using the previewer

## 8 General Discussion

Currently the VR Authoring tool requires installation of an X3D player as plug-in for the Internet browser. The Vivaty player<sup>7</sup> has been used in this example. Over the past few months there has been some progress within the X3D community to have the next generation of Internet browsers like Firefox<sup>28</sup> to integrate HTML5 and X3D. This evolution will avoid the need to download and install plug-ins. The architecture presented in this document will still be valid.

The approach applied in this deliverable allows an author to import existing VR material. To achieve this, some pre-processing steps are necessary. This may look complicated at first, but once those steps have been taken a few times, it will soon become easy to use. Additionally, once the VR material has been prepared, it can be used for different courses.

Defining new VR adaptation states (VRASs) is also allowed. It is recognised that it may not be easy for authors to do this and that is why it may be left to someone who has some VR background. But again, once a new adaptation state has been added, it can be used for all the different courses being created.

Chapters 6 and 7 explained how an author can create new VRCRTs for their use in a VRCAM. Again it is recognised that creating a VRCRT is not easy. However, once a library of VRCRTs has been created, they can be used in many situations.

A special effort has been made to make the specification of a VRCAM more VR author oriented. As VR authors tend to think in terms of scenario's, a VRCAM is specified as a script or scenario. The different steps

<sup>7</sup> <http://www.vivaty.com/downloads/player/>

<sup>8</sup> [http://en.wikipedia.org/wiki/Mozilla\\_Firefox](http://en.wikipedia.org/wiki/Mozilla_Firefox)

are specified using course entities representing specific adaptations of VR objects and using VRCRTs to specify when the adaptations need to take place.

## 9 Conclusion

Deliverable 3.4a (Kleiner mann and De Troyer, 2009) provided related work and the initial design of the VR authoring component. The deliverable was revised a first time, resulting in deliverable 3.4.b. This deliverable,(3.4c)contains:

- Revised material from the previous deliverable (D3.4.b) and in particular the revised process of creating a VRCAM;
- A more in depth explanation of the current version of the VR authoring tool;
- An explanation of how VR authoring tool is integrated with the general Grapple Authoring Tool (GAT);
- A user guide.

## References

- Alexiou, A.; Bouras, C.; Giannaka, E.; Kapoulas, V.; Nani, M.; Tsiatsos T. (2004): Using VR technology to support e-learning: the 3D virtual radiopharmacy laboratory, In Distributed Computing Systems workshops held in conjunction of 24th International Conference on Volume, Issue , 23-24, pp. 268-273.
- Albert, D., Nussbaumer, A., Steiner, C. M., Hendrix, M., Cristea, A. (2009): Design and Development of an Authoring Tool for Pedagogical Relationship Types between Concepts. Poster at the 17th International Conference on Computers in Education (ICCE 2009), 30 November - 4 December 2009, Hong Kong
- Barab, S.A., Thomas, M., Dodge, T., Canteaux, R., Tuzan, H. (2005): Making learning Fun: Quest Atlantis, a game without guns, Educational Research and Development, vol 53, no 1, pp. 86-107.
- Blanco, A., Torrente, J. Moreno-Ger, P., Fernández-Manjón, B. (2009): Bridging the Gap: Adaptive Games and Student-Centered VLEs, In Proceedings of the International Conference on Web-based Learning (ICWL 2009), LNCS, Springer, vol. 5686, pp130–139, 2009
- Brusilovsky, P., Hughes, S., Lewis, M. (2002): “Adaptive Navigation Support In 3-D E-Commerce Activities”, Workshop on Recommendation and Personalization in eCommerce, Proceedings of 2nd International Conference on Adaptive Hypermedia and Adaptive Web Based Systems, Malaga, Spain, pp. 132-139.
- Brutzman, D., Daly L. (2008): X3D: Extensible 3D graphics for Web Authors, The Morgan Kaufmann Series in Interactive 3D technology.
- Celentano, A., Pittarello, F. (2004): Observing and Adapting User Behaviour in Navigational 3D interface, In: Proc of 7<sup>th</sup> International Conference on Advanced Visual Interfaces 2004, (AVI 2004), ACM Press, pp. 275-282.
- Chittaro, L., Ranon, R. (2000): Adding Adaptive Features to Virtual Reality Interfaces for E-Commerce, Proceedings of AH-2000: International Conference on Adaptive Hypermedia and Adaptive Web-based Systems, Lecture Notes in Computer Science 1892, Springer-Verlag, Berlin, August. pp.86-97.
- Chittaro, L., Ranon, R. (2002): Dynamic Generation of Personalized VRML Content: a General Approach and its Application to 3D E-Commerce, Proceedings of 7th International Conference on 3D Web Technology, Web3D'2002, ACM Press, pp. 145-154.
- Chittaro, L., Ranon R. (2004): Using the X3D Language for Adaptive Manipulation of 3D Web Content, Proceedings of the 3rd International Conference on Adaptive Hypermedia and Adaptive Web-based Systems (AH 2004), Eindhoven, Netherlands. Springer- Verlag, Berlin Heidelberg New York, pp. 287-290.
- Chittaro, L., Ranon, R. (2007a): Adaptive Hypermedia Techniques for 3D Educational Virtual Environments, IEEE Intelligent Systems (Special Issue on Intelligent Educational Systems), pp. 31-37.

- Chittaro, L., Ranon, R. (2007b): Adaptive 3D Web Sites, In Brusilovsky, P., Kobsa, A., Nejd, W. (eds.), The Adaptive Web - Methods and Strategies of Web Personalization, Lecture Notes in Computer Science Vol. 4321, Springer-Verlag, Berlin, pp. 433-464.
- Chittaro, L., Ranon R. (2008): An Adaptive 3D Virtual Environment for Learning the X3D Language, Proceedings of the 2008 International Conference on Intelligent User Interfaces (IUI 2008), ACM Press, New York, pp. 419-420.
- Chopra, A. : Google Sketchup 7 for Dummies, John Wiley & Sons, 2009
- Dachselt, R., Hanz, M., Meissner, K. (2002): Contigra: an XML-based architecture for component-oriented 3D application, In proc of the 7<sup>th</sup> international conference on 3D web technology, ACM Press, pp.155-163.
- De Byl, P. (2009): An architecture for embedding Authentic learning Experiences into Serious Game Application, In handbook of Research Effective Electronic Gaming in Education.
- Dicerto, M., Oneto, L. (2009): Initial Implementation of the Domain Model Tool, Deliverable 3.1b, GRAPPLE
- Flanagan, D. : Javascript The Definitive Guide, 3<sup>rd</sup> edition, O'Reilly, 1998
- Gutierrez, M.A., Vexo, F., Thalmann, D. (2008): Stepping into Virtual Reality: A Practical Approach , Springer.
- Hendrix, M. , Nussbaumer, A., Dicerto, M., Oneto, L., Cristea, A.: GAT: The FP7 GRAPPLE Authoring Tool Set. Demonstration at the Fourth European Conference on Technology Enhanced Learning (EC-TEL 2009), 29 September - 2 October 2009, Nice, France, 2009
- Hendrix, M., Harrigan, M.: Initial Implementation of Concept Adaptation Model tool, Deliverable 3.3b, GRAPPLE, 2009
- Kleineremann, F., De Troyer, O. (2008a): Design of an authoring platform for adaptive VR learning Material, Deliverable 3.4a, GRAPPLE
- Kleineremann, F., De Troyer, O. (2008b): Design of the infrastructure for the adaptive delivery of VR learning material, Deliverable 4.2a, GRAPPLE
- Livingstone, D., Kemp, J. (2008): Integrating web-based and 3D learning environments : secondlife meets Moddles, UPGRADE, vol 9, no 3, pp. 8-14.
- Moreno-Ger, P., Blesisu, C., Currier, P., Sierra-Rodriguez, J.L, Baltzar, F. (2008a): Online Learning and Clinical Procedures: Rapid Development and Effective Deployment of Game-Like Interactive Simulations, Transactions on Edutainment I, Lecture Notes in Computer Science 5080, pp. 288–304.
- Moreno-Ger, P., Sierra-Rodriguez, J.L., Fernandez-Manjon, B. (2008b): Games-based learning in e-learning Environments, UPGRADE, vol. 12, no 3, pp. 15-20.
- Moreno-Ger, P., Martínez-Ortiz, I., Sierra, J.L., Fernández-Manjón, B. (2008c): A content-Centric Development Process Model, IEEE Computer, vol 41, no 3, pp. 24-30.
- Moreno-Ger, P., Burgos, D., Martinez-Ortiz, I, Sierra, J.L., Fernández-Manjón, B. (2008d): Educational Game design for online Education, Computer in Human Behavior, vol 24, pp. 414-431.
- Murdock, K.L. : 3Ds Max 2010 Bible, John Wiley & Son, 2010
- Noble, J. and Anderson, T. : Flex 3 cookbook, O'Reilly, 2008
- Lott, J., Schall, D., Peters, K. : Actionsript 3.0 Cookbook, O'Reilly, 2006
- Steiner, C.M., Nussbaumer, A. : Initial Implementation of the Concept Relationship Type Tool, Deliverable 3.2b, GRAPPLE, 2009
- Santos, D., C. T., Osorio, F. S. (2004): AdapTIVE: An Intelligent Virtual Environment and Its Application in E-Commerce, Proceedings of 28<sup>th</sup> Annual International Computer Software and Applications Conference (COMPSAC'04), pp. 468-473.
- Torrente, J, Moreno-Ger, P., Martínez-Ortiz, I., Fernández-Manjón, B. (2009): Integration and Deployment of Educational Games in e-Learning Environments: The Learning Object Model Meets Educational Gaming. Educational Technology & Society, 12 (4), pp359–371
- Westra, J., van Hasselt, H., Dignum, F., Dignum, V. (2009): Adaptive Serious Games Using Agent Organizations, The First International Workshop on Agents for Games and Simulations (AGS), LNCS, Springer, vol. 5920, pp206-220
- , AliveX3d: <http://www.alivex3d.org>, accessed on the 2<sup>nd</sup> of December, 2008

- , Adobe Flex: <http://www.adobe.com/fr/products/flex/>, accessed on the 2<sup>nd</sup> of December, 2008
- , Internet Explorer : [http://en.wikipedia.org/wiki/Internet\\_Explorer](http://en.wikipedia.org/wiki/Internet_Explorer), accessed 26 October 2009
- , Firefox : [http://en.wikipedia.org/wiki/Mozilla\\_Firefox](http://en.wikipedia.org/wiki/Mozilla_Firefox), accessed 26 October 2009
- , KM Quest: <http://www.kmquest.net>, accessed on the 2nd of November, 2008
- , Moodle: <http://moodle.org/>, accessed on the 2nd of November, 2008
- , X3D: <http://www.x3d.org>, accessed on the 2nd of November, 2008
- , SecondLife: [http:// www.SecondLife.org](http://www.SecondLife.org), accessed on the 2nd of November, 2008

## 10 Appendix

The appendix shows provides the XML specification for each Data Model. Note that these DATA models are still under development until the final deliverable 3.4c.

### 10.1 XML Schema for Virtual Reality Conceptual Adaptation Model (VRCAM)

```
<?xml version="1.0" encoding="UTF-8"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema" targetNamespace="http://www.example.org/CAM-external-oct2009" xmlns:tns="http://www.example.org/CAM-external-oct2009" elementFormDefault="qualified" xmlns:Q1="http://www.imsglobal.org/xsd/imsvdex_v1p0" xmlns:Q2="http://grapple-project.org/GAT/" xmlns:Q3="http://www.grapple-project.org">

  <import schemaLocation="dm-vdex-extensions.xsd" namespace="http://www.grapple-project.org"></import>
  <import schemaLocation="VRCRT.xsd" namespace="http://grapple-project.org/GAT/"></import>
  <import schemaLocation="vdex.xsd" namespace="http://www.imsglobal.org/xsd/imsvdex_v1p0"></import>
  <element name="model" type="tns:modelType"></element>

  <complexType name="modelType">
    <sequence maxOccurs="1" minOccurs="1">
      <element name="header" type="tns:headerType" maxOccurs="1" minOccurs="1"></element>
      <element name="body" type="tns:vrCAMBodyType" maxOccurs="1" minOccurs="1"></element>
    </sequence>
  </complexType>

  <complexType name="headerType">
    <sequence>
      <element name="modeluid" type="tns:UUIDtype" maxOccurs="1" minOccurs="1"></element>
      <element name="modeltype" type="string" maxOccurs="1" minOccurs="1"></element>
      <element name="authoruid" type="tns:UUIDtype" maxOccurs="1" minOccurs="1"></element>
      <element name="authorisation" type="tns:authorisationType" maxOccurs="1" minOccurs="1"></element>
      <element name="creationtime" type="dateTime" maxOccurs="1" minOccurs="1"></element>
      <element name="updatetime" type="dateTime" maxOccurs="1" minOccurs="1"></element>
      <element name="title" type="string" maxOccurs="1" minOccurs="1"></element>
      <element name="description" type="string" maxOccurs="1" minOccurs="1"></element>
    </sequence>
  </complexType>
```

```

<complexType name="vrCAMBodyType">
  <sequence>
    <element name="vrCAM" type="tns:vrCAMType" maxOccurs="1"
minOccurs="1"></element>
  </sequence>
</complexType>

<simpleType name="authorisationType">
  <restriction base="string">
    <enumeration value="read"></enumeration>
    <enumeration value="write"></enumeration>
    <enumeration value="readwrite"></enumeration>
  </restriction>
</simpleType>

<simpleType name="UUIDtype">
  <restriction base="string">
    <pattern
      value="[a-z0-9]{8}\-[a-z0-9]{4}\-[a-z0-9]{4}\-[a-z0-9]{4}\-[a-z0-9]{12}">
    </pattern>
  </restriction>
</simpleType>

<complexType name="vrCAMType">
  <sequence>
    <element name="vrCAMInternal" type="tns:vrCAMInternalType"></element>
  </sequence>
</complexType>

<complexType name="vrCAMInternalType">
  <sequence>
    <element name="domainModel" type="tns:dmModelType" maxOccurs="1"
      minOccurs="1">
    </element>
    <element name="vrcrtModel" type="tns:vrcrtModelType" maxOccurs="1"
      minOccurs="1">
    </element>
    <element name="vrcrt" type="string" maxOccurs="unbounded"
minOccurs="0"></element>
  </sequence>
</complexType>

<complexType name="positionType">
  <sequence>
    <element name="x" type="int"></element>
    <element name="y" type="int"></element>
  </sequence>
</complexType>

<complexType name="vrCAMSocketType">
  <sequence>
    <element name="caption" type="string" maxOccurs="1"
      minOccurs="0">
    </element>
    <element name="uuid" type="tns:UUIDtype" maxOccurs="1"
      minOccurs="1">
    </element>
    <element name="socketId" type="tns:UUIDtype" maxOccurs="1"
      minOccurs="1">
  </sequence>

```

```

</element>
<element name="position" type="tns:positionType"
    maxOccurs="1" minOccurs="1">
</element>
<element name="shape" type="string" maxOccurs="1" minOccurs="0"></element>
<element name="colour" type="string" maxOccurs="1" minOccurs="0"></element>
<element name="entity" type="tns:entityType" maxOccurs="unbounded"
minOccurs="0"></element>
</sequence>
</complexType>

<complexType name="entityType">
<sequence>
<element name="dmID" type="tns:UUIDtype" maxOccurs="1"
    minOccurs="0">
</element>
<element name="label" type="string" maxOccurs="unbounded"
    minOccurs="0">
</element>
<element name="relationshipType" type="string" maxOccurs="1"
minOccurs="0"></element>
<element name="position" type="tns:positionType" maxOccurs="1"
minOccurs="0"></element>
<element name="size" type="int" maxOccurs="1" minOccurs="0"></element>
<element name="shape" type="string" maxOccurs="1" minOccurs="0"></element>
<element name="image" type="string" maxOccurs="1" minOccurs="0"></element>
<element name="colour" type="string" maxOccurs="1" minOccurs="0"></element>
</sequence>
</complexType>

<complexType name="dmModelType">
<sequence>
<element name="model" type="tns:dmModelType2" maxOccurs="unbounded"
minOccurs="0"></element>
</sequence>
</complexType>

<complexType name="dmModelType2">
<sequence>
<element name="header" type="tns:headerType"></element>
<element name="body" type="tns:dmType"></element>
</sequence>
</complexType>

<complexType name="dmType">
<sequence>
<element name="vdex" type="Q1:vdexType" maxOccurs="1" minOccurs="1"></element>
</sequence>
</complexType>

<complexType name="vrcrtModelType">
<sequence>
<element name="model" type="tns:crtModelType2" maxOccurs="unbounded"
minOccurs="0"></element>
</sequence>
</complexType>

<complexType name="vrcrtModelType2">
<sequence>
<element name="header" type="tns:headerType" maxOccurs="1"
    minOccurs="1">
</element>

```

```

        <element name="body" type="tns:crtType" maxOccurs="1"
minOccurs="1"><complexType></complexType></element>
    </sequence>
</complexType>

<complexType name="vrcrtType">
    <sequence>
        <element name="vrcrtdialect" type="Q2:vrcrtdialecttype"
            maxOccurs="1" minOccurs="1">
        </element>
        <element name="comment" type="Q3:contentType" maxOccurs="1"
            minOccurs="1">
        </element>
        <element name="visualrepresentation" type="string"></element>
        <element name="vrcrtsockets" type="string"></element>
        <element name="adaptationbehaviour" type="string"></element>
        <element name="constraints" type="string"></element>
        <element name="associateddmrelations" type="string"></element>
    </sequence>
</complexType>
</schema>

```

## 10.2 XML Schema for Virtual Reality Concept Relation Type (VRCRT)

```

<?xml version="1.0"?>
<xs:schema
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    targetNamespace="http://grapple-project.org/GAT/"
    xmlns="http://grapple-project.org/GAT/"
    elementFormDefault="qualified">

<!-- (A) The overall structure and the model header -->

<xs:element name="model">
    <xs:complexType>
        <xs:sequence>
            <xs:element name="header">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element name="modeltype" type="modeltypetype" minOccurs="1" maxOccurs="1"/>
                        <xs:element name="modeluuid" type="xs:string" minOccurs="1" maxOccurs="1"/>
                        <xs:element name="authoruuid" type="xs:string" minOccurs="1" maxOccurs="1"/>
                        <xs:element name="authorisation" type="xs:string" minOccurs="1" maxOccurs="1"/>
                        <xs:element name="title" type="xs:string" minOccurs="1" maxOccurs="1"/>
                        <xs:element name="description" type="xs:string" minOccurs="1" maxOccurs="1"/>
                        <xs:element name="creationtime" type="xs:dateTime" minOccurs="1" maxOccurs="1"/>
                        <xs:element name="updatetime" type="xs:dateTime" minOccurs="1" maxOccurs="1"/>
                    </xs:sequence>
                </xs:complexType>
            </xs:element>
            <xs:element name="body">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element name="vrcrt" minOccurs="1" maxOccurs="1">
                            <xs:complexType>
                                <xs:sequence>
                                    <xs:element ref="vrcrtdialect" minOccurs="1" maxOccurs="1"/>
                                    <xs:element ref="comment" minOccurs="1" maxOccurs="1"/>
                                    <xs:element ref="visualrepresentation" minOccurs="1" maxOccurs="1"/>
                                    <xs:element ref="vrcrtsockets" minOccurs="1" maxOccurs="1"/>
                                    <xs:element ref="adaptationbehaviour" minOccurs="1" maxOccurs="1"/>
                                </xs:sequence>
                            </xs:complexType>
                        </xs:element>
                    </xs:sequence>
                </xs:complexType>
            </xs:element>
        </xs:sequence>
    </xs:complexType>
</xs:element>

```

```

        </xs:sequence>
        </xs:complexType>
    </xs:element>
    </xs:sequence>
    </xs:complexType>
</xs:element>
</xs:sequence>
</xs:complexType>
</xs:element>

<xs:simpleType name="modeltypetype">
    <xs:restriction base="xs:string">
        <xs:enumeration value="vrcrt"/>
    </xs:restriction>
</xs:simpleType>

<!-- (B) The seven elements of the CRT definition -->

<!-- (B.1) the CRT dialect -->
<xs:element name="vrcrt dialect" type="vrcrt dialecttype"/>

<xs:simpleType name="vrcrt dialecttype">
    <xs:restriction base="xs:string">
        <xs:enumeration value="vrcrt"/>
    </xs:restriction>
</xs:simpleType>

<!-- (B.2) the comment field -->
<xs:element name="comment" type="xs:string"/>

<xs:element name="visualrepresentation">
    <xs:complexType>
        <xs:sequence>
            <xs:element name="shape" type="xs:string" minOccurs="1" maxOccurs="1"/>
            <xs:element name="colour" type="xs:string" minOccurs="1" maxOccurs="1"/>
        </xs:sequence>
    </xs:complexType>
</xs:element>

<xs:element name="vrcrtsockets" >
    <xs:complexType>
        <xs:sequence>
            <xs:element name="socket" minOccurs="1" maxOccurs="unbounded">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element name="name" type="xs:string" minOccurs="1" maxOccurs="1"/>
                        <xs:element name="uuid" type="xs:string" minOccurs="1" maxOccurs="1"/>
                        <xs:element name="mincardinality" type="xs:string" minOccurs="1" maxOccurs="1"/>
                        <xs:element name="maxcardinality" type="xs:string" minOccurs="1" maxOccurs="1"/>
                        <xs:element name="colour" type="xs:string" minOccurs="0" maxOccurs="1"/>
                    </xs:sequence>
                    <xs:attribute name="type" type="xs:string" use="required"/>
                </xs:complexType>
            </xs:element>
        </xs:sequence>
    </xs:complexType>
</xs:element>

<xs:element name="adaptationbehaviour">
    <xs:complexType>
        <xs:sequence>
            <xs:element name="usermodel" minOccurs="1" maxOccurs="1">

```

```

<xs:complexType>
  <xs:sequence>
    <xs:element ref="umvariable" minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>
</xs:element>
<xs:element name="galcode" type="xs:string" minOccurs="1" maxOccurs="1"/>
</xs:sequence>
</xs:complexType>
</xs:element>

<xs:element name="umvariable">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="umvarname" type="xs:string" minOccurs="1" maxOccurs="1"/>
      <xs:element name="public" type="xs:boolean" minOccurs="1" maxOccurs="1"/>
      <xs:element name="persistent" type="xs:boolean" minOccurs="1" maxOccurs="1"/>
      <xs:element name="type" type="xs:string" minOccurs="1" maxOccurs="1"/>
      <xs:element name="range" type="xs:anyType" minOccurs="0" maxOccurs="1"/>
      <xs:element name="default" type="xs:string" minOccurs="1" maxOccurs="1"/>
      <xs:element name="location" minOccurs="0" maxOccurs="1">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="web" minOccurs="0" maxOccurs="1"/>
            <xs:element name="remotecourse" type="xs:anyType" minOccurs="0" maxOccurs="1"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:element>
</xs:schema>

```

### 10.3 XML Schema for Virtual Reality AdaptationState (VRAS)

```

<?xml version="1.0" encoding="UTF-8"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema" targetNamespace="http://www.example.org/CAM-external-oct2009" xmlns:tns="http://www.example.org/CAM-external-oct2009" elementFormDefault="qualified">

  <element name="model" type="tns:modelType"></element>

  <complexType name="modelType">
    <sequence maxOccurs="1" minOccurs="1">
      <element name="header" type="tns:headerType" maxOccurs="1"
minOccurs="1"></element>
      <element name="body" type="tns:vrasBodyType" maxOccurs="1"
minOccurs="1"></element>
    </sequence>
  </complexType>

  <complexType name="headerType">
    <sequence>
      <element name="modeluid" type="tns:UIDtype" maxOccurs="1"
minOccurs="1">
      </element>
      <element name="modeltype" type="string" maxOccurs="1"
minOccurs="1">
      </element>
      <element name="authoruid" type="tns:UIDtype" maxOccurs="1"
minOccurs="1"></element>

```

```

        <element name="authorisation" type="tns:authorisationType" maxOccurs="1"
minOccurs="1"></element>
        <element name="creationtime" type="dateTime" maxOccurs="1"
minOccurs="1"></element>
        <element name="updatetime" type="dateTime" maxOccurs="1"
minOccurs="1"></element>
        <element name="title" type="string" maxOccurs="1" minOccurs="1"></element>
        <element name="description" type="string" maxOccurs="1" minOccurs="1"></element>
    </sequence>
</complexType>

<complexType name="vrasBodyType">
    <sequence>

        <element name="adaptationstate" type="tns:adaptationstateType" maxOccurs="1"
minOccurs="1"></element>
    </sequence>
</complexType>

<simpleType name="authorisationType">
    <restriction base="string">
        <enumeration value="read"></enumeration>
        <enumeration value="write"></enumeration>
        <enumeration value="readwrite"></enumeration>
    </restriction>
</simpleType>

<simpleType name="UUIDtype">
    <restriction base="string">
        <pattern
            value="[a-z0-9]{8}\-[a-z0-9]{4}\-[a-z0-9]{4}\-[a-z0-9]{4}\-[a-z0-9]{12}">
        </pattern>
    </restriction>
</simpleType>

<complexType name="adaptationstateType">
    <sequence>
        <element name="name" type="string" maxOccurs="1" minOccurs="1"></element>
        <element name="description" type="string" maxOccurs="1" minOccurs="1"></element>
        <element name="parameters" type="tns:parametersType" maxOccurs="unbounded"
minOccurs="1"></element>
        <element name="vrcode" type="tns:vrcodeType" maxOccurs="1"
minOccurs="1"></element>
    </sequence>
</complexType>
<complexType name="parametersType">
    <sequence>
        <element name="parameter" type="tns:parameterType" maxOccurs="1"
minOccurs="1"></element>
    </sequence>
</complexType>

<complexType name="parameterType">
    <sequence>
        <element name="type" type="string" maxOccurs="1" minOccurs="1"></element>
        <element name="defaultvalue" type="string" maxOccurs="1" minOccurs="1"></element>
        <element name="description" type="string" maxOccurs="1" minOccurs="1"></element>
    </sequence>
</complexType>

<complexType name="vrcodeType">

```

```

<sequence>
  <element name="code" type="string" maxOccurs="1" minOccurs="1"></element>
  <element name="triggeron" type="tns:triggeronType" maxOccurs="1"
minOccurs="1"></element>
  <element name="triggeroff" type="tns:triggeroffType" maxOccurs="1"
minOccurs="1"></element>
</sequence>
</complexType>

<complexType name="triggeronType">
  <sequence>
    <element name="vrtag" type="string" maxOccurs="1" minOccurs="1"></element>
    <element name="vrtriggerinstruction" type="string" maxOccurs="1"
minOccurs="1"></element>
    <element name="vrtriggervalue" type="string" maxOccurs="1"
minOccurs="1"></element>
    <element name="vrtriggertype" type="string" maxOccurs="1" minOccurs="1"></element>
  </sequence>
</complexType>

<complexType name="triggeroffType">
  <sequence>
    <element name="vrtag" type="string" maxOccurs="1" minOccurs="1"></element>
    <element name="vrtriggerinstruction" type="string" maxOccurs="1"
minOccurs="1"></element>
    <element name="vrtriggervalue" type="string" maxOccurs="1"
minOccurs="1"></element>
    <element name="vrtriggertype" type="string" maxOccurs="1" minOccurs="1"></element>
  </sequence>
</complexType>
</schema>

```

## 10.4 XML Schema for Virtual Reality RAW Model (VRRAW model)

```

<?xml version="1.0" encoding="UTF-8"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema" targetNamespace="http://www.example.org/CAM-
external-oct2009" xmlns:tns="http://www.example.org/CAM-external-oct2009"
elementFormDefault="qualified" >

  <element name="model" type="tns:modelType"></element>

  <complexType name="modelType">
    <sequence maxOccurs="1" minOccurs="1">
      <element name="header" type="tns:headerType" maxOccurs="1"
minOccurs="1"></element>
      <element name="body" type="tns:vrrawBodyType" maxOccurs="1"
minOccurs="1"></element>
    </sequence>
  </complexType>

  <complexType name="headerType">
    <sequence>
      <element name="modeluid" type="tns:UIDtype" maxOccurs="1"
minOccurs="1">
    </element>
      <element name="modeltype" type="string" maxOccurs="1"
minOccurs="1">
    </element>
      <element name="authoruid" type="tns:UIDtype" maxOccurs="1"
minOccurs="1"></element>

```

```

        <element name="authorisation" type="tns:authorisationType" maxOccurs="1"
minOccurs="1"></element>
        <element name="creationtime" type="dateTime" maxOccurs="1"
minOccurs="1"></element>
        <element name="updatetime" type="dateTime" maxOccurs="1"
minOccurs="1"></element>
        <element name="title" type="string" maxOccurs="1" minOccurs="1"></element>
        <element name="description" type="string" maxOccurs="1" minOccurs="1"></element>
    </sequence>
</complexType>

<complexType name="vrrawBodyType">
    <sequence>
        <element name="x3d" type="string" maxOccurs="1" minOccurs="1"></element>
    </sequence>
</complexType>

<simpleType name="authorisationType">
    <restriction base="string">
        <enumeration value="read"></enumeration>
        <enumeration value="write"></enumeration>
        <enumeration value="readwrite"></enumeration>
    </restriction>
</simpleType>

<simpleType name="UUIDtype">
    <restriction base="string">
        <pattern
            value="[a-z0-9]{8}\-[a-z0-9]{4}\-[a-z0-9]{4}\-[a-z0-9]{4}\-[a-z0-9]{12}">
        </pattern>
    </restriction>
</simpleType>
</schema>

```

## 10.5 XML Schema for Virtual Reality Learning Material (VRLM)

```

<?xml version="1.0" encoding="UTF-8"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema" targetNamespace="http://www.example.org/CAM-
external-oct2009" xmlns:tns="http://www.example.org/CAM-external-oct2009"
elementFormDefault="qualified" >

    <element name="model" type="tns:modelType"></element>

    <complexType name="modelType">
        <sequence maxOccurs="1" minOccurs="1">
            <element name="header" type="tns:headerType" maxOccurs="1"
minOccurs="1"></element>
            <element name="body" type="tns:vrlmBodyType" maxOccurs="1"
minOccurs="1"></element>
        </sequence>
    </complexType>

    <complexType name="headerType">
        <sequence>
            <element name="modeluuid" type="tns:UUIDtype" maxOccurs="1"
minOccurs="1">
            </element>
            <element name="modeltype" type="string" maxOccurs="1"

```

```

        minOccurs="1">
            </element>
            <element name="authoruid" type="tns:UIDtype" maxOccurs="1"
minOccurs="1"></element>
            <element name="authorisation" type="tns:authorisationType" maxOccurs="1"
minOccurs="1"></element>
            <element name="creationtime" type="dateTime" maxOccurs="1"
minOccurs="1"></element>
            <element name="updatetime" type="dateTime" maxOccurs="1"
minOccurs="1"></element>
            <element name="title" type="string" maxOccurs="1" minOccurs="1"></element>
            <element name="description" type="string" maxOccurs="1" minOccurs="1"></element>
        </sequence>
    </complexType>

    <complexType name="vrlmBodyType">
        <sequence>
            <element name="x3d" type="string" maxOccurs="1" minOccurs="1"></element>
        </sequence>
    </complexType>

    <simpleType name="authorisationType">
        <restriction base="string">
            <enumeration value="read"></enumeration>
            <enumeration value="write"></enumeration>
            <enumeration value="readwrite"></enumeration>
        </restriction>
    </simpleType>

    <simpleType name="UIDtype">
        <restriction base="string">
            <pattern
                value="[a-z0-9]{8}\-[a-z0-9]{4}\-[a-z0-9]{4}\-[a-z0-9]{4}\-[a-z0-9]{12}">
            </pattern>
        </restriction>
    </simpleType>
</schema>

```

## 10.6 XML Schema for Virtual Reality Learning Object (VRLO)

```

<?xml version="1.0" encoding="UTF-8"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema" targetNamespace="http://www.example.org/CAM-
external-oct2009" xmlns:tns="http://www.example.org/CAM-external-oct2009"
elementFormDefault="qualified" >

    <element name="model" type="tns:modelType"></element>

    <complexType name="modelType">
        <sequence maxOccurs="1" minOccurs="1">
            <element name="header" type="tns:headerType" maxOccurs="1"
minOccurs="1"></element>
            <element name="body" type="tns:vrlmBodyType" maxOccurs="1"
minOccurs="1"></element>
        </sequence>
    </complexType>

    <complexType name="headerType">
        <sequence>
            <element name="modeluid" type="tns:UIDtype" maxOccurs="1"

```

```

        minOccurs="1">
      </element>
      <element name="modeltype" type="string" maxOccurs="1"
        minOccurs="1">
      </element>
      <element name="authoruuid" type="tns:UUIDtype" maxOccurs="1"
minOccurs="1"></element>
      <element name="authorisation" type="tns:authorisationType" maxOccurs="1"
minOccurs="1"></element>
      <element name="creationtime" type="dateTime" maxOccurs="1"
minOccurs="1"></element>
      <element name="updatetime" type="dateTime" maxOccurs="1"
minOccurs="1"></element>
      <element name="title" type="string" maxOccurs="1" minOccurs="1"></element>
      <element name="description" type="string" maxOccurs="1" minOccurs="1"></element>
    </sequence>
  </complexType>

  <complexType name="vrloBodyType">
    <sequence>

      <element name="location" type="string" maxOccurs="1" minOccurs="1"></element>
      <element name="vw" type="tns:vwType" maxOccurs="1" minOccurs="1"></element>
    </sequence>
  </complexType>

  <simpleType name="authorisationType">
    <restriction base="string">
      <enumeration value="read"></enumeration>
      <enumeration value="write"></enumeration>
      <enumeration value="readwrite"></enumeration>
    </restriction>
  </simpleType>

  <simpleType name="UUIDtype">
    <restriction base="string">
      <pattern
        value="[a-z0-9]{8}\-[a-z0-9]{4}\-[a-z0-9]{4}\-[a-z0-9]{4}\-[a-z0-9]{12}">
      </pattern>
    </restriction>
  </simpleType>

  <complexType name="vwType">
    <sequence>
      <element name="x3d" type="string" maxOccurs="1" minOccurs="1"></element>
    </sequence>
  </complexType>
</schema>

```

## 10.7 XML Schema for VRResource for GALE and the VR plug-in

```

<?xml version="1.0" encoding="UTF-8"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema" targetNamespace="http://www.example.org/CAM-
external-oct2009" xmlns:tns="http://www.example.org/CAM-external-oct2009"
elementFormDefault="qualified">

  <element name="VRResource" type="tns:VRResourceType"></element>

```

```

<complexType name="VRResourceType">
  <sequence maxOccurs="1" minOccurs="1">
    <element name="idvw" type="tns:UUIDtype" maxOccurs="1" minOccurs="1"></element>
    <element name="titlevw" type="string" maxOccurs="1" minOccurs="1"></element>
    <element name="idconcept" type="tns:UUIDtype" maxOccurs="1"
minOccurs="1"></element>
    <element name="nameconcept" type="string" maxOccurs="1"
minOccurs="1"></element>
    <element name="idobj" type="string" maxOccurs="1" minOccurs="1"></element>
    <element name="locationvr" type="string" maxOccurs="1" minOccurs="1"></element>
    <element name="type" type="tns:vrresourceType" maxOccurs="unbounded"
minOccurs="1"></element>
  </sequence>
</complexType>

<complexType name="vrresourceType">
  <sequence>
    <element name="vras" type="tns:vrastype" maxOccurs="1" minOccurs="1"></element>
  </sequence>
</complexType>

<complexType name="vrastype">
  <sequence>
    <element name="idas" type="string" maxOccurs="1" minOccurs="1"></element>
    <element name="tagid" type="string" maxOccurs="1" minOccurs="1"></element>
    <element name="parameter" type="string" maxOccurs="1" minOccurs="1"></element>
    <element name="value" type="string" maxOccurs="1" minOccurs="1"></element>
    <element name="type" type="string" maxOccurs="1" minOccurs="1"></element>
  </sequence>
</complexType>

<simpleType name="UUIDtype">
  <restriction base="string">
    <pattern
      value="[a-z0-9]{8}\-[a-z0-9]{4}\-[a-z0-9]{4}\-[a-z0-9]{4}\-[a-z0-9]{12}">
    </pattern>
  </restriction>
</simpleType>

</schema>

```