Introducing Adaptive Hypermedia Techniques in 3D Educational Virtual Environments

Luca Chittaro and Roberto Ranon
HCI Lab, Department of Math and Computer Science
University of Udine, Italy
{chittaro | ranon}@dimi.uniud.it

Abstract

Highly interactive learning environments, such as 3D Educational Virtual Environments can help people in learning through direct experience by visualizing concepts and performing tasks in a reproduction of the real world or in completely fictional worlds that are suited to the learning task. For these applications, adaptivity can play an important role in increasing the effectiveness of the learning process and usability of the interface. In this paper, we present a system that is able to deliver personalized learning content in 3D Educational Virtual Environments by exploiting techniques that extend those employed the field of Adaptive Hypermedia by the well-known AHA! system. Moreover, we present a case study in which the proposed system is used to build an adaptive 3D application that teaches students how to build interactive 3D graphics content using the eXtensible 3D (X3D) ISO standard language.

1. Introduction

Highly interactive learning environments, such as 3D Educational Virtual Environments (hereinafter, EVEs) can help people in learning through direct experience by visualizing concepts and performing tasks in a reproduction of the real world or in completely fictional worlds that are suited to the learning task [3,6]. Moreover, compared to 2D multimedia, employing interactive 3D graphics allows for more informative representations of subjects or phenomena, offering the possibility of analyzing the same subject from different points of view.

Adaptivity can play an important role in increasing the effectiveness of the learning process and usability of the interface in EVEs. Personalized learning content and strategies can make e-learning closer to one-to-one tutoring, where a student is exposed to content, exercises and support that are tailored to her knowledge level and learning style. For example, an adaptive EVE for maintenance training could visualize in 3D a sequence of actions with a level of detail appropriate to the knowledge of the trainee, e.g. focusing on less practiced actions, while devoting less time and detail to well-known ones.

In the fields of Intelligent Tutoring Systems and Artificial Intelligence in Education, some researchers have proposed EVEs that include adaptive capabilities, e.g. for operations and maintenance training (the Steve project [6]), language learning (the Tactical Iraqi project [7]) and dialogue-based tutoring augmented with 3D interactive simulations (the AutoTutor-3D project [5]). The field of Adaptive Hypermedia [1] has particularly focused on generic adaptivity techniques, also exploring their application to education, but, as we will explain, these techniques cannot be straightforwardly applied to EVEs.

The purpose of this paper is to explore whether adaptive hypermedia approaches can inspire new and general adaptivity techniques for EVEs. In the past, only a few preliminary attempts at bringing together 3D Virtual Environments (hereinafter, VEs) and hypermedia-based adaptivity have been made [2], and no one has yet explored their combination in the context of EVEs. However, in educational hypermedia as well as EVEs, the learning process is based on student-driven navigation and interaction within the educational material. It makes thus sense, as a promising strategy for managing adaptivity into EVEs, to start from adaptive hypermedia techniques and explore how they could be changed or extended in the new context.

From a technical point of view, hypermedia-based adaptivity techniques have been mostly developed for material consisting of interconnected pages of text and 2D multimedia. In this

context, adaptive hypermedia techniques mainly operate by changing the page structure and content (adding or removing text fragments, changing the order in which items appear on the page,…) and by acting on links (e.g., highlighting the more suited ones and hiding those that the student should not currently follow). A VE is structured and navigated in a completely different way, therefore adaptive hypermedia techniques do not straightforwardly apply. For example, one cannot add 3D objects as easily as inserting paragraphs one after another in a page without the risk of creating a too cluttered, not understandable, or not navigable VE.

In general, which kinds of adaptation are effective in an EVE is still an open question. While the field of adaptive educational hypermedia has defined and experimented general personalization strategies (e.g., one can add or remove explanation details by modifying text or suggest next suitable topics by manipulating links), it is not clear which could be their effective counterparts in an EVE. For example, how do we modify the EVE such that a topic is explained or a task performed at the appropriate level for a specific student?

In this paper, we present a system that is able to deliver personalized learning content in EVEs, including Web-based ones. The system exploits AHA! (Adaptive Hypermedia Architecture) [4], a well-known adaptive hypermedia educational framework, to build and maintain the student model and to select and personalize learning content. However, before content is delivered to the student, the system is able to translate it into a format suitable for presentation and interaction in an EVE. Moreover, by monitoring and recording student's interactions within the EVE, the system is able to track how the student is progressing and update the student model accordingly in more accurate ways than typical adaptive educational hypermedia.

Finally, we present a case study in which the proposed system is used to build an adaptive EVE that teaches students how to build interactive 3D graphics content using the eXtensible 3D (X3D) ISO standard language [10]. The case study allows us to explore a number of different personalization techniques for EVEs.

The paper is structured as follows. In Section 2, we discuss the state of the art in combining adaptivity and VEs. In Section 3, we describe the architecture of the system and its functioning. In Section 4, we describe the case study and present the adaptation techniques we have developed for EVEs. In Section 5, we discuss the limitations of our approach, and future work aimed at overcoming them.

2. Adaptive Virtual Environments

Augmenting VEs with adaptive capabilities could greatly increase their usability and effectiveness. For example, an adaptive navigation support system could effectively help users with different navigation abilities in typically difficult tasks, such as finding targets, orienting themselves, and gaining spatial knowledge of the environment.

The main reason why it is difficult to develop adaptive VEs is that general adaptive techniques have been mostly developed for 2D interfaces (in most cases, pages of text and other multimedia) and for hypermedia information spaces, where users mostly read and navigate from one page to another by following hyperlinks. In a VE, content (mainly composed by 3D models) is organized in a 3D space, and is not browsed by following links, but by seamless movements in 3D space and other kinds of (possibly complex) interactions with 3D objects. Therefore, well-known adaptive presentation (e.g., page fragments manipulation) and navigation support (e.g., link annotation) techniques [2] cannot be straightforwardly applied.

However, some research projects have tried to introduce adaptive hypermedia-based techniques in VEs, particularly focusing on Web-based VEs (also called Web3D sites). Examples described in [2] include a software architecture for dynamic construction of personalized 3D Web content (that has been applied to e-commerce as well as virtual museums) and methods for personalized navigation support and adaptive content presentation in 3D VEs. In this section, we summarize the main results that have been obtained so far [2].

From the point of view of building and updating the user model, the main difference between VEs and adaptive hypermedia concerns how user actions determine updates of the user model. In adaptive hypermedia, the typical assumption is that every requested page (with all included content) will be read by the user (and the user model can be thus updated accordingly). Employing a similar assumption with VEs (e.g. updating the user model by assuming that the user has experienced all content in a part of the VE) is likely to make the user model inaccurate. In VEs, users might indeed not see some objects because they are occluded by other objects (from the user's viewpoint) or simply do not notice them while navigating. Moreover, when some object manipulation is possible, users might not perform it or might do it in unexpected ways. For example, in a medical training application where the trainee is required to virtually perform a certain sequence of actions with virtual medical tools, one would like to update the user model according to how actions were actually performed. A solution we proposed exploits specific time-stamped users' actions in the VE (e.g., movements, objects clicked,…) to trigger user model updates. For example, one can record user's position and orientation in a VE every few seconds to update the user model depending on which parts of the VE have been actually seen.

From the point of view of adaptive navigation support, there are methods that, given a list of most suitable objects/places in the VE, are able to guide users' navigation towards them. One solution employs virtual characters acting as VE guides, while another derives ideal viewpoints from which suitable objects can be observed, and then uses constrained navigation techniques or annotations (such as arrows or flashlights) to guide users.

From the point of view of adaptive presentation of content, current techniques rely on adaptively inserting, deleting or modifying code fragments that draw, position, and move 3D objects, or define interactive behaviors in the VE. To simplify the definition of adaptive VEs, some approaches propose to adopt high-level languages to specify a 3D scene and its adaptive variants, and then translate the adapted version into the required code format before it is visualized. However, using adaptive fragments techniques is more difficult than in adaptive hypermedia. Text fragments or images can be simply juxtaposed in a page, with the only possible drawback of being unable to preserve a good graphic layout. On the contrary, special care has to be taken in the case of 3D content to preserve a meaningful and understandable 3D space. Once relevant objects have been chosen, one needs to properly arrange them in 3D space (and time, if they involve animations). For example, included objects should not intersect each other, have to be adequately visible from the positions the user will take in space, and free space has to be enough for the user to move. Unfortunately, it is very difficult to develop general algorithms for these purposes. This forces one to limit the set of possible adaptations to a few variants that are guaranteed to be safe with respect to navigability and understandability of a 3D space or to implement adaptation strategies that might work only in a specific VE.

Coming back to EVEs, although studies have been carried out to evaluate their effectiveness (e.g. [8,9]), and we will report about them in the discussion at the end of the paper, there are presently no studies that specifically investigate the effect of adaptation techniques on EVEs users. For this reason, we can hypothesize which adaptations might be useful and which might be counterproductive mainly on the basis of general Human-Computer Interaction knowledge. For example, it is likely that several changes in the navigational structure of a VE will disorient the user and will make it much harder to learn how to navigate the environment. Therefore, structural changes need to be chosen carefully and be limited in scope and frequency. Moreover, user's experience in VEs should be as continuous as possible to maintain user engagement, while adaptive changes among pages in hypermedia are not (or much less) perceived as annoying breakdowns since the experience is already 'divided into pages'. Therefore, changes in the position, appearance or behavior of visible objects while the user is visiting the VE should be carefully performed, otherwise they will likely turn out as annoying or counterproductive for the user's experience.

3. The Proposed System

The architecture of the system we propose is shown in Figure 1. We chose the *AHA!* Engine [4] as the adaptive component, because it is a well-known and mature open-source adaptive hypermedia platform that has been used in several educational applications. The user interacts with the personalized VE through a Web browser with a X3D plug-in; relevant usage data (such as user position, orientation, and mouse actions) are recorded by a *Usage Data Sensing* component (which runs inside the VE) and then sent to a *Usage Data Analysis* component, which turns them into User Model updates requests to AHA!. Content requests from the user's browser are forwarded to AHA!, which sends the resulting personalized content to a *Content Transformer* component, that in turn may apply further changes to personalized content before it is passed to the user's browser. All communications among components are carried out through HTTP connections, therefore the system can be used in a Web-based scenario. In the following, we describe in more detail each component of the system. For reasons of space, we summarize in a few sentences the functioning of AHA!, and refer the reader to [4] for a full-detail description.

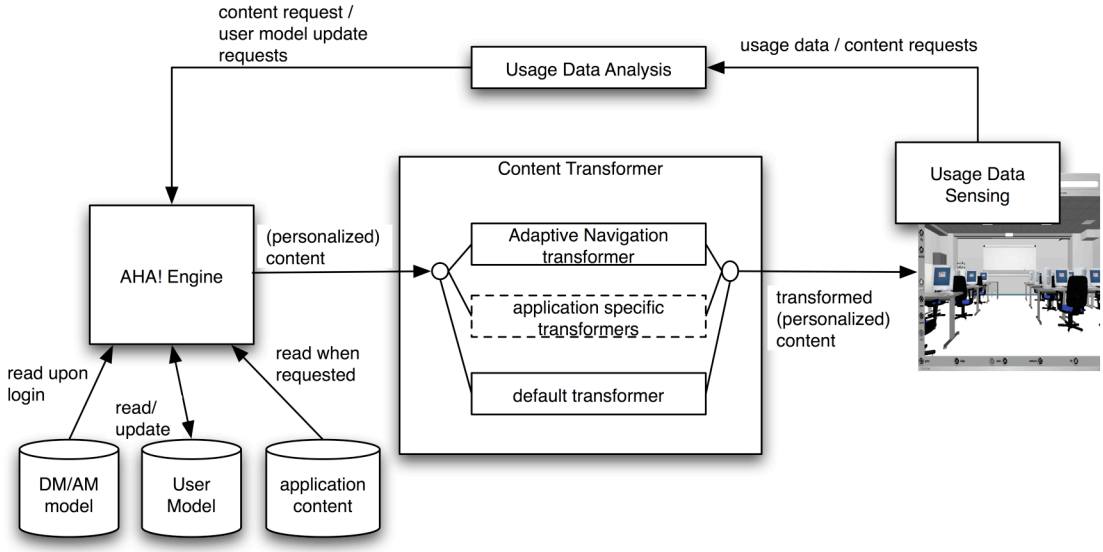

Figure 1. Architecture of the system

The AHA! engine uses three types of information. The combined *Domain Model* (*DM*)/*Adaptation Model* (*AM*) represents the conceptual structure of the educational application (in terms of a graph of concepts with attributes and relations among them) and adaptation rules. When the user is interacting with the application, the *User Model* (*UM*) is constantly updated, e.g. to reflect the degree of knowledge the user has gained about each concept. DM/AM and UM are used to decide which application file or “page” to retrieve upon a request from the user (and which fragments, e.g. text or images, to conditionally include in that page). AHA! reacts to HTTP requests and sends HTTP responses that contain a document (typically in HTML or XHTML, but any XML format is in principle allowed). The direct interaction between the user and AHA! thus happens only on a page-by-page basis.

Each time AHA! receives a HTTP request, a concept (specified in the request) is retrieved from the DM. The AHA! engine executes rules from the AM, starting from the rules associated to the *access* attribute of the requested concept/page. These rules can cause updates to attributes of UM concepts. A concept may have just one associated page, or it may have several, one of which is adaptively selected. The selected page is retrieved and processed according to AM and UM, by conditionally including fragments and/or objects, and by deciding the suitability of *conditional* links (e.g., special <a> tags in HTML) to other concepts or pages (e.g., in Web pages suitability determines how the link anchor is annotated or colored).

By encoding a VE in a XML format (e.g., using the X3D language [10]), we can use AHA! to personalize the VE, provided that we associate each concept in the DM with a proper part of the VE code (e.g., an object, an animation or an interactive behavior). When the user's client

(in the X3D case, a Web browser with an X3D plug-in) requests that concept, the corresponding VE code is personalized according to the UM and forwarded to the client, by which it is displayed. With AHA!, the possible adaptations can be expressed with the conditional fragments mechanism, i.e., for each adaptive code fragment, one needs to specify alternative variants, the conditions under which a variant will be inserted in the resulting code, and let AHA! decide which one to select on the basis of the UM and AM.

The main problem in employing AHA! with VEs is that no adaptive navigation or interaction support can be available, because, as discussed above, navigation and interaction in VEs are not performed through textual hyperlinks. Moreover, we cannot use high-level formats to specify adaptive VE code (since AHA! cannot translate them before they are sent to the user). Finally, the UM would be updated only when a concept (and its associated content) is requested. As discussed in the previous section, this is too limiting in the case of VEs. To overcome these problems, our system includes other components that mediate the communication between AHA! and the user's client.

In particular, our system exploits a *Usage Data Sensing* component, whose purpose is to monitor user's interaction with the VE and send the relevant events to the *Usage Data Analysis* component. The *Usage Data Sensing* component is implemented by a set of X3D sensors (i.e., primitives that sense interactions with the VE, such as the ProximitySensor [10] for tracking user's position) and scripts which select and send relevant usage events. The type, number and specific settings of X3D sensors depend on the type and number of usage data that needs to be collected for a specific application. In the simplest case, one would need one sensor for each event that has to be sensed (user's position, user's collisions with an object, mouse actions on an object, or visibility of an object). By combining the output of multiple sensors in a script, one can obtain higher level sensing of the user's actions. For example, a complex action that requires a sequence of clicks and drags can be monitored by using appropriate sensors to detect these low-level events. Then, a script receives the sensors' output, recognizes the correct sequence and sends the resulting high-level event to the *Usage Data Analysis* component, which determines the needed UM updates and asks AHA! to perform them by accessing concepts or directly increasing the knowledge of specific concepts.

The purpose of the *Content Transformer* component is to allow formats different from X3D to be used for specifying VE content. It is composed by a number of alternative transformers, which modify, through XSL transformations, the personalized XML content generated by AHA!. In this way, we can use any XML format within AHA! and then use a proper Transformer to convert to X3D or provide alternative X3D translations for the same content (e.g. implementing different visual presentations in the VE). We have currently implemented an Adaptive Navigation transformer that introduces navigation support into a X3D VE, and a default transformer, which simply forwards the content requested to AHA!. The latter is useful for any content that does not need to be adapted (e.g., images) or that AHA! is already able to fully adapt (e.g., HTML pages). In the following section, we will also introduce an application-specific transformer that produces adaptive visualizations of X3D source code. Transformers can also be put in a pipeline to perform a sequence of different transformations of AHA! output.

Finally, adaptive navigation and interaction support are implemented by using conditional <a> tags (which AHA! can recognize and personalize) to enclose a 3D object code (for navigation support) or the code that implements an interaction (e.g., the possibility of clicking an object). Once AHA! has determined the suitability of the object or interaction, the Adaptive Navigation Transformer will remove the <a> tag and introduce code to draw user's attention toward the objects or actions that are considered suitable according to the UM.

4. An EVE for learning the X3D language

By using the system proposed in the previous section, we have developed an EVE that teaches interactive 3D graphics programming allowing students to learn through direct interaction with the environment. Among the available technologies for building interactive 3D graphics applications, our case study focuses on teaching the X3D language: X3D integrates

3D graphics, 2D graphics, text, and multimedia into a coherent model, and combines them with scripting and network capabilities [10]. The language includes most of the common primitives used in 3D applications, such as geometry, light sources, viewpoints, animation, material properties, and texture mapping. One of the available formats for X3D documents consists of an XML file that describe 3D objects and VEs using a graph structure called hierarchical scene graph, a data structure common to high-level 3D graphics formats and libraries. Entities in the scene graph are called *nodes*. X3D defines several node types, including geometry primitives, appearance properties, sound and video, and nodes for animation and interactivity. Nodes store their properties in *fields*, which can contain different types of data, e.g. single integers, 3D rotations and arrays of vertices.

Traditional learning of interactive 3D graphics programming is accomplished through textual explanations (e.g., textbooks or Web pages explaining how the chosen language works, the syntax and semantics of commands or language primitives), images (e.g., screenshots) and (small) code examples that demonstrate language features and usage. Since code examples are fundamental to fully understand a language, an interesting feature of adaptivity would be to focus the user on examples that are more suitable to her knowledge level, present them in a personalized way, and use them as a motivation to learn unknown (or partially known) language features. In the following, we discuss how we have exploited our architecture to implement such adaptive learning strategies. Since, in this case study, the goal is more to learn abstract concepts than abilities in the VE, we will use the Usage Data Sensing and Analysis components only to a basic level.

The 3D space of our EVE reproduces a real-world computer science classroom (Figure 2) to situate the learning experience in a well-known environment. However, it could be possible to personalize also the 3D space according to the preferences of the student. For example, if the student is interested in architectural applications of X3D, the EVE could reproduce a virtual building or a virtual city.

The student can walk in the EVE and click on any object to see, in a separate window, the X3D code that implements that object, thus learning from actual examples. Code visualization is adaptive (see section 4.2 and the example in Figure 3), i.e. it depends on the student's level of X3D knowledge which is represented in the UM (see section 4.1). In the code visualization, the student can click on any node and get a HTML page explaining the node syntax and semantics. The explanation is also personalized to the student's knowledge level using standard adaptive hypermedia techniques, but we do not consider this here**,** since we are focusing on EVEs.

Based on the student's understanding of the X3D language, some 3D objects in the environment may be more suitable to examine than others. Our EVE thus focuses user’s attention by visually highlighting (see section 4.3) those 3D objects that demonstrate language features the student is ready to learn. Moreover, the visual appearance of suitable objects is also adapted (see section 4.4) to eliminate details that might distract or hinder learning (e.g., textures, when the student is not ready to learn them) or add details that might be useful for learning (e.g., indicating the position of light sources which is not visible in VEs).

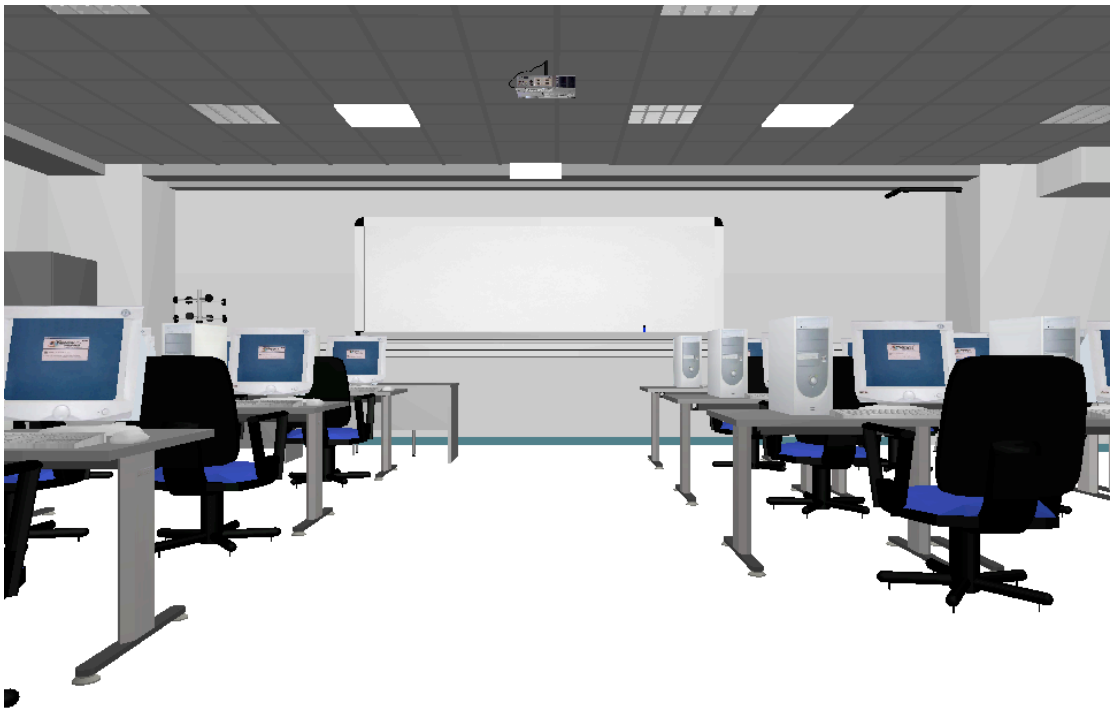

Figure 2. The EVE for learning X3D as seen from the starting position of the student.

4.1 DM/AM, UM and Application Content

The DM model of the developed EVE is composed by concepts related to the X3D language, falling in three categories:

- general, high-level concepts. They refer to general interactive 3D graphics topics (such as geometry, texture, transformations), and are associated with HTML pages explaining them.
- language specific concepts. Each of them concerns one of the X3D nodes, field data types, or statements, and is associated with a HTML page explaining it.
- example concepts. Each of them concerns one of the 3D objects in the VE, and is associated with the corresponding X3D code file, which may include adaptively selected fragments (see Section 4.4).

Figure 3 shows a portion of the DM model visualized as a graph, where nodes are concepts, and arcs are relations among concepts. We use three types of relation: (i) *part-of*, e.g., the concept *Box*, which refers to the X3D Box node for building box-shaped geometries, is part of the more general concept *GeometryNodes*, i.e. nodes to specify the geometry of an object, (ii) *prerequisite*, e.g., the concept *Appearance*, which is a node to control the visual appearance of an object, is a prerequisite for nodes that control specific appearance features, such as the concept *ImageTexture,* which concerns a node to apply textures, and (iii) *exemplifies*, e.g., *PC Case* objects in Figure 2 exemplify, among others, the *Box* and *ImageTexture* concepts.

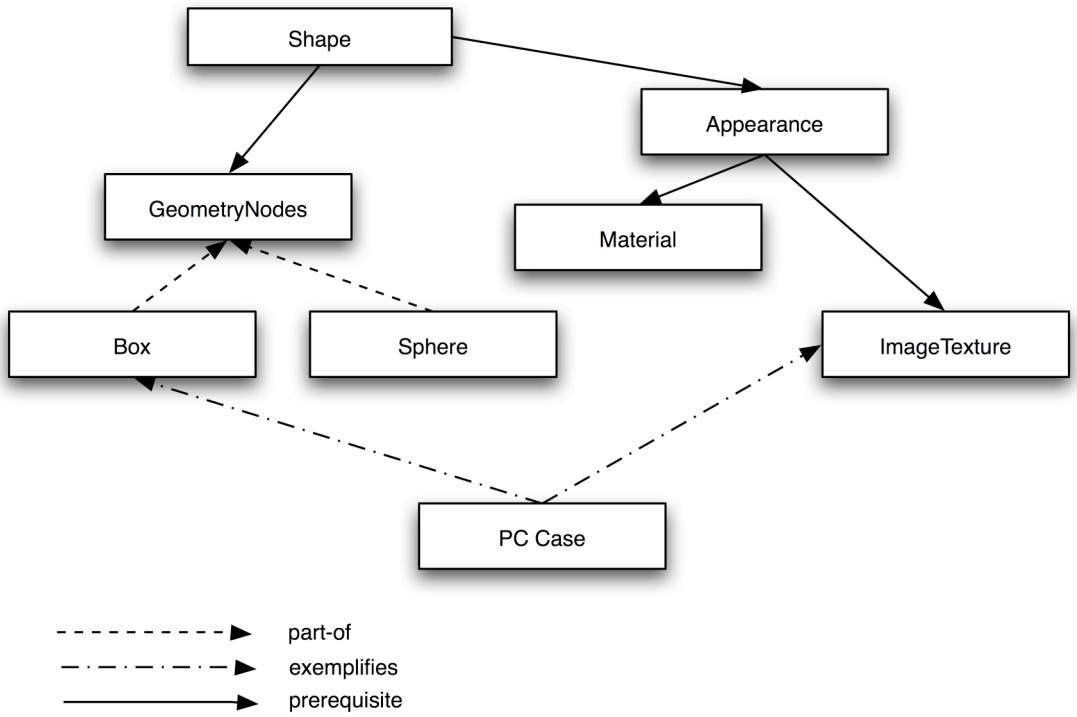

Figure 3. Portion of the DM used in this case study.

The AM controls how student's knowledge about each concept, stored in the UM, increases. In particular, knowledge about a concept is increased by either reading HTML pages associated to concepts (for concepts associated with X3D features and nodes, or for high-level concepts) or by looking at exemplifying code. However, to gain full knowledge of an X3D language feature, the student has to read its explanation as well as see some examples.
By default, AHA! associates attributes such as *visited* (has the associated page been visited?), *knowledge* (how much the concept is known by the student?), and *suitability* (are prerequisites sufficiently known?) to each defined concept. The part-of, prerequisite and exemplifies relations are then used by AHA!, in response to actions of the student, to propagate knowledge and suitability updates in all UM concepts.

The X3D main file that defines the EVE contains statements that include code (using the X3D Inline node) for all example concepts in the UM. As a result, when the user enters the EVE, all exemplifying objects are requested to AHA!, possibly adapted, and displayed in the browser.

4.2 Adaptive Code Visualization

Code presentation uses well-known adaptive text presentation techniques [1] (an example is shown in Figure 4) that are able to maintain context, i.e., adaptively highlight more relevant information while still allowing for visualization or access to less relevant information. This is fundamental in our case, since completely hiding parts of the X3D code for an object would likely confuse or misinform the student. In particular, we use:

- syntax-highlighting color-coded visualization for code that the student already understands or is suitable to learn (e.g., the Box node in Figure 4). Code which is unclear and not yet suitable to learn is displayed in light grey (e.g., the Viewpoint node in Figure 4) .
- *scaling fragments* [1], i.e., using text size to emphasize content based on relevancy, to encode the suitability of X3D nodes concepts (the greater the size, the more a node is suitable for learning; e.g., the Box node in Figure 4 is the more suitable node to learn);
- *stretchtext* [1], i.e., using placeholders to highlight the presence of and allow access to secondary information, for hiding code fragments that the student cannot currently

understand and are longer than one line (e.g., in Figure 4, the content of the Appearance node has been hidden). This makes it easier for the student to focus on the relevant parts of the code, while not preventing access to full code.

```
<X3D version='3.0' profile='Immersive'>
  <head>
    <meta name='created' content='27 August 2000'/>
    <meta name='modified' content='22 February 2005'/>
    <meta name='description' content='A texture-mapped Box example'/>
  </head>
  <Scene>
    <Viewpoint description='Texture test on a basic shape' position='0 0 5'/>
    <Shape>
      <Appearance> ... </Appearance>
      <Box size='2 2 2'/>
    </Shape>
  </Scene>
</X3D>
```

Figure 4. Personalized code visualization. This X3D document visualizes a box-shaped object with a certain appearance (whose code fragment has been hidden). Black arrowheads can be clicked to hide/show code fragments.

4.3 Adaptive Navigation Support

Adaptive navigation support visually highlights objects that are more suitable for code exploration. More specifically, highlighting is accomplished by surrounding the object with a glowing green wireframe marking the bounding volume, as shown in Figure 5.

The objects to be highlighted are determined when the main X3D file of the EVE is requested to AHA!, e.g., when the student loads the EVE. In this file, each statement that includes an example concept (i.e., a 3D object) is surrounded by AHA! conditional <a> tags (note that this is not allowed in X3D, but we use the Adaptive Navigation transformer component to deal with this potential problem). During the adaptation of the main file, the AHA! engine checks the UM to decide the suitability of each conditional link in the file, and stores the resulting suitability value (which can be good, neutral or bad) in the class attribute of the <a> tag. The adapted main file is then passed to the Adaptive Navigation transformer, that removes the <a> tags and introduces the code for drawing wireframe bounding volumes for any exemplifying object that AHA! deemed as good.

Since the UM might change during a visit to the EVE (e.g., because the student has examined some examples and read HTML node explanations), the main file is reloaded in the background at fixed time intervals, so that the list of suitable objects (and their highlighting) is consistent with the student's current UM.

4.4 Adaptive Object Visualization

Adaptive object visualization changes, on the basis of the UM, the X3D code that is loaded by the browser, and therefore determines how an object is visualized and behaves in the EVE. This feature is implemented by using AHA! optional fragments, i.e. specifying possible variants in the file and letting AHA! choose which one to select depending on the UM.

Figure 5. The green wireframe box visually highlights an object suitable for code exploration.

In our case study, we exploit this feature to remove or add visual details and/or behaviors to the objects. Examples where this adaptive feature is used include:

- visually showing the position in the EVE of X3D nodes that are not meant to visualize a geometry (e.g., light sources) so that the student can explore their code;
- remove features that are not currently suitable for the student's knowledge level, e.g. textures, materials, animations, interactive capabilities,…
- add interactions that are useful to learn some concept, e.g. the possibility for the student of translating, rotating or scaling objects to learn about X3D nodes that perform transformations.

Figure 6 shows an adaptive X3D code fragment that adds a yellow semi-transparent sphere to the EVE to visually show where a light source is positioned (see Figure 7 for the visual result).

```
…
<Group>
        <PointLight attenuation='0 0.3 0' radius='12'/>
        <if expr=”x3d.lights.suitability&gt;70”>
        <block>
        <!—the following sphere has been inserted just to show light position in the VE -->
        <Shape>
                <Sphere/>
                <Appearance>
                        <Material emissiveColor='1 1 0' transparency='.3'  />
                </Appearance>
        </Shape>
        </block>
</Group>
…
```

Figure 6. X3D code fragment to adaptively insert a yellow semi-transparent sphere where a light is positioned. The *PointLight* node inserts a point light source in the 3D scene. The *if* element is used in AHA! to conditionally insert the fragment enclosed by the *block* element. Therefore the yellow semi-transparent sphere that is defined by the *Shape* node and enclosed nodes is inserted only if the suitability of the high-level concept *lights* is greater than 70%.

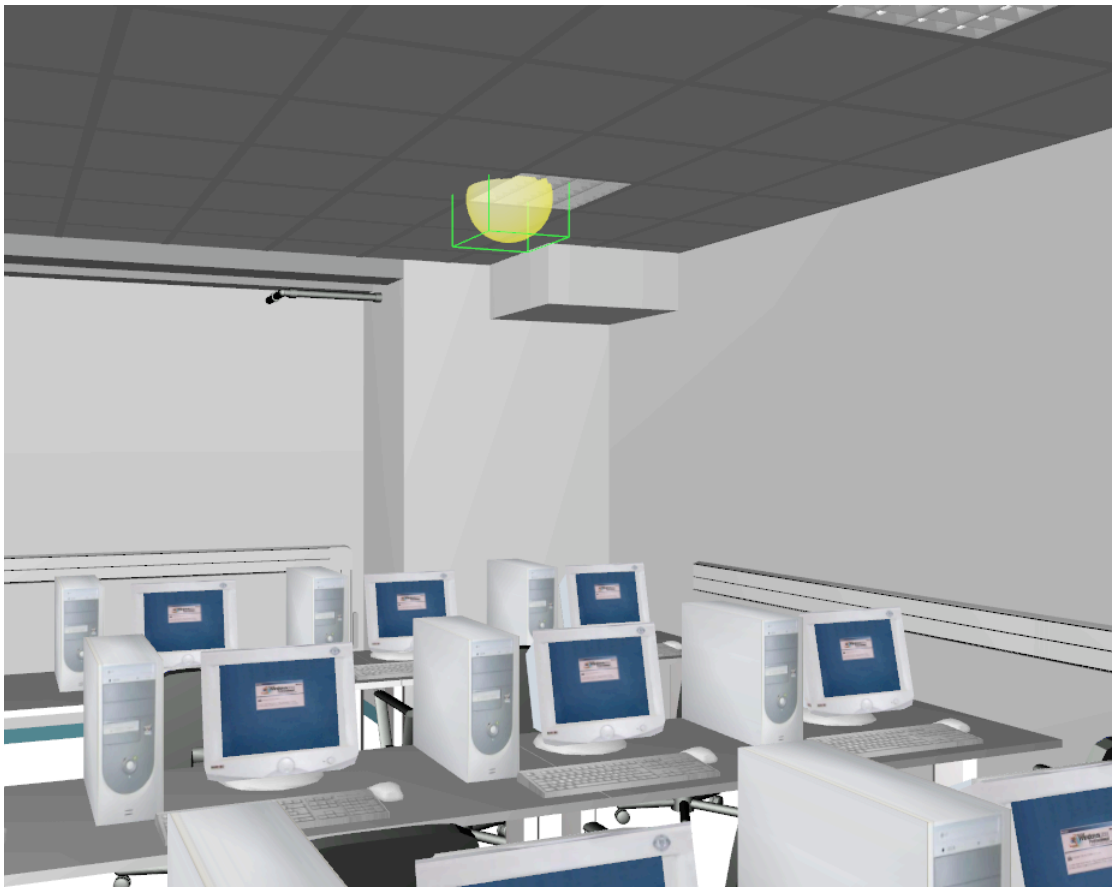

Figure 7. Showing the position of a light source in the EVE using the adaptive code fragment of Figure 6.

5. Discussion and Conclusions

In this paper, we have presented a system that is able to deliver personalized learning content in EVEs, and described its application to a specific domain.

Although no formal evaluation of the system has been carried out yet, the X3D learning application will be used as a supporting tool during an X3D course at our university. This will allow us to evaluate the learning gains the system might be able to produce in a real class setting.

Past evaluations of EVEs have shown encouraging results [5,8,9]. In particular, AutoTutor-3D has been evaluated and shown to produce statistically significant learning gains [5]. In the employed procedure, students carried out a pretest on the subject matter, then interacted with the system for a given amount of time, and finally completed a post-test to assess learning advancements. This condition was compared to conditions where students read a textbook for an equivalent amount of time or used reduced versions of the system. In our case, it could be interesting to use the same experimental procedure to assess the learning gains with respect to textbooks (or even no training at all), as well as to compare different versions of the system where adaptive features are enabled or disabled to specifically evaluate their effectiveness. Moreover, evaluations of Tactical Iraqi [8] and AutoTutor-3D [9] highlight the importance of providing users with encouragements to interact and feedback about learning progress. Therefore, we plan to extend the system and the X3D case study by introducing an embodied mentor [5,6] with the aim of improving learners' motivation and retention. In this case, adaptivity could also be applied to the mentor's appearance and behavior in the EVE.

With respect to other EVEs mentioned in the paper, it is interesting to note that the techniques proposed by our system can be considered complementary rather than alternative. While our system surely lacks sophisticated pedagogic mechanisms as the ones employed by AutoTutor-3D or engaging gameplay as in Tactical Iraqi, it provides general 3D content adaptation techniques that could be effectively integrated into other systems. More specifically, adaptive navigation support and adaptive object visualization techniques could be employed in systems such as Steve, Tactical Iraqi or AutoTutor-3D to focus the user towards more suitable actions, or to tailor 3D content visualization according to the student knowledge.

While using AHA! as the adaptive engine in our system has been useful to include adaptive hypermedia techniques into EVEs, it brought also some limitations. One of the most important issues is the fact that AHA! has no explicit representation of student's misconceptions and thus makes it difficult to relate users' actions in the EVEs with reductions of knowledge level in the user model. Future work will be thus also directed at evaluating how our system can be extended to overcome such limitation.

References

1. Brusilovsky P., Adaptive hypermedia. *User Modeling and User Adapted Interaction*, *Ten Year Anniversary Issue*, No. 11, Issue 1/2, 2001, pp. 87-110.
2. Chittaro L., Ranon R. Adaptive 3D Web Sites. In: Peter Brusilovsky, Alfred Kobsa, Wolfgang Nejdl (eds.): The Adaptive Web: Methods and Strategies of Web Personalization, Springer-Verlag, in press, 2007.
3. Chittaro L., Ranon R. Web3D Technologies in Learning, Education and Training: Motivations, Issues, Opportunities, Computers & Education Journal, No. 49, Issue 2, pp. 3 -18.
4. De Bra P., Aerts A., Berden B., De Lange B., Rousseau B., Santic T., Smits D., Stash N., AHA! The Adaptive Hypermedia Architecture. Proceedings of the ACM Hypertext Conference, Nottingham, UK. ACM Press (2003) 81-84.
5. Graesser A.C., Chipman P., Haynes B.C. and Olney, A. AutoTutor: An intelligent tutoring system with mixed-initiative dialogue. *IEEE Transactions in Education*, 48, 2005, pp. 612-618.
6. Johnson W.L., Rickel J.W., and Lester J.C.. Animated Pedagogical Agents: Face-to-Face Interaction in Interactive Learning Environments. *International Journal of Artificial Intelligence in Education,* 11, 2000, pp. 47-78.
7. Johnson W.L., Vilhjálmsson H. H., and Marsella S. Serious games for language learning: How much game, how much AI? Proceedings of the 12th International Conference on Artificial Intelligence in Education, July 18-22, Amsterdam, The Netherlands, 2005.
8. Johnson, W.L. and Beal, C. Iterative evaluation of an intelligent game for language learning. Proceedings of the 12th International Conference on Artificial Intelligence in Education , July 18-22, Amsterdam, The Netherlands, 2005.
9. Kim H., Graesser A., Jackson T., Olney A. and Chipman P. The effectiveness of computer simulations in a computer based learning environment. Proceedings of the World Conference on E-Learning in Corporate, Government, Healthcare, and Higher Education, 2005, pp. 1362-1367.
10. X3D International Standard (2004). X3D framework & SAI. ISO/IEC FDIS (Final Draft International Standard) 19775:200x. Available at www.web3d.org/x3d/specifications (last access on March 2007).