14

Adaptive 3D Web Sites

Luca Chittaro and Roberto Ranon

HCI Lab, Dept. of Math. and Computer Science, University of Udine,
via delle Scienze 206, 33100 Udine, Italy
{chittaro, ranon}@dimi.uniud.it

Abstract. In recent years, technological developments have made it possible to build interactive 3D models of objects and 3D Virtual Environments that can be experienced through the Web, using common, low-cost personal computers. As in the case of Web-based hypermedia, adaptivity can play an important role in increasing the usefulness, effectiveness and usability of 3D Web sites, i.e., Web sites distributing 3D content. This paper introduces the reader to the concepts, issues and techniques of adaptive 3D Web sites.

14.1 Introduction

In recent years, technological developments have made it possible to build interactive 3D models of objects and 3D Virtual Environments (hereinafter, 3D VEs) that can be experienced through the Web, using common, low-cost personal computers. As a result, 3D content is increasingly employed in different Web application areas, such as education and training [18, 30, 40], e-commerce [26, 36], architecture and tourism [42, 44], virtual communities [2,45] and virtual museums [4].

Web sites distributing 3D content (hereinafter, we call them *3D Web sites* for simplicity) can be divided into two broad categories:

- sites that display interactive 3D models of objects embedded into Web pages, such as e-commerce sites allowing customers to examine 3D models of products [26], and
- sites that are mainly based on a 3D VE which is displayed inside the Web browser, such as tourism sites allowing users to navigate inside a 3D virtual city [44].

In the first case, the primary information structure and user's interaction methods are still based on the hypermedia model, with the additional possibility of inspecting 3D objects. In the second case, the primary information structure is a 3D space, within which users move and perform various actions. For example, a furniture e-commerce site might be based on a 3D virtual house where users can walk, choose furniture from a catalogue, and place it in the various rooms [36].

3D Web sites are not meant to substitute the hypermedia model which is the mainstream in today's Web, but they can be more effective when there is added value

in interacting with a 3D visualization, or in providing a first-person virtual experience close to a real-world one. For example, in the case of e-commerce, 3D models give customers the ability to visually inspect, manipulate, try and customize products before purchasing as they are accustomed to do in the real world [27]. In the case of cultural heritage, a Web museum implemented as a 3D VE allows one not only to display the museum items, but also to convey their "cultural setting" by placing them in a proper environment.

As in the case of Web-based hypermedia, adaptivity can play an important role in increasing the usefulness, effectiveness and usability of 3D Web sites. For example, an intelligent adaptive navigation support system could help users with different navigation abilities in finding targets, orienting themselves, and gaining spatial knowledge of the environment. Unfortunately, there are currently no well-established techniques or commercial tools to build adaptive 3D Web sites. Moreover, because of conceptual and technical peculiarities of 3D Web sites, most approaches, techniques and software tools developed for the Adaptive Web cannot be straightforwardly applied to personalize 3D Web content, navigation and presentation. However, some research projects have addressed the issue of adaptivity for 3D Web sites. For example, a first software architecture [17] for dynamic construction of personalized 3D Web content has been proposed and applied to e-commerce [14,16] and virtual museums [13]. Some researchers have developed methods for personalized navigation support [12,27], adaptive interaction [11] and content presentation [24] in 3D VEs. Recently, there have been some attempts at experimenting with general-purpose frameworks for Web adaptivity to deliver personalized 3D content [15,21].

This Chapter will introduce the reader to the concepts, issues and techniques of adaptive 3D Web sites. We will mainly focus on 3D Web sites based on 3D VEs, since this category is the most general and complex one (but most of the techniques we will present can be applied also to Web sites with interactive 3D objects). The Chapter is structured as follows. Section 14.2 provides an introduction to 3D Web sites for the novice reader, overviewing the major application areas, and mentioning the main technologies, with a focus on standards. Section 14.3 discusses adaptivity in the context of 3D Web sites and with respect to Web-based hypermedia, separating the problems of modeling and adaptation. Section 14.4 describes an example of a full generic architecture for adapting 3D Web content, which is instantiated in Section 14.5 considering a detailed example in the domain of e-commerce. Finally, Section 14.6 concludes the Chapter.

14.2 3D Web Basics

The languages, protocols and software tools that make it possible to build 3D models and 3D VEs that can be experienced through the Web are collectively identified with the term *Web3D technologies*. Nowadays, thanks to the increase in network bandwidth and processing power (especially 3D graphics capabilities), Web3D technologies allow a large number of users worldwide to experience complex 3D Web content, such as virtual cities, visualizations of scientific data, or virtual museums.

Web3D technologies are based on the basic technical and architectural choices typical of Web technologies: content, represented in a proper (and typically textual) format, is stored on a server, requested by a client, typically through HTTP, and displayed by a browser, or, more often, by a plug-in for a Web browser. As a result, 3D content can be strongly integrated with other kinds of Web content, by augmenting Web sites with 3D interactive objects (a 3D model can appear into a Web page together with HTML content) as well as by displaying most types of Web content (such as images, sounds, videos) inside a 3D VE accessible through the Web. This is the main distinctive features of Web3D technologies with respect to other kinds of interactive 3D graphics-related technologies, such as those historically employed in Virtual Reality. Moreover, while Virtual Reality typically focuses on immersive 3D experiences, for example employing head-mounted displays and data gloves, 3D Web content is typically experienced with the input/output devices of today's common personal computers (CRT or LCD monitor, keyboard and mouse).

14.2.1 Applications and Motivations

In the following, we overview the main application domains for 3D Web sites, present possible advantages for using 3D content on the Web, and cite some available systems.

14.2.1.1 Learning and Training

3D VEs offer the possibility to reproduce the real world or to create imaginary worlds, providing experiences that can help people in understanding concepts as well as learning to perform specific tasks in a safe environment. The possibility of delivering educational 3D VEs through the Web allows one to reach potentially large numbers of learners worldwide, at any time (see [18] for a thorough discussion of 3D Web applications in education, learning and training). Employing 3D graphics allows for more realistic representations of subjects or phenomena, offering the possibility of analyzing the same subject from different points of view. Examples in medical education [30] include 3D reconstructions of parts of the human body [47] and 3D simulators [39], like the one shown in Figure 1. Other applications have been developed for foreign language education [40], maintenance training [19, 37], special needs education [31] and optics teaching [50].

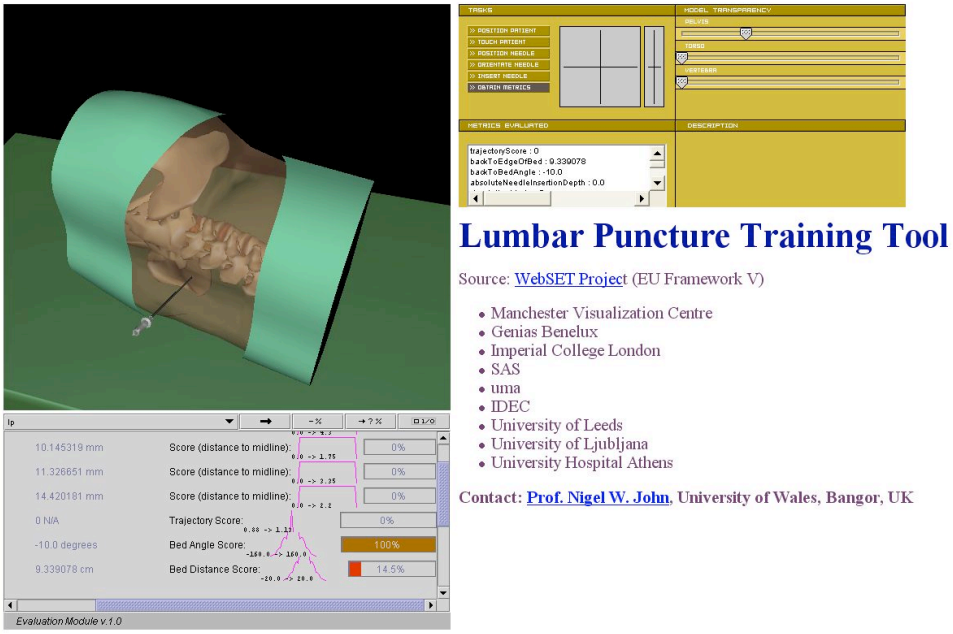

Fig. 1. 3D Web medical training simulator. Image from the WebSET project, reproduced with permission of Nigel W. John.

14.2.1.2 E-commerce and Product Visualization

Although almost all e-commerce Web sites use hypermedia-based interfaces, a few sites have attempted to provide users also with 3D interfaces [1], allowing them to explore a 3D VE representing a store, as in Figure 2. A 3D Web store can have some advantages, if properly implemented:

- it is closer to the real-world shopping experience, and thus more familiar to the customer,
- it supports customer's natural shopping actions (such as walking, looking around the store, picking up products,…),
- it can satisfy emotional needs of customers, by providing a more immersive and visually attractive experience,
- it can satisfy social needs of customers, by allowing them to meet and interact with people (e.g., other customers or salespeople).

On today's e-commerce sites, the simple integration of interactive 3D objects into Web pages, rather than full 3D store environments, is more common, for example in the automotive market [26].

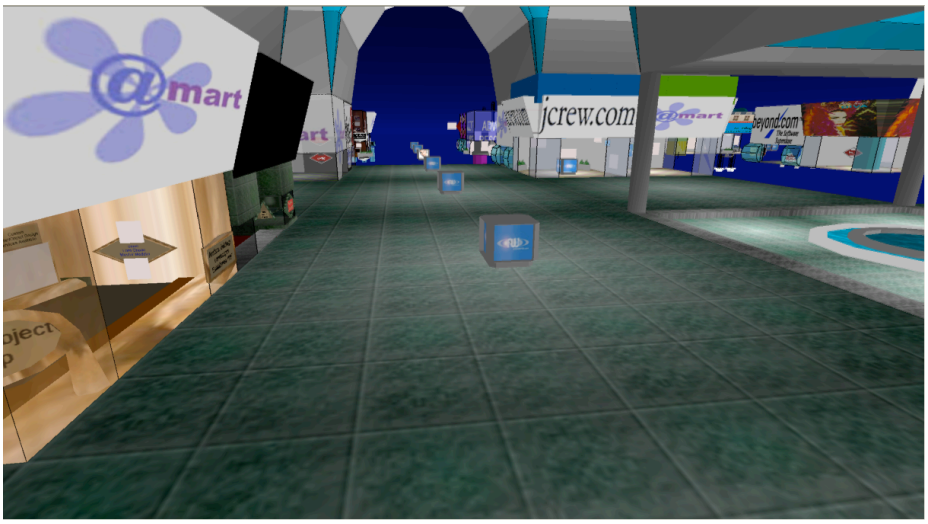

Fig. 2. @Mart 3D Shopping Mall [1].

14.2.1.3 Virtual Museums

Online collections of cultural information are useful if the digital representations of physical items contain enough detail to support the needs of visitors, e.g. researchers. Collections such as photographs or manuscripts can often be effectively acquired and displayed with 2D digital images. However, images are less effective as surrogates for three-dimensional items, such as sculptures, since much spatial information is lost, as the 3D shape of an object has to be flattened onto a two-dimensional view from a single perspective. In these cases, using 3D models can better support the needs of virtual museums visitors.

One can also build 3D VEs that contain representations of cultural objects as well as their contextual environment, e.g. to:

- provide a situated representation of objects;
- virtually reconstruct objects, structures and environments that have been damaged in the past, or do not exist anymore;
- build environments that never existed physically, but represent an appropriate conceptual or architectural environment, such as the virtual reconstruction of Leonardo's ideal city [4] shown in Figure 3.

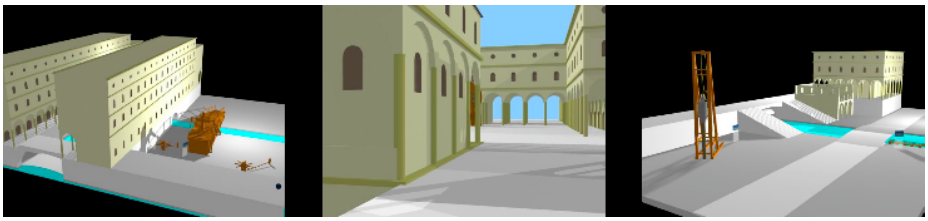

Fig. 3. The virtual reconstruction of Leonardo's ideal city [4]. Image reproduced with permission of Thimoty Barbieri.

14.2.1.4 Architecture and Virtual Cities

Many 3D Web sites allow users to move inside 3D models of buildings and virtual cities [42,44], sometimes providing the capability of seeing each other and chatting. Although most of these sites, such as the one shown in Figure 4, focus on simply reproducing real-world places, there are many possible applications for virtual cities, such as:

- improving the planning, design and management of real cities (e.g., developers looking for sites for new buildings, local authorities managing urban infrastructure),
- providing tourists with detailed guides
- providing community resources for residents.

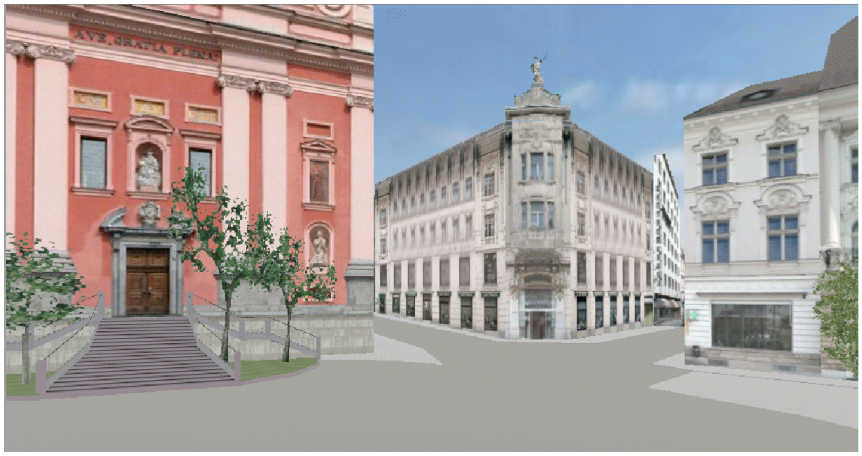

Fig. 4. Virtual Ljubljana [44].

14.2.1.5 Virtual Communities

3D virtual communities on the Web allow a large number of users to build and interact among each other inside a visual 3D space. In the last years, the number of these 3D VEs and their users has grown steadily: for example, Alphaworld [2], one of

the oldest multi-user 3D VEs on the Internet, has hundreds of thousands of users, is roughly as large as the state of California, and contains more than 60 million virtual objects. The main distinctive feature of this kind of 3D Web sites is that users are allowed to build and inhabit visual community spaces, collaboratively engaging in the construction of large scale spaces (including artwork, buildings and full towns) and other social activities, like the virtual ceremony illustrated in Figure 5.

Fig. 5. Social activity in a multi-user 3D VE: wedding in Alphaworld [2].

14.2.2 Available Web3D Technologies

The history of 3D Web sites begins in 1995 with the birth of *VRML* (*Virtual Reality Modeling Language*), which is still the most known and used technology for building and delivering 3D Web content. More specifically, VRML is an open ISO standard [46] for a file format and corresponding run-time behavior to describe interactive 3D objects and 3D VEs delivered through the Web.

Recently, a new ISO standard, called *eXtensible 3D Graphics* (*X3D*) [49], has been proposed as a successor of VRML. Both VRML and X3D are managed by the Web3D Consortium [41], and result from the effort of several organizations, researchers and developers worldwide. Parts of VRML and X3D have been also integrated into the MPEG-4 standard [34], which adopts most of their concepts and instructions to describe interactive multimedia content that includes 3D objects and 3D VEs.

Access to VRML/X3D Web content is possible through one of the available Web browser plug-ins, such as (at the time of writing this Chapter) Parallelgraphics Cortona [37], Bitmanagement Contact [3], Octaga Player [35], and Mediamachines Flux [33].

Besides open ISO standards, there are many other (non-standardized) technologies for 3D on the Web. The best known examples are probably Java3D [29], an extension of the Java language for building 3D applications and applets and Shockwave 3D [32] from Macromedia. Although most of these technologies can be effectively used to build 3D Web content, in this paper we will focus on open Web3D standards. In general, open standards allow for lower costs, easier reusability of content, and easier integration with existing and future content and applications. In the following, we briefly describe the main technical features of VRML and X3D, referring the reader to published books and manuals for complete and detailed explanations.

14.2.2.1 The Virtual Reality Modeling Language (VRML)

The idea of a language for building 3D content for the Web originated back in 1994, when Mark Pesce and Tony Parisi built an early prototype of a 3D browser for the Web, called Labyrinth. Later that year, at the Second International Conference on the World Wide Web, the first specification of VRML was published. In the following years, the language underwent a series of improvements, leading to version 2.0, which was published as an ISO standard in 1997 with the name *VRML97* [46].

VRML is a language that 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.

From a more technical point of view, VRML documents are text files that describe 3D objects and 3D VEs using a *hierarchical scene graph* (i.e., a directed acyclic graph). Entities in the scene graph are called *nodes*. VRML defines 54 different node types, including geometry primitives, appearance properties, sound and video, and nodes for animation and interactivity. For example, hyperlinks are implemented in VRML using the `Anchor` node, through which clicking on a 3D object has the effect of retrieving the resource at a specific URL.

Nodes store their properties in *fields*; the language defines 20 different types of fields that can be used to store different types of data, from single integers to arrays of 3D rotations. It is also possible for the programmer to define new nodes (i.e., extend the language) using a mechanism called *prototyping* through a statement called `Proto`. For example, this mechanism has been used to extend VRML with nodes to represent and animate 3D humanoids [28] and to implement distributed simulations in multi-user, networked 3D VEs [8].

VRML defines a message-passing mechanism that allows nodes in the scene graph to communicate with each other by sending events. This mechanism, together with special types of nodes, called *sensors* and *interpolators*, enables user interaction and animation. For example, the `TimeSensor` node generates temporal events as time passes and is the basis for all animated behaviors. Interpolators nodes are then able to continuously translate temporal events into data needed for animation. For example, the `PositionInterpolator` node is able to translate temporal events into 3D coordinates, allowing one to move objects in space. Other sensors are useful in managing user interaction, by generating events as the user moves through the 3D VE or when the user interacts with some input device (e.g. mouse pointing or clicking).

For example, the `ProximitySensor` node is able to detect the user's position in the 3D VE, while the `TouchSensor` node is able to detect mouse clicks on 3D objects.

More complex behaviors (such as realistic physics simulation) can be implemented by using `Script` nodes, that allow one to manage VRML nodes with programs written in Java or JavaScript.

14.2.2.2 eXtensible 3D (X3D)

The eXtensible 3D (X3D) language for defining interactive 3D Web content was recently released as the successor of VRML, and was approved in 2004 as an ISO standard [49]. X3D inherits most of the design choices and technical features of VRML described in the previous section. As a result, it is mostly backward-compatible, that is, many VRML files require only minimal changes for translation to X3D.

X3D improves upon VRML mainly in three areas. First, it adds new nodes and capabilities, mostly to support advances in 3D graphics techniques and hardware, such as programmable shaders and multi-texturing. Second, it introduces additional data encoding formats. More specifically, it is possible to represent, store and transmit X3D content using a VRML-like textual encoding, an XML-based textual encoding, and a binary encoding, that enables better data compression and thus faster downloads. Third, similarly to XHTML, it divides the language into functional areas called *components*, which can be combined to form different *profiles* (i.e., subsets of the entire language) that are suited to specific classes of applications or devices. For example, this feature would enable one to create a specific profile to take into account the limited capabilities of mobile devices.

14.3 Adaptivity for 3D Web sites

In Web-based hypermedia, which is the mainstream model in today's Web, information is organized and presented into (a graph of connected) pages using various media, with text being the main form of content/medium. Users interact with information mainly by reading, filling forms (e.g., using search engines), and navigating from one page to another by selecting the desired link from those contained in the current page. Many approaches to Web adaptivity presented in this book are targeted towards this model. For example, most techniques for adaptive content presentation discussed in Chapter 7 of this book [9] work with pages and textual content.

The 3D Web model is more complex than Web-based hypermedia, as Table 1 shows. In general, multimedia information, which can include 3D models, images, text, and audio, is organized and presented into a 3D space (or even in multiple 3D spaces connected by hyperlinks), following an arbitrarily complex spatial arrangement, such as a building or an entire city. Users navigate 3D space by controlling the position of their viewpoint through mouse, keyboard, or, more rarely, 3D pointing devices, and sometimes have the ability to teleport from place to place or to other 3D Web sites. As in Web pages, users can exploit hyperlinks to reach other

Web resources. Besides navigation, additional interaction possibilities include the manipulation of 3D objects (e.g., clicking them to perform an action, moving them in space) and even building new objects.

Given these conceptual differences, it is not surprising that the techniques and tools for adaptivity in Web-based hypermedia cannot be straightforwardly applied to personalize 3D Web content, navigation and presentation. As mentioned above, most adaptive hypermedia techniques have been developed for content organized in pages (and not in a 3D space) and mainly made up of text (which is not the prevalent medium in 3D Web sites). With respect to adaptive navigation support, for example, link manipulation as presented in Chapter 8 of this book [5] could accommodate only navigation through hyperlinks in 3D VEs. Moreover, there are also technical differences to be taken into account, namely different file formats. Therefore, alternative techniques, or modifications of existing ones, needs to be developed for adaptivity in 3D Web sites.

In the following, we will describe these techniques, highlighting the main differences with respect to their Web-based hypermedia counterparts. To make practical comparisons, we will use *AHA!* [22] (also discussed in Chapter 1 [7] and Chapter 13 [9] of this book) as a representative example of Web-based hypermedia adaptive systems. First, we will discuss how to build and update the user model, i.e., the *modeling* task, and then how to deliver personalized 3D content, i.e., the *adaptation* task.

Table 1. Analogies and differences between Web-based hypermedia and 3D Web sites

	Web-based hypermedia	3D Web sites
presentation container	page	3D space
content media	mainly text, but also images, videos, ...	mainly 3D models, but also text, images, videos, ...
structural organization	graph of pages	3D space or graph of 3D spaces
navigation	through hyperlinks	by moving in 3D space (e.g., walking, flying) and teleporting; also through hyperlinks
other common users' activities	reading pages, filling forms	3D object manipulation (clicking, moving, …),

14.3.1 Modeling

The approaches to adaptive 3D Web content developed so far have reused standard user model representation and reasoning techniques, such as stereotypes, graphs of concepts, and inference rules. Those techniques indeed are not specific to the hypermedia model. However, the task of user model acquisition (building and updating the model) requires a different approach in 3D Web sites.

With adaptive Web-based hypermedia, user model updates are typically triggered each time the browser requests a page. For example, in *AHA!* the adaptation engine starts by executing the rules associated with the attribute *access* of the requested page. Then, the user model is updated assuming that the requested page will be read, for example increasing the user's knowledge level about the concepts described in the page. This technique is effective under the assumption that *the user will fully read the page*, or, in other words, that all content accessed from the server will be read by the user. This is a strong assumption, since the user might skip parts of the page and thus cause inappropriate updates to the user model, but there are no easy methods to track which parts of a page have been actually read. Although there are available techniques for this purpose, such as eye tracking, they are costly or unpractical to adopt for Web sites and their visitors, except in special situations such as marketing research.

With 3D Web sites, assuming that all content accessed from the server is going to be seen or properly employed by the user is even more likely to cause erroneous user model updates. In many cases, users see only a part of the downloaded content (3D models, images, …) that constitutes the 3D VE, for example because exploring a large or complex environment can require hours. Even in a smaller 3D VE, users might not see some objects because they are occluded by other objects (from the user's path during the visit) or simply do not notice them while navigating. Moreover, when some object manipulation is possible, users might not perform it or 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 proposed [16, 17] consists in closely monitoring users' behavior in the 3D VE, and send relevant time-stamped users' actions (e.g., movements, objects clicked) to the server, where they can trigger user model updates. In this way, we can update the user model not when content is accessed from the server, but only when we are confident the user has actually seen it or interacted with it. For example, by recording user's position in the 3D VE every few seconds and sending it to the server, it is possible to know which parts of the environment were actually visited and update the user model accordingly.

This approach does not require much implementation effort or special hardware because most Web3D technologies include mechanisms (called *sensors* in VRML and X3D, see Section 14.2.2) to monitor low-level events, such as mouse movements, as they are necessary for interactivity. Relevant interaction data gathered through sensors can be collected and sent to the server through programs (e.g., VRML Script nodes). For example, such technique has been used:

- to monitor user's position in 3D space, and determine which parts of the 3D VE have been actually visited,
- to check whether the virtual head of the user is oriented towards a certain 3D object, and determine whether the object might have been actually seen by the user (e.g., considering distance),
- to check whether and how a certain 3D object has been clicked or dragged by the user, and determine whether a certain action has been properly performed.

A more detailed technical explanation of the proposed solution, in the case of VRML-based 3D Web sites, is presented in Sections 14.4.1 and 14.5.2.

14.3.2 Adaptation

In this section, we discuss techniques for adaptive navigation support and adaptive presentation of content in 3D Web sites. A general issue concerns how frequently adaptation can and should be made. With adaptive Web-based hypermedia, adaptation is normally performed on each requested page, although it might be desirable, for some content, to reduce the frequency of the adaptation process, for example once per session [22]. However, since users typically read one page at a time, adapting each requested page enables them to see the effects of adaptation during a browsing session and at the right time.

So far, the approaches to adaptive 3D Web sites have adopted a similar solution, i.e., adaptation is performed when 3D content is requested from the server [17,21]. However, in the typical situation where the full 3D VE is downloaded at the beginning of the user's visit, with this solution only adaptations between visits are possible. For example, an adaptive 3D virtual store where all content (store building, 3D models of products, advertisement banners) is downloaded at the beginning of the user's visit does not allow the user to see adaptations taking into account which products have been more examined since the beginning of the visit.

With most Web3D technologies, one can however download or update parts of the 3D VE during the user's visit. For example, both VRML and X3D provide this possibility, but developers are required to write ad-hoc scripts. Alternatively, there are extensions to VRML, such as *X-VRML* [48], that provide easier mechanisms to implement updates or downloads of content during visits, and thus carry out adaptations during visits. A simple but effective example of this strategy has been used in a 3D virtual museum [13]. The museum features a virtual human acting as a guide, leading the user around and describing museum items using speech synthesis. Each time an item needs to be presented, the text to be spoken is requested to the server, where it is tailored according to the user model, and then downloaded and fed to the speech synthesizer.

In general, which kinds of adaptations are best suited during visits, and their optimal frequency, are open issues. Typically, user's experience of 3D VEs should be as continuous as possible to maintain user engagement, while in Web-based hypermedia adaptive changes among pages are not (or much less) perceived as annoying breakdowns since the experience is already 'divided into pages'. For example, modifying the position, appearance or behavior of visible objects while the user is visiting the 3D VE, even if the user model would suggest to do so, should be

carefully performed, otherwise it will likely turn out as annoying or counter-productive for the user's experience. In the following, we first discuss how to adaptively support navigation and interaction, and then how to adaptively present 3D content. Finally, in Section 14.3.2.3, we consider adaptivity in the context of multi-user 3D VEs.

14.3.2.1 Adaptive Navigation and Interaction Support

Although Web-based hypermedia and 3D VEs are different, they are both targeted for user-driven navigation and exploration [27]. Like in the case of Web-based hypermedia, it seems thus interesting to develop adaptive navigation and interaction support techniques that can help users in finding and using information more efficiently, and prevent navigation and interaction problems. Moreover, navigation is a very relevant usability issue in the context of 3D VEs. In current 3D VEs, people often become disoriented and tend to get lost, and these problems are exacerbated by difficulties such as controlling movements in a 3D space, and limited field of view compared to the real-world experience. Inadequate navigation support is likely to result in users taking wrong directions, leaving the 3D VE before reaching their targets of interest, or with the feeling of not having adequately explored the visited 3D VE. These problems become even more critical in the case of novice users, who might become easily frustrated in learning how to navigate.

Although many techniques (called *electronic navigation aids*), such as electronic 2D and 3D maps, have been developed to help users in navigating 3D VEs, they are not able to adapt to users with different navigation and interaction abilities. For this reason, some researchers [6] have proposed to develop adaptive navigation support techniques, mostly by deriving them from established methods in adaptive Hypermedia.

Fig. 6. Annotation by means of flashlight (left) and arrows (right) [27]. Image courtesy Stephen Hughes.

Hughes et al. [27] propose a number of adaptive navigation support techniques based on computing *ideal viewpoints* in the 3D VE on the basis of the user model, and then use them to prevent erroneous directions, disorientation or missed parts. The ideal viewpoints correspond to locations in the 3D VE (more specifically, positions and corresponding orientations in 3D space) from which objects or parts of the 3D VE that are interesting for the user are well visible. The idea is to constrain navigation or draw additional information to help the user in reaching the ideal viewpoints. The proposed techniques are derived from the link manipulation techniques discussed in Chapter 8 of this book [5]:

- *direct guidance* (a strict linear order through the navigation space) computes a path through the 3D VE that encompasses all ideal viewpoints, and then automatically moves the user's viewpoint along this path;
- *hiding* (restricting the number of navigation options to a limited subset) hides all irrelevant orientations by letting the user move her position freely, but having the system dictate the orientation of the users' virtual head to force it to fixate on certain objects while moving;
- *sorting* (altering the order in which navigation decision are presented to the user) orders ideal viewpoints and let the user move freely, but, as with hiding, the system dictates the orientation of the users' virtual head to force it to fixate on certain objects in the computed order. In this case, the user still has the possibility to override system decisions and orient the virtual head to explore other objects;
- *annotation* (displaying additional information on navigation options) displays attention-drawing signs, such as the arrows in the right part of Figure 6, to indicate interesting objects, or highlights them using a flashlight while unimportant features are left in the dark as in the left part of Figure 6.

An alternative approach [12,16] to implement sorting and annotation-like adaptive support exploits virtual characters, such as the ones in Figure 7 and Figure 14, that act as navigation guides to:

- show users the path to an object, or the path through a sorted list of objects of interest, i.e. implementing sorting-like navigation support;
- provide annotations in the form of additional information on navigation and interaction possibilities; for example, the virtual character in Figure 7 is showing a new user that an object can be opened to see its interior.

This style of adaptive support has been employed in two different contexts. In a 3D virtual museum [12], the virtual character acts as the museum guide, leading the user around, giving information on museum items and showing possible interactions. The first time the user visits the museum, a sequence of museum items (i.e., a museum tour) is generated on the basis of the user profile, and the virtual character guides the user through them. In successive visits, only those items that have not been seen are included in the tour (this has similarities with the hiding technique explained above). In a 3D virtual store [14,16], multiple animated characters are employed to guide the user to different products (this technique is described in more detail in Section 14.5). The animated characters look like products (see Figure 14), and their actual appearance (i.e., the specific product they represent) is adapted to take into account user's potential buying interests.

While using 3D virtual characters does not directly help the user in controlling navigation as direct guidance, hiding, and sorting, it has the following distinctive features:

- it can draw the user's attention with natural and familiar methods. For example, the humanoid character in Figure 7 uses gaze, pointing gestures, body orientation, and provides textual information through voice;
- it may have an emotional impact on the user, and increase motivation and engagement: users tend to experience presentations given by animated characters as lively and engaging [43]. Moreover, it can make the virtual place more lively, attractive, and less intimidating to the user;
- it does not restrict the navigation possibilities, since the user can choose whether to employ adaptive support or not by not following the virtual character and explore the 3D VE on its own.

Fig. 7. A humanoid character shows the user how an object can be opened [12]

Another kind of adaptive navigation and interaction support has been proposed by Celentano and Pittarello [11]. Their idea is to monitor user's behavior and to exploit the acquired knowledge for anticipating user's needs in forthcoming interactions. More specifically, the approach is based on using sensors (as described in Section 14.3.1) to collect usage data, and compare them with previous patterns of interaction stored in the user profile. The patterns of interaction are sequences of activities which the user performs in some specific situation during the interactive execution of a task,

and are encoded as Finite State Machines (FSM). 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. For example, figure 8 shows an example of interaction adaptation in a virtual fair application. The FSM on the top of the Figure shows the sequence of actions that must be performed to interact with an object inside a showcase. The FSM on the bottom of the Figure is computed by the interaction support system after the first FSM has been detected as recurring. In the FSM in Figure 8, the dotted arrow represents an automatic execution of actions performed by the system. More specifically, if the user is closer than 3 meters from the showcase, the open button, even if it is not visible, is automatically pressed to open the showcase on behalf of the user.

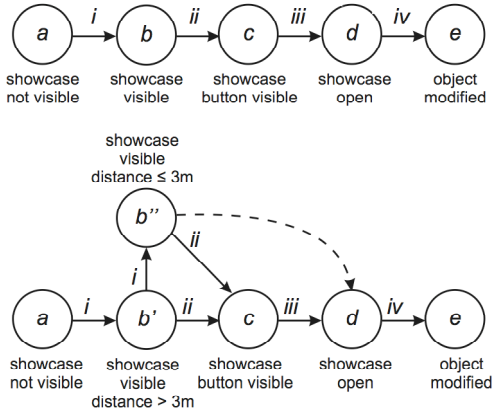

Fig. 8. Interaction adaptation in a virtual fair application [11]. Image courtesy Fabio Pittarello.

14.3.2.2 Adaptive Presentation of Content

Adaptive presentation of content concerns deciding what content is most relevant to the user, how to structure it in a coherent way, and how to present it in the best way. For the first two tasks, the most widely used techniques in Web-based hypermedia are *optional fragments* and *altering fragments*. As mentioned in Chapter 7 of this book [9], those techniques build adaptive pages by selecting and combining an appropriate set of fragments, where each fragment typically corresponds to a self-contained information element, such as text paragraphs or pictures.

The techniques for adaptive presentation of 3D content developed so far follow the same fragment-based approach, and can therefore be thought as variations of the above mentioned adaptation approaches.

The approach proposed in [17] uses the VRML `PROTO` construct to define each kind of self-contained adaptive fragment. In general, `PROTO` defines a new VRML node by specifying its *interface*, i.e. fields and events the node receives and sends, and its *body*, i.e. how the node is implemented in terms of existing or previously defined VRML nodes. As with any other VRML node, each time the new node is inserted, or *instantiated*, in the 3D VE, one can change the values of the fields

declared in the interface to customize the features of the node. For example, the following code defines a very simple node for a box-shaped product in a 3D store, where the size of the box and the image printed on its sides are encoded as fields:

```
PROTO BoxProduct
[ field SFVec3f bsize 0 0 0    // size of the box in x, y, z
  field MFString imageURL [] // url of image that will appear on the box
]
{
   Shape {                      // node to define a 3D object
   appearance Appearance {    // appearance of the 3D object
          texture ImageTexture {
             url IS imageURL }   // applies the image to the box
   }
   geometry Box {      // the geometry of the 3D object is defined by a box
          size is bsize } // size of the box
  }
}
```

The idea is that fields in the interface define the adaptive features of the node, abstracting from other non-adaptive details. In the product example, therefore, the adaptive features are the size of the box, and the image printed on its sides. With this approach, 3D adaptive content is defined by a set of `BoxProduct` node instantiations, such as in the following code fragment, which includes a milk box in a 3D VE:

```
BoxProduct {
    bsize 1 2 1
    imageURL "milkBox.jpg"
}
```

The idea is that field values (such as "`milkBox.jpg`") are chosen among a set of alternatives (that have to be stored separately) or computed by the adaptive engine when content is requested.

The alternative technique proposed in [15] for the X3D language does not uses a prototyping mechanism (which is available also with X3D), but requires an additional file, called *Content Personalization Specification* (*CPS*), for each X3D document with adaptive content. The CPS file defines adaptive features and may also specify possible variants. With this technique, the milk box example above would be implemented by the following X3D code fragment:

```
<Shape>
  <Box DEF="size1" />
  <Appearance>
    <ImageTexture DEF="imgUrl1" />
  </Appearance>
</Shape>
```

and a separate CPS file specifying that the size of the box and the image on its side are adaptive features. The following CPS does that, also defining two possible actual adaptations for the product image:

```
<CPS>
   <adaptiveContent DEF="imgUrl1" attribute="url">
      <value>"milkBox.jpg"</value>
      <value>"cerealBox.jpg"</value>
   <adaptiveContent DEF="size1" attribute="size"/>
</CPS>
```

One of the advantages of using XML-encoded content (such as X3D) is the possibility of using adaptation techniques developed for other kinds of XML-based content. For example, the approach proposed by Dachselt et al. [21] uses the Amacont general-purpose architecture [25] with X3D content or more high-level formats [20]. For example, the fact that the image printed on the sides of the box-shaped product is an adaptive parameter would be expressed in the approach of Dachselt et al. by the following code fragment, which, contrary to the techniques above, includes also the logic of adaptation:

```
<Parameter name="url" dataType="CoAnyURI" ... >
 <Variants>
  <Logic>
   <If>
    <Expr>
     <Term type="=">
      <UserParam>Favorite Product</UserParam>
      <Const>Milk</Const>
     </Term>
    </Expr>
    <Then>
      <ChooseVariant>milk</ChooseVariant>
    </Then>
    <Else>
     <ChooseVariant>cereals</ChooseVariant>
    </Else>
   </If>
  </Logic>
  <Variant name="milk">
    <CoAnyURI>"milkBox.jpg"</CoAnyURIs>
  <Variant>
  <Variant name="cereals">
   <CoAnyURIs>"cerealsBox.jpg"</CoAnyURIs>
  </Variant>
 </Variants>
</Parameter>
```

The `Parameter` element encodes an adaptive feature (in this case, an image depicting a product). The enclosed `Variants` element define possible variants for the feature. Inside the `Variants` element, a `Logic` element defines the logic of adaptation (if the user's favorite product is milk, we will use the `milk` variant, else we will use the `cereals` variant. Then, a list of `Variant` elements defines the possible variants as URLs of the images.

While these approaches provide fragment-based techniques to perform adaptation of content, using them is not as easy as in Web-based hypermedia. Text fragments or images can be simply juxtaposed in a page, with the only possible drawback of not preserving 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 fragments have been chosen, one needs to properly arrange them in 3D space and time (if there are animations) such as, for example, included objects do not intersect each other, are adequately visible from the positions the user will take in space, and free space is enough for the user to move. Unfortunately, it is very difficult to develop general algorithms for this purpose. This forces one to limit the space of possible adaptations to a few variants that are guaranteed to be safe with respect to the above mentioned constraints, or to implement adaptation strategies that might work only in a specific 3D VE.

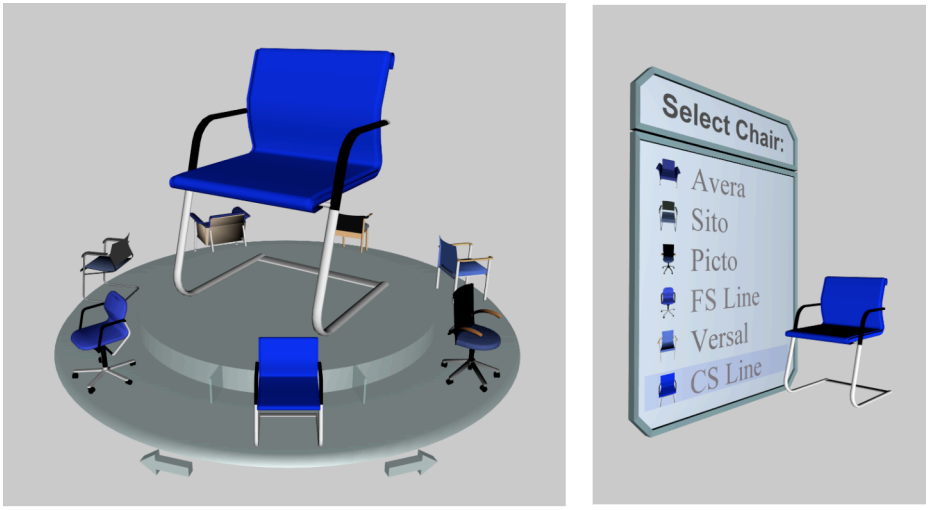

Fig. 9. On the left, a ring menu for choosing a chair; on the right, the same menu adapted for smaller displays, such as PDAs [21] Image courtesy Raimund Dachselt.

Even if one could easily implement any kind of adaptation, there are presently no studies that investigate the effect on users of content adaptation in 3D VEs. Therefore, we can only try to hypothesize which adaptations might be useful and which might be counterproductive. For example, it is likely that changes in the navigational structure of a 3D 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. In the following, we mention some examples of adaptations of 3D content that have been proposed in the literature.

In the adaptive 3D e-commerce example we will discuss in Section 14.5, the number of instances of products in shelves can vary in a given range (one to four) to adaptively increase or decrease the visibility of the product itself (see Figure 15). The limited number of variants guarantees that each product will not take the space reserved to other products.

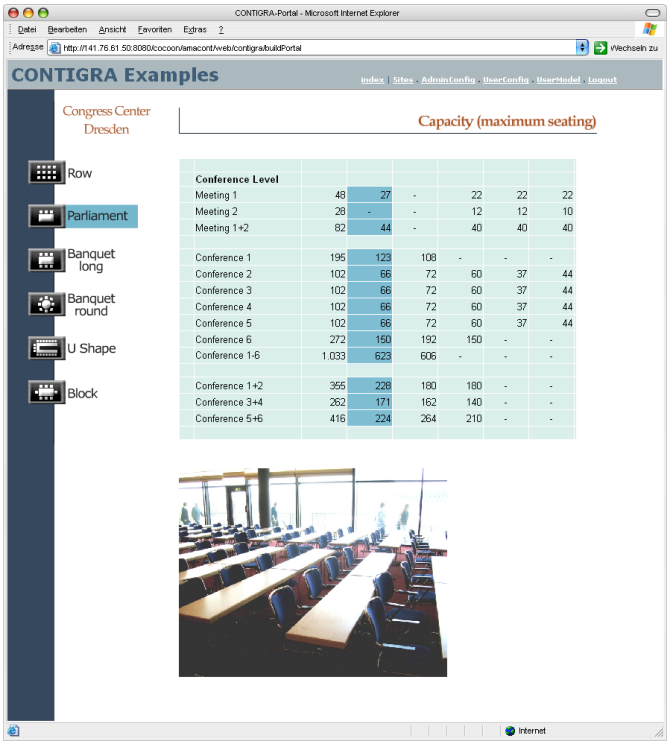

Fig. 10. Web site for adjusting the seating capacity of conference rooms Image courtesy Raimund Dachselt.

A 3D adaptive e-learning system [24] organizes learning content into a building made of rooms, and the adaptation engine places rooms (by just exchanging content among equally-sized rooms) that correspond to the areas of higher user's interest before rooms whose contents are less interesting for the user.

The 3D menus shown in Figure 9 [21] are examples of adaptation to the user's device. The idea is to provide different alternatives, with respect to screen space usage, for the same 3D interface element and information presented. In particular, the screenshot on the right shows a smaller-sized version of the ring menu on the left, and is better suited to small displays, such as PDAs.

Finally, 3D content could also be considered in media adaptation. Figures 10 and 11 shows two different versions of the same Web site, whose purpose is to adjust the seating capacity of conference rooms [21]. Figure 10 shows an HTML-based version,

which might be more suited to low-bandwidth connections or users that are not familiar with 3D. Figure 11 shows a 3D-based version, where the conference room is represented by a 3D VE to better visualize the final result.

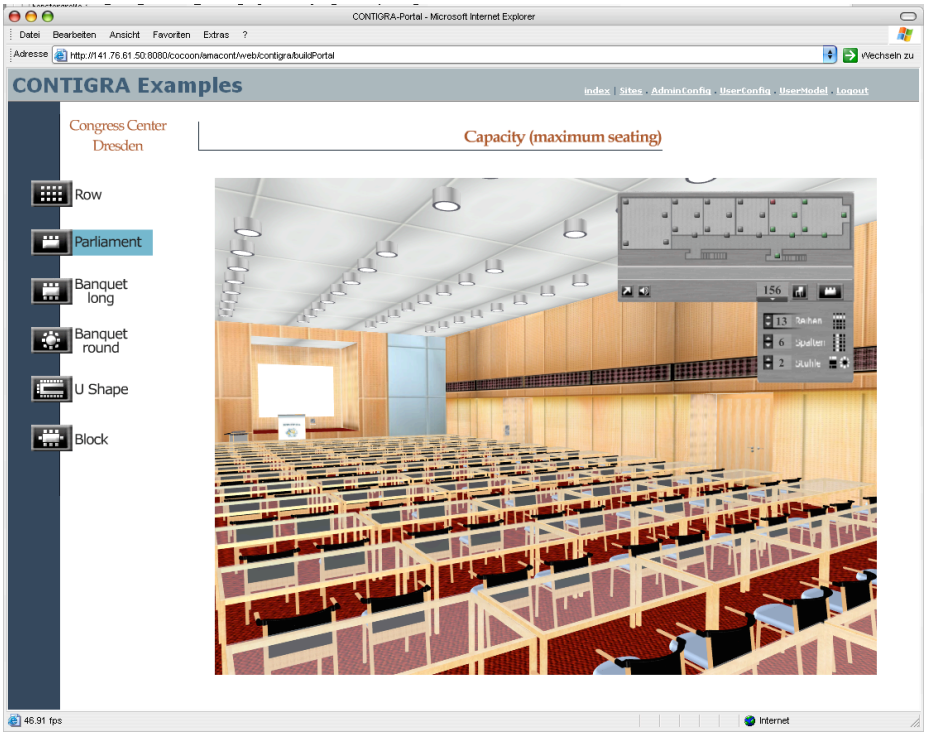

Fig. 11. Adapted version of the content in Figure 10, using a VRML 3D VE Image courtesy Raimund Dachselt.

14.3.2.3 Multi-User 3D VEs

No examples of adaptivity in multi-user 3D VEs have been reported in the literature. This might be due to the fact that multi-user 3D VEs can conflict with personalization aspects, making some of the adaptations presented in the previous section troublesome. In general, if multiple users navigate and interact together in the same 3D VE, adaptation of content cannot safely target the specific profile of a single user. For example, adaptations that cause one person to see the 3D VE differently from others could cause deep misunderstandings (e.g., a reference to a highlighted object that is not highlighted for another person) that may hinder social activities.

There are however strategies that could be pursued to prevent this kind of problems. For adaptations that conflict with multi-user activities, one could try to find the best common adaptation which maximizes the match with the different user models. However, considering that the set of users could continuously change, this might not be easy to implement. A second possibility could be to clearly mark what is personalized in the 3D VE, and see if users are able to adopt new conventions.

Another possibility would be to find useful adaptations that do not conflict with multi-user activities, or even result from them. For example, an idea that has been developed for adaptive multi-user textual environments is to change the description of objects in the environment to reflect usage [23], such as doors or books showing signs of wear. A similar idea could be used in a multi-user 3D VE to visually represent frequently accessed paths or objects.

14.4 A Generic Software Architecture for Adaptive 3D Web sites

A few software architectures for adaptive 3D Web sites can be found in the literature. The AWe3D (Adaptive Web 3D) architecture [17] is a general purpose architecture for generating and delivering adaptive VRML content which was proposed in 2002. More recently, a few researchers [15,21] have focused on integrating 3D content into existing technologies, such as the Amacont general-purpose architecture [25], for Web adaptivity.

In the following, we describe a generic architecture (depicted in Figure 12) that generalizes the ideas of AWe3D, for delivering adaptive content in 3D Web sites. The architecture is composed by the following modules:

- a *Usage Data Sensing* module, whose purpose is to monitor user's interaction with the 3D VE, and send the relevant events through the Internet. This module is located on the client side, run by the user's browser;
- a *Usage Data Recorder* module, whose purpose is to receive, on the server side, the events sent by the Usage Data Sensing module, and record them in the *User Model Database*;
- an *Adaptivity Engine*, that: (i) performs inferences needed to update the user model on the basis of recorded usage data, and (ii) given the current user model, computes a set of adaptation choices for requested adaptive content;
- a *3D Content Creator* module, that: (i) accepts content requests from the client; (ii) when adaptive content is requested, asks the Adaptivity Engine to provide the correct adaptation choices, and uses them to build the adapted 3D content, retrieving needed files (3D models, images, sounds,…) from the *3D Content Database*; (iii) delivers the requested 3D content to the client.

We now describe in more detail a possible set of technical choices to implement each module in the case of VRML-based 3D Web sites.

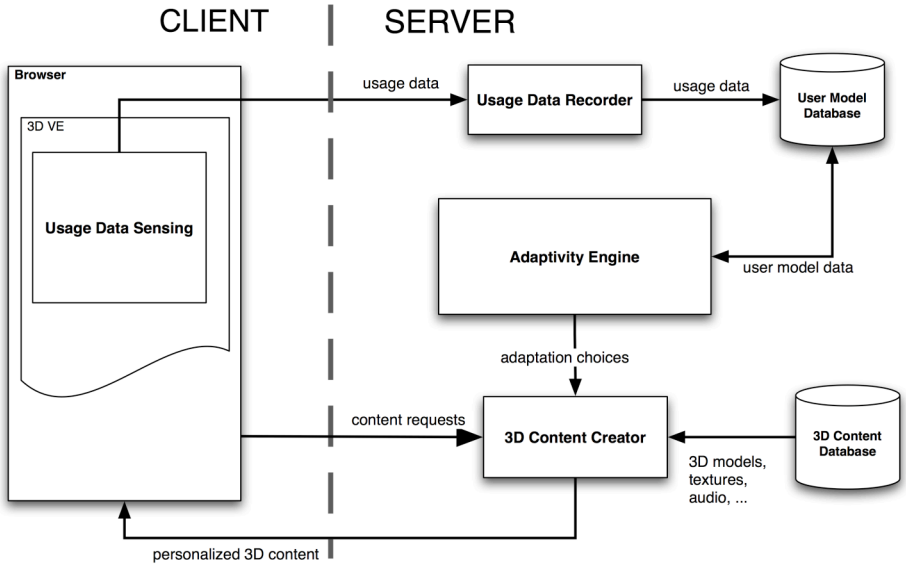

Fig. 12. Schema of a general architecture for adaptive 3D Web sites

14.4.1 Usage Data Sensing

Following the technique outlined in Section 14.3.1, this module is implemented by a set of VRML sensors whose output is routed to a Script node, which transmits relevant usage events to the Usage Data Recorder module by using a HTTP connection.

The type, number and specific settings of VRML sensors in this module 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. VRML sensors allow one to track the user's position, or user's collisions with an object, or mouse actions on an object, or visibility of an object. By combining the output of multiple sensors in a Script node, 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, and a Script node that receives the sensors' output, recognizes the correct sequence, and send the resulting high-level event to the server.

14.4.2 Usage Data Recorder

The Usage Data Recorder is implemented by a simple server-side program that receives usage data and stores them with a DBMS. A more elaborate version could also perform calculations on the usage data before storing, for example filtering, averaging, sums, …

14.4.3 3D Content Creator Module

The *3D Content Creator* receives requests for 3D content, and returns that content to the client. Adaptive fragments are represented through VRML `PROTO` constructs (using the technique illustrated in Section 14.3.2.2), whose fields encode the adaptive features, such as object geometry, color, and size. The 3D Content Creator Module asks the Adaptivity Engine to compute actual values for each `PROTO` field (i.e., a set of *adaptation choices*), and use the result to instantiate the `PROTO` in the file that is returned to the client, possibly retrieving needed code (such as 3D models, animations, and images) from the 3D Content Database.

14.4.4 Adaptivity Engine

The technical choices that have to be taken in implementing the *Adaptivity Engine* depend on how complex are the inferences that have to be performed. A simple solution, using a rule-based approach, is to write a set of *User Model Update* rules to update the user model on the basis of collected usage data, and a set of *Content Adaptation* rules to compute personalized field values for adaptive content. The User Model Update rules can be activated each time usage data are received from the client, or periodically, at given intervals of time or after a certain number of user's visits to the Web site. The Content Adaptation Rules are activated each time the 3D Content creator asks for personalized versions of adaptive fragments.

14.5 An Application in E-Commerce

In the following, we describe a detailed example in the domain of e-commerce implemented using the architecture introduced in the previous section. We first describe the considered 3D store, then we discuss specific technical choices to implement an adaptive version of it. The example we propose is a simplified version of the 3D adaptive store presented in [17], to which we refer the reader for more detailed technical specifications and code examples.

14.5.1 A 3D Store VE

The 3D VE we consider is composed by a 3D model of a department store, displaying products on several shelves. The customer can wander through the store, obtain information on products by clicking on them, put them in the cart, which is also represented in 3D (see Figure 13), and go to the checkout counter to conclude her shopping session. Besides shelves, customers' attention towards products is sought by exploiting special rotating display spots in prominent places, advertisements on the walls, and audio messages. Moreover, the store is populated by *Walking Products* (*WPs*, see Figure 14), a navigation support feature to help users in finding products [14]. WPs are 3D animated representations of products that move through the store and walk to the place where the corresponding type of products is. A customer in the

3D store sees a number of WPs wandering around: if she is looking for a specific type of products, she has just to follow any WP of that type and will be quickly and easily lead to the desired destination.

Fig. 13. A 3D store with products on shelves

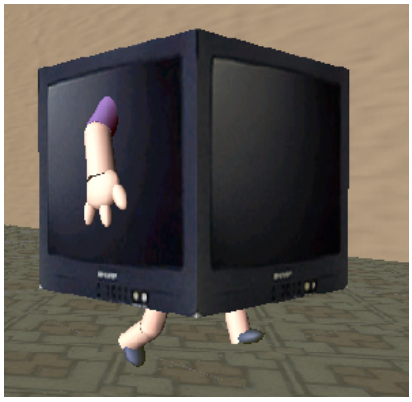

Fig. 14. Example of a Walking Product

14.5.2 Usage Data Sensing and Recording

Usage data we are interested in concern typical interactions with products in the store. More specifically, the data collected by the *Usage Data Sensing* module by monitoring customer's actions are:

- *Seen Products*. While the customer wanders around the store, she voluntarily or involuntarily looks at the products which fall in her field of view;
- *Clicked Products*. When the customer wants to know more about a product, she clicks on it to get the product description;

- *Cart Products*. The product description allows the customer to put the product in the shopping cart for a possible later purchase;
- *Purchased Products*. A product in the cart can be later purchased by going to the checkout counter.

Seen Products and *Clicked Products* data are acquired through Visibility and Touch sensors associated to each product. The following is a slightly simplified `PROTO` defining a product node with the required sensing capabilities:

```
PROTO Product
[  field MFNode product3DModel []
   eventOut SFTime productSeen
   eventOut SFTime productClicked  ]
{
   Group { children IS product3DModel }

   TouchSensor  {
                  touchTime IS productClicked }

   VisibilitySensor {
                   enterTime IS productSeen }
}
```

The interface of the node `Product` (the first three lines after the `PROTO` statement) include a field for the 3D model of the product (`product3DModel`), and two events that can be sent to other nodes, respectively indicating when the product is seen (`productSeen`) and clicked (`productClicked`). The `Product3DModel` field is an adaptive feature of the product: its 3D model can be chosen among different alternatives, for example to occupy less or more space in shelves, as shown in Figure 15. The body of the node `Product` (the code between braces) includes a reference to the 3D Model of the product, a Touch Sensor to detect click events, and a Visibility Sensor to detect when the product is visible.

In similar ways, *Cart Products* and *Purchased Products* data is acquired in the VRML code describing the cart and the checkout counter, respectively.

14.5.3 User Model

Customer models in the User Model Database of the 3D store contain the following information:

- *demographic data*, including gender, year of birth, and product categories of interest among those available in the store, which the customer can enter through an HTML form the first time she enters the store;
- *user preferences about the store*, such as presence of audio and music, and preferred music genre, which are also entered or modified by the user through the HTML form;
- *usage data*, described in the previous section, and exploited to dynamically update the user model. Usage data allows one to obtain a precise quantitative

measurement of which brands, product categories, specific products, price categories, and special offers have been respectively seen, clicked, put in the shopping cart or purchased by the customer;
- *Product Interest Ranking*, which ranks products and products categories according to customer's interests.

To determine the Product Interest Ranking, an initial value is determined by using a HTML form that allows the customer to indicate her products of interests: if she chooses to fill it, the information is used to initialize the ranking. If the customer does not provide product interests in the HTML form, one can try to predict interests by using demographic profiles. Then, regardless of the quality of the initial value, product interests will be continuously updated by the Adaptivity Engine which exploits usage data: each purchase, cart insertion, and click of a product increases (with different weights) the level of interest in the corresponding product and product category or in related products.

14.5.4 3D Store Adaptivity

The adaptive features of the 3D store mainly concern where and how products are displayed:
- each product is displayed in the shelf assigned to its product category, but the amount of shelf space devoted to the product is adaptively changed to increase or decrease product visibility;
- additionally, a product may appear also in display spots, banners, or WPs to increase its exposure towards the user.

Other adaptive features concern the music that is played, and the audio messages that advertise products.

In the following, we present some examples of rules that perform adaptations in the 3D store. Simple rules are given by the direct associations between user's preferences about presence of music and preferred genres, and songs that are played during visit. More complex examples concern the exploitation of the user model to change the level of product exposure in the 3D store. The level of exposure of each product can vary the product visibility and attractiveness, for example by increasing space devoted to the product in the store or adding banners advertising the product. We call *ExposureLevel(X)* the parameter which represents the level of exposure for product *X*. The value of *ExposureLevel(X)* is determined by five more specific parameters:
- *ShelfSpace(X)* indicates the space assigned to product *X* on the shelf. It can take four different values: higher values make *X* more visible to the customer, increasing *ExposureLevel(X)*. The products in Figure 15 show two different allocations of shelf space;
- *DisplaySpot(X)* is false if product *X* is displayed only on its shelf, while it is true if product *X* is displayed also in a separate display spot in a prominent place (we could have also used numerical values to allow the same product to be displayed on more than one display spot);
- *Banner(X)* is true if there is a banner advertising product *X* in the store;
- *AudioMessage(X)* is true if audio advertisements for product *X* are played;
- *WP(X)* is true if there is a WP representing product *X* in the store.

A true value for any of the last four boolean parameters increases *ExposureLevel(X)*. Personalization rules first suggest changes to exposure level by asserting increase or decrease goals for specific products. Then, they focus on achieving those goals, by changing one or more of the above described parameters, according to the availability of store resources (e.g., if a shelf is full, shelf space for products cannot be increased on that shelf).

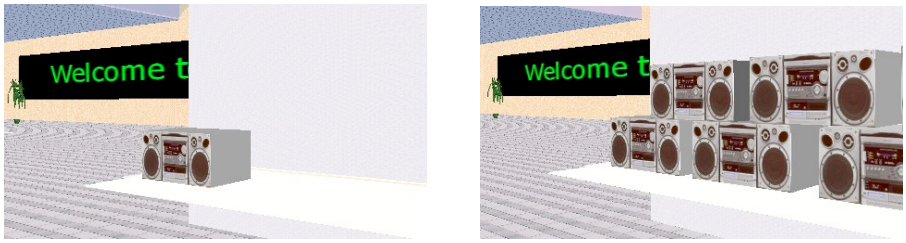

Fig. 15. Two different possible allocations of shelf space for the same product

We now examine some specific rules and how they relate to the information recorded in the user model. Suppose that a product *X* has never been seen by the customer or that changes in the Product Interest Ranking show an increasing attention towards the product. In both cases, a seller would like to increase the exposure of the product (in the first case, to give the customer the opportunity of seeing the product; in the second case, to better match customer interests). The rules that implement the two cases can be expressed as follows, where `seen(X)` is the recorded number of times a product has been seen, `ProductInterest(X)` is the rank in the product interest ranking, and `NumberOfVisits` is the number of times the user has visited the store:

```
IF seen(X)=0 AND NumberOfVisits>3 THEN
 goal(IncreaseExposureLevel(X))

IF increasing(ProductInterest(X)) THEN
 goal(IncreaseExposureLevel(X))
```

As another example, consider the cross-sell case where the purchase of a specific product *X* is an indicator of a likely future interest for related products and we want to update the user model accordingly. For example, if a customer buys a computer and has never purchased a printer, she could be soon interested in a printer. The rule can be expressed as follows, where `purchased(X)` is the recorded number of times a product has been purchased, `lastVisit` extracts the value of data considering only the last visit to the store, and `RelatedProduct(X,Y)` relates products by using associations provided by the seller:

```
IF lastVisit(purchased(X))>0 AND RelatedProduct(X,Y)
AND purchased(Y)=0 THEN increase(ProductInterest(Y))
```

As an effect of the increased product interest, the second rule examined above will then suggest an increase in the exposure level of related products which have not been purchased yet. Note that the `RelatedProduct` relation cannot be used transitively, because this could lead to counterproductive merchandising strategies. For example, an ink cartridge is obviously related to a printer, and a printer is obviously related to a computer, but it does not make sense to increase the exposure level of ink cartridges if a customer has purchased a computer but not a printer.

Finally, to prevent an excessive number of changes to the 3D store from one session to another, we impose a limit on their number for any given session. The idea is to keep the experience of returning to the 3D store consistent with the familiar experience of returning to a known real-world store: the store layout remains essentially the same, and a limited number of changes concern what products are displayed, and how the attention of the customer towards those products is sought.

14.6 Conclusions

Adaptivity of 3D content for the Web is a very recent and largely unexplored research topic. As shown in Section 14.3, there are only a few examples of adaptation of 3D content in the literature, and no thorough evaluations with users have been carried out. To understand the true potential of adaptivity of 3D content, we need to explore in more depth the space of possible adaptations, including less obvious ones. For example, most 3D VEs (including those built with VRML or X3D) allow the use of spatial audio. An interesting possibility could be to use adaptive spatial audio to provide information to the user, for example navigation support.

It is also important to investigate users' reactions to adaptive changes in 3D content. As discussed in the Chapter, adaptivity may break or hinder important features of a user's experience in a 3D VE, such as the construction of spatial knowledge, and the continuity of the experience. Studies on users are therefore needed to establish when and how it is useful to adaptively change a 3D VE.

We hope that this Chapter has provided an easy-to-read introduction for students, as well as a stimulating starting point for researchers that aim at advancing this line of research.

14.7 References

1. @mart 3D store, www.activeworlds.com (last access on August 2006)
2. AlphaWorld virtual community, www.activeworlds.com (last access on August 2006)
3. Bitmanagement Contact, www.bitmanagement.com (last access on August 2006)
4. Barbieri, T., Paolini, P.: Reconstructing Leonardo's Ideal City - from Handwritten Codexes to Webtalk-II: a 3D Collaborative Virtual Environment System. In: Proc. of the 2001 Conference on Virtual Reality, Archeology, and Cultural Heritage (VAST 2001), Athens, Greece. ACM Press, 61-66
5. Brusilovsky, P.: Adaptive navigation support. In Brusilovsky, P., Kobsa, A., Nejdl, W. (eds.): The Adaptive Web: Methods and Strategies of Web

Personalization, Lecture Notes in Computer Science, Vol. 4321. Springer-Verlag, Berlin Heidelberg New York (2007) this volume
6. Brusilovsky, P.: Adaptive Navigation Support: From Adaptive Hypermedia to the Adaptive Web and Beyond. Psychnology Journal **2** 1 (2004) 7-23
7. Brusilovsky, P., Millán, E.: User models for adaptive hypermedia and adaptive educational systems. In Brusilovsky, P., Kobsa, A., Nejdl, W. (eds.): The Adaptive Web: Methods and Strategies of Web Personalization, Lecture Notes in Computer Science, Vol. 4321. Springer-Verlag, Berlin Heidelberg New York (2007) this volume
8. Brutzman, D.: The Virtual Reality Modeling Language and Java. Communications of the ACM **41** 6 (1998) 57-64
9. Bunt, A., Carenini, G., Conati, C.: Adaptive Content Presentation for the Web. In Brusilovsky, P., Kobsa, A., Nejdl, W. (eds.): The Adaptive Web: Methods and Strategies of Web Personalization, Lecture Notes in Computer Science, Vol. 4321. Springer-Verlag, Berlin Heidelberg New York (2007) this volume
10. Carey, R., Bell, G.: The annotated VRML97 Reference Manual. Addison Wesley (1997)
11. Celentano A., Pittarello F.: Observing and Adapting User Behavior in Navigational 3D Interfaces. In: Proc. of 7th International Conference on Advanced Visual Interfaces 2004 (AVI 2004), Gallipoli, Italy. ACM Press (2004) 275-282
12. Chittaro L., Ieronutti L., Ranon R.: Navigating 3D Virtual Environments by Following Embodied Agents: a Proposal and its Informal Evaluation on a Virtual Museum Application. Psychnology Journal **2** 1 (2004) 24-42
13. Chittaro L., Ranon R., Ieronutti L.: Guiding Visitors of Web3D Worlds through Automatically Generated Tours. In: Proc. of the 8th International Conference on 3D Web Technology (Web3D 2003), St. Malo, France. ACM Press (2003) 85-91
14. Chittaro L., Ranon R.: New Directions for the Design of Virtual Reality Interfaces to E-Commerce Sites. In: Proc. of the 5th International Conference on Advanced Visual Interfaces (AVI 2002), Trento, Italy. ACM Press (2002) 308-315
15. Chittaro L., Ranon R.: Using the X3D Language for Adaptive Manipulation of 3D Web Content. In: Proc. of the 3rd International Conference on Adaptive Hypermedia and Adaptive Web-based Systems (AH 2004), Eindhoven, Netherlands. Springer-Verlag, Berlin Heidelberg New York (2004) 287-290
16. Chittaro L., Ranon R.: Adding Adaptive Features to Virtual Reality Interfaces for E-Commerce. In: Proc. of the 1st International Conference on Adaptive Hypermedia and Adaptive Web-based Systems (AH 2000), Trento, Italy. Springer-Verlag, Berlin Heidelberg New York (2000) 86-97.
17. Chittaro L., Ranon R.: Dynamic Generation of Personalized VRML Content: a General Approach and its Application to 3D E-Commerce. In: Proc. of the 7th International Conference on 3D Web Technology (Web3D 2002), Tempe, Arizona. ACM Press (2002) 145-154.
18. Chittaro L., Ranon R.: Web3D Technologies in Learning, Education and Training: Motivations, Issues, Opportunities. Computers and Education, in press. doi:10.1016/j.compedu.2005.06.002, published online in 2005.
19. Corvaglia D., Virtual Training for Manufacturing and Maintenance based on Web3D Technologies. In: Proc. of the 1st International Workshop on Web3D Technologies in Learning, Education and Training (LET-WEB3D 2004), Udine, Italy, (2004) 28-33
20. Dachselt, R. , Hinz , M., Meissner , K. : Contigra: an XML-based architecture for component-oriented 3D applications. In Proc. of the 7th International Conference on 3D Web Technology (Web3D 2002), Tempe, Arizona, USA. ACM Press (2002) 155-163

21. Dachselt , R., Hinz, M., Pietschmann, S. Using the Amacont Architecture for Flexible Adaptation of 3D Web Applications. In: Proc. of the 11th International Conference on 3D Web Technology (Web3D 2006), Columbia, Maryland, USA. ACM Press (2006), 75-84.
22. De Bra, P., Aerts, A., Berden, B., De Lange, B., Rousseau, B., Santic, T., Smits, D., Stash, N.: AHA! The Adaptive Hypermedia Architecture. In: Proc. of the 14th Conference on Hypertext and Hypermedia (Hypertext 2003), Nottingham, UK. ACM Press (2003) 81-84.
23. Dieberger, A.: Browsing the WWW by interacting with a textual virtual environment - A framework for experimenting with navigational metaphors. In: Proc. of the 7th Conference on Hypertext (Hypertext 1996), Washington DC, USA. ACM Press (1996) 170-179.
24. dos Santos, C. T., Osorio, F. S.: An Intelligent and Adaptive Virtual Environment and its Application in Distance Learning. In: Proc. of the 6th International Conference on Advanced Visual Interfaces (AVI 2004), Gallipoly, Italy. ACM Press (2004) 362-365.
25. Fiala, Z., Hinz, M., Meissner, R. K., Wehner, F. A Component-based Approach for Adaptive, Dynamic Web Documents. Journal of Web Engineering **2** 1-2 (2003) 58–73.
26. Fiat Ireland home page, www.fiat.ie (last access on August 2006)
27. Hughes, S., Brusilovsky, P., Lewis, M.: Adaptive navigation support in 3D e-commerce activities. In: Proc. of Workshop on Recommendation and Personalization in eCommerce at AH 2002 (2002) 132-139.
28. ISO/IEC FCD 19774 (2004). Humanoid animation (H-Anim) Available: www.web3d.org/x3d/specifications/ISO-IEC-19774-HumanoidAnimation/ (last access on August 2006)
29. Java3D media framework, java.sun.com/products/java-media/3D/ (last access on August 2006)
30. John, N. W.: The Impact of Web3D Technologies on Medical Education and Training. Computers and Education, in press. doi:10.1016/j.compedu.2005.06.003, published online in 2005.
31. Karpouzis, K., Caridakis, G., Fotinea, S. E., Efthimiou, E.: Educational Resources and Implementation of a Greek Sign Language Synthesis Architecture. Computers and Education, 2005, in press. doi:10.1016/j.compedu.2005.06.004, published online in 2005.
32. Macromedia Shockwave, www.adobe.com/products/shockwaveplayer/ (last access on August 2006)
33. Mediamachines Flux, www.mediamachines.com (last access on August 2006)
34. MPEG-4 International Standard (2002). MPEG-4 Specification. International Standard ISO/IEC JTC1/SC29/WG11 N4668.
35. Octaga Player, www.octaga.com (last access on August 2006)
36. Outline 3D Web site, www.outline3d.com (last access on August 2006)
37. ParallelGraphics Cortona, www.parallelgraphics.com /products/cortona/ (last access on August 2006)
38. ParallelGraphics Virtual Manuals, www.parallelgraphics.com/virtual-manuals (last access on August 2006)
39. Phillips, N., John, N.W.: Web-based Surgical Simulation for Ventricular Catheterisation. Neurosurgery **46** 4 (2000) 933-937.
40. Sims, E.M.: Reusable, Lifelike Virtual Humans for Mentoring and Role-Playing. Computers and Education, in press. doi:10.1016/j.compedu.2005.06.006, published online in 2005.
41. The Web3D Consortium, www.web3d.org (last accessed on August 2006).

42. Udine3D, udine3d.uniud.it (last access on August 2006)
43. van Mulken, S., André, E., Muller, J.: The Persona Effect: How Substantial is it? In: Proc. of the 1998 Human Computer Interaction Conference (HCI'98), Sheffield, UK. Springer-Verlag, Berlin Heidelberg New York (1998) 53-66.
44. Virtual Ljubljana, www.ljubljana-tourism.si/en/ljubljana/virtual_ljubljana/ (last access on August 2006)
45. VR for all community site, www.vr4all.net (last access on August 2006)
46. VRML International Standard (1997). VRML97 Functional Specification. International Standard ISO/IEC 14772-1:1997. Available at www.web3d.org/x3d/specifications/vrml/ISO-IEC-14772-VRML97/ (last access on August 2006).
47. Wakita, A., Hayashi, T., Kanai, T., Chiyokura, H.: Using Lattice for Web-based Medical Applications. In: Proc. of the 6th International Conference on 3D Web Technology (Web3D 2003), Saint Malo, France. ACM Press (2003), 29-34.
48. Walczak, K., Cellary, W.: X-VRML for Advanced Virtual Reality Applications. IEEE Computer, **36** 3 (2003) 89-92.
49. 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 August 2006).
50. Mzoughi, T., Davis Herring, S., Foley, J. T., Morris, J. M., Gilbert, P. J: WebTOP: A 3D Interactive System for Teaching and Learning Optics. Computers and Education, in press. doi:10.1016/j.compedu.2005.06.008, published online in 2005.