Dynamic Generation of Personalized VRML Content: a General Approach and its Application to 3D E-Commerce

Luca Chittaro
HCI Lab, Dept. of Math and Computer Science
University of Udine
via delle Scienze 206, 33100 Udine, Italy
+39 0432 558450
chittaro@dimi.uniud.it

Roberto Ranon
HCI Lab, Dept. of Math and Computer Science
University of Udine
via delle Scienze 206, 33100 Udine, Italy
+39 0432 558457
ranon@dimi.uniud.it

ABSTRACT

The capability of (semi)automatically adapting the content, structure, and/or presentation of a Web site to address the interests and preferences of each individual user is more and more considered as a key factor to increase user satisfaction and building customer loyalty. However, while a large body of literature is available about making traditional Web sites adaptive, it is surprising that no research effort has been yet devoted to the problem of adapting Web3D content and presentation. This paper focuses on this topic, proposing a general approach to build adaptive 3D Web sites, and illustrating a specific application of the approach to a 3D e-commerce case study. The architecture we propose, called AWE3D (Adaptive WEb 3D), is based on a combination of readily available and platform-independent Web technologies so that it can be reproduced by interested readers.

Categories and Subject Descriptors

I.3.6 [**Computer Graphics**]: Methodology and Techniques – *Interaction techniques*. H.5.4 [**Information Interfaces and Presentation**]: Hypertext/Hypermedia – *Architectures.* D.2.11 [**Software Engineering**]: Software Architectures. H.5.1 [**Information Interfaces and Presentation**]: Multimedia Information Systems – *Artificial, augmented, and virtual realities.*

General Terms

Design, Human Factors.

Keywords

Adaptive interfaces, VRML, Web architectures, e-commerce.

1. INTRODUCTION

A software system that is capable of personalizing its interaction with individual users, based on the information it has about them, is typically called *user-adaptive system* [9]. In several application domains (ranging from tutoring to e-commerce), user-adaptive systems have already proven to be more effective and/or usable than non-adaptive ones [9], and *personalization*, i.e. the capability of (semi)automatically addressing the interests and preferences of each individual user, is more and more considered as a key factor to add value to various software applications, increasing user satisfaction and building customer loyalty. The Web is one of the domains where user-adaptive systems can bring many advantages. For example, *adaptive Web sites* are capable of improving their organization and presentation by learning from visitor access patterns [14].

The general idea behind user-adaptive systems is to build a *user model* (i.e., a suitable representation of the user's characteristics - such as preferences and interests - that are relevant in the considered application), and exploit it for personalization purposes. For example, in a tutoring system a model of the student's knowledge can be exploited to tailor the choice of specific exercises and the presentation of the teaching material to the student's needs. A user model can be built by employing different techniques, ranging from directly asking specific questions (e.g., asking the student to rate its initial knowledge about different topics) to deriving information from monitoring the usage of the system itself (e.g., checking the answers given by the student to exercises).

Adaptive software systems have been first studied by the User Modeling community, and the problem of building adaptive Web sites is of central importance in the field of Adaptive Hypermedia [4]. However, while a large body of literature is available about making traditional Web sites adaptive, it is surprising that no research effort has been yet devoted to the problem of adapting Web3D content and presentation. For this reason, our research focuses precisely on this topic, proposing an approach to build adaptive 3D Web sites.

The paper has three main goals. First, we aim at showing that personalization of Web3D content requires to take a different approach with respect to those available for traditional Web sites, because there are both new problems to solve and different (in some cases, more powerful) available features for building the user model and performing the adaptation.

***Web3D'02*, February 24-28, 2002, Tempe, Arizona, USA.**

Second, we illustrate our general approach to build adaptive 3D Web sites. The architecture we propose, called AWE3D (Adaptive WEb 3D), is based on a combination of readily available and platform-independent Web technologies so that it can be reproduced by interested readers. In the paper, we describe the logic modules of AWE3D and their functions, also discussing the technical choices we made in order to implement the architecture.

Third, we provide a practical example of how the general AWE3D architecture can be instantiated in a specific domain, by describing in detail a 3D e-commerce case study concerning an adaptive 3D store, in which some features (e.g., product display, advertising, store size and style,…) are adapted to the customer preferences and interests.

2. ADAPTING WEB3D CONTENT: OUR PROPOSED APPROACH

In this section, we propose our approach to adaptive 3D Web sites, by: (i) discussing the problem of personalization of VRML worlds, (ii) describing our AWE3D architecture in terms of modules and their functions, and (iii) discussing the technical choices we made to implement AWE3D. Although we choose VRML97 as the specific language for describing 3D content, our proposal can be reformulated for other Web3D technologies, provided that they allow one to (i) monitor user's actions while (s)he visits the 3D world (this is achieved with *sensors* in VRML), and (ii) assemble files of 3D content starting from parametric descriptions of 3D models (this is achieved with PROTOs in VRML).

2.1 The Adaptation Problem and VRML

In general, the problem of making a system adaptive can be broken down into three tasks [11]. The *acquisition task* is concerned with (i) identifying available information about the user, either by monitoring the system usage or by obtaining information from external sources, (ii) making this information accessible to the adaptation component of the system, and (iii) building an initial model of the user. A distinction is usually made between *user data* (i.e., information about personal characteristics of the user such as demographic data), and *usage data* (i.e., information about user's behavior with the interactive system and/or user's environment). The *representation task* is concerned with (i) expressing the content of the user and usage data appropriately in a formal system to allow access and further processing, and (ii) drawing further assumptions about users and/or user groups, their behavior, and their environment, thereby integrating information from various sources. Finally, the *production task* is concerned with generating and adapting content, presentation, and structure, based on a given user, usage and environment model.

Acquisition of data in 3D worlds and production of personalized 3D worlds are the tasks which present new problems with respect to the traditional 2D Web site scenario; traditional user model representation techniques can be instead reused in the 3D context. Therefore, in this paper, we mostly focus on the acquisition and production tasks. As we illustrate in the following, VRML offers powerful possibilities for both tasks.

With respect to the acquisition task, VRML comes with a range of sensors (e.g., visibility, proximity, touch,…) that can be used in order to monitor user's behavior in the 3D world. For example, a typical problem of adaptive Web sites is to keep track of which elements of a page have been seen by a specific user. Unlike traditional 2D sites, where it is often assumed that every element of a downloaded page has been seen, a VRML world allows one to track better what elements the user is looking at, by checking that two conditions hold: (i) the virtual head of the user is oriented towards the element (this can be checked with VRML visibility sensors), and (ii) the user is near enough the element in the 3D space (this can be checked with VRML proximity sensors). Moreover, in case the predefined sensors are not sufficient for the Web site designer's purpose, one could possibly resort to the External Authoring Interface (EAI) to obtain complete monitoring of the user behavior (e.g., one could write a Java program that tracks user's paths in the VRML world).

With respect to the production task, VRML PROTOs are a natural tool to build a set of generic elements, whose possible instantiations can be used to generate different personalizations. In particular, the different features that need to be personalized will correspond to fields of the PROTO. As an example, consider the problem of including a 3D model of a given car in a VRML world according to what is known about the user's preferred car style (from colorful and trendy to traditional and serious). To make the adaptation possible, the car is generically represented as a PROTO in which colors and textures for the exterior and interior have to be passed as values of some fields. Then, when assembling the personalized VRML world, the car PROTO is instantiated with the colors and textures that better match user's preferences. In this way, a large number of possibilities for personalization is available. The size of the adaptation space can indeed be exponential in the number of PROTOs: for example, if the VRML world is composed by n PROTOS and each PROTO can have i different instantiations, then the space of possible adaptations is composed by i^n VRML worlds, where the i parameter depends on the number of fields of PROTOs and their number of possible values (for example, if every PROTO is composed by e fields and each field can take v different values, then $i=v^e$).

Moreover, if the VRML content to be adapted is systematically organized as a library of PROTOs, the work of the 3D model designer can be logically separated from the work of the personalization designer, who can develop a program that automatically assembles a personalized VRML file, by properly choosing and instantiating a set of available PROTOs.

2.2 The AWE3D Architecture and its Modules

The architecture we propose for adaptive 3D Web sites is illustrated in Figure 1, and is composed by the following logic modules:

- A *Usage Data Sensing* module, whose purpose is to monitor the user's interaction with the VRML world, and send the relevant events through the Internet. This module is located on the client side, running in the VRML 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*, possibly performing simple calculations on that data (such as averages or totals of values).
- A *Personalization* module, composed by two sub-modules, respectively called *User Model Update Rules* and *Web3D Personalization Rules*. The purpose of the first sub-module is to perform inferences needed to update the user model, e.g. deriving assumptions on the user's interests on the basis of the elements (s)he examined more frequently in the 3D world. The purpose of the second sub-module is to decide the personalizations that should be performed on the basis of the current user model. These personalization choices are then stored in the user model itself. The *Personalization* module can be periodically run at given intervals of time or after a certain number of user's visits to the Web site or when the contents and the adaptation strategies of the Web site are updated by the application designer. If the module is run after each visit, users will notice the effect of the adaptation in their next visit.
- A *VRML World Creator* module, that takes as input the personalization choices derived by the previous module, and produces a VRML world that implements them, by choosing a set of proper VRML PROTOs and instantiating them as required by the personalization choices. The *VRML World Creator* retrieves the code and files needed to build the VRML world (e.g., PROTOs, textures, sounds,…) from the *VRML Content Database*. The personalized VRML world is then made accessible to the user.

We now describe in detail the technical choices we took to implement each module.

2.2.1 The Usage Data Sensing Module

To implement *Usage Data Sensing* in VRML worlds, we use a combination of VRML sensors and a *SendUsageData* script (written in Java). By using VRML sensors, one can for example track what the user touches, moves, and is near to in the 3D world. Type and location of VRML sensors in the 3D world has to be decided by the application designer according to what needs to be measured about user's behavior. The sensors output is routed to the *SendUsageData* script, which transmits that usage information to the *Usage Data Recorder* module by using HTTP "post" requests, or temporarily caches it for later transmission. The latter case is appropriate for applications that need to send a large amount of usage data: to avoid burdening the Internet connection, usage data can be periodically sent in packets at an appropriate frequency.

By combining the output of different VRML sensors, one can obtain a more elaborate tracking of user's actions: for example, as we have discussed in the previous section, a combination of visibility and proximity sensors allows one to determine with good confidence if the user has actually seen an element of the world. To perform the tracking, application-dependent scripts are added to the module in order to pre-process sensors' outputs following the required logic and then forward the derived usage data to the *SendUsageData* script. Some additional examples of user's actions tracking can be: (i) in a 3D virtual city, one has the possibility to track the places the user has visited and how much time she has spent there by using proximity and time sensors associated to buildings, (ii) in a training application, where the user solves exercises by moving 3D objects in proper spatial configurations (e.g., assembly of mechanical parts), the employed motion sensors can be used to evaluate how adequately exercises have been solved by the user.

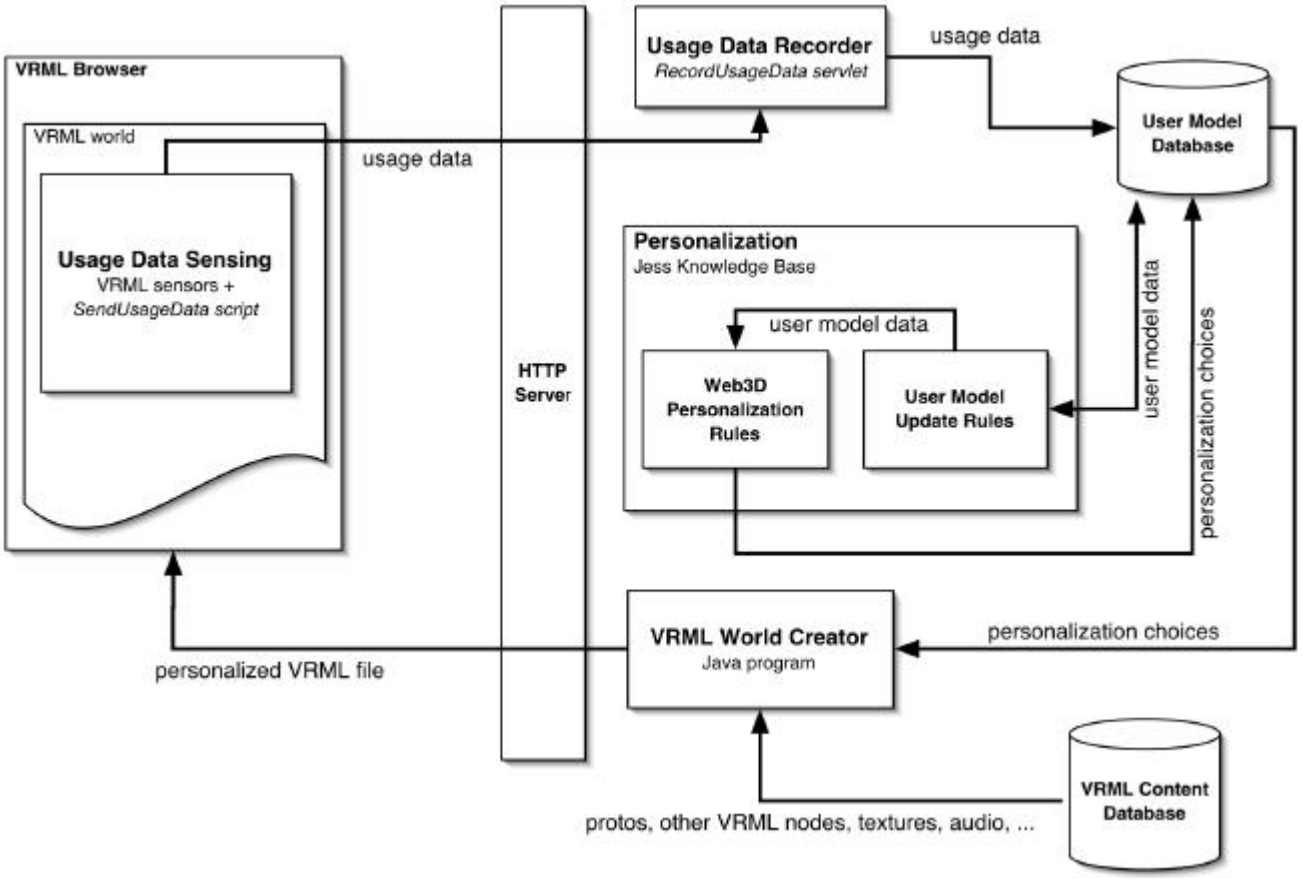

Figure 1. Schema of the AWE3D architecture for adaptive 3D Web sites.

2.2.2 The Usage Data Recorder Module

To implement the *Usage Data Recorder*, one has to choose among the available technologies (such as CGI programs or Java servlets) used by Web sites to deal with data sent by the browser. We chose the (more flexible and scalable) servlet solution, and adopted the JDBC protocol to connect to a DBMS server, where data are stored by using SQL commands. The implemented servlet (called *RecordUsageData*) simply catches the HTTP "post" requests sent by the *SendUsageData* script, stores the received data in the *User Model Database*, and possibly performs simple calculations (averages, totals,…) on the acquired data. It can be specialized for a given application to return back data that are possibly needed to respond to user's actions. For example, in a 3D e-commerce site, the servlet can receive and record data about user's clicks on the displayed products, but also retrieve a textual description of the clicked products from a product database and return it to be shown to the user in a HTML frame.

2.2.3 The Personalization Module

The technical choices that have to be taken in implementing the *Personalization* module depend on how complex are the inferences that have to be performed. Since, in general, non-trivial personalizations require the designer to resort to rule-based software tools, we adopted *Jess* (*Java Expert System Shell*) as the language for expressing personalization rules in our AWE3D architecture. Jess [10] is a rule engine and scripting environment (inspired by the *CLIPS* expert system shell) written entirely in Java, and thus easily integrated with our servlet and databases. By writing rules in *Jess*, it is possible to retrieve user model data from the *User Model Database* through JDBC, and use it both to update the user model (*User Model Update Rules*), and to generate personalization choices for the 3D world (*Web3D Personalization Rules*). The generated choices are then stored in the *User Model Database*, and given as input to the *VRML World Creator* module when the user visits the Web site.

2.2.4 The VRML World Creator Module

The *VRML World Creator* is implemented by a simple Java program that, depending on the specifications produced by the *Personalization* module, assembles a VRML file for the adapted world. More specifically, the *VRML World Creator* retrieves PROTO definitions from the *VRML Content Database* through JDBC, and adds to them the statements required for their instantiations. The personalized VRML file is then made accessible to the user's browser.

In general, the personalized VRML file contains only one adaptation of the VRML world. This is done mainly for efficiency considerations: in this way, only the strictly needed code is included in the VRML file (thus keeping file size to a minimum). However, if one needs to include the possibility of changing some adaptation features while being inside the world (and immediately seeing the effect), one can still allow this by including in the PROTO of the corresponding objects all the necessary adaptations (and a mechanism to interactively choose between them). In this way, the *VRML World Creator* would include a PROTO that can undergo interactive changes. However, one should be very careful with this feature, since it can (potentially) increase a lot the size of the VRML file.

To give some examples of personalized world creation, we consider again the two examples made at the end of Section 2.2.1: (i) in the virtual city example, if the PROTOs for the different attractions have a field for the shape node, one can instantiate the PROTOs with geometries and textures of different quality (according to how relevant that kind of attraction is for the user) to keep the world size under a given limit (set according to user's available bandwidth), (ii) in the tutoring application, the initial position of an object in the 3D space can be a field of the corresponding PROTO, allowing the system to generate personalized exercises, taking into account the configurations of objects from those exercises which have been inadequately solved.

3. APPLICATION TO A 3D E-COMMERCE EXAMPLE

Although almost all e-commerce sites on the Internet use traditional 2D interfaces, some sites are deploying 3D interfaces to attract customers. A 3D interface can bring relevant benefits, if properly designed: (i) it is closer to the real-world shopping experience, and thus more familiar to the customer, (ii) it supports customer's natural shopping actions, (iii) it can satisfy the needs of customers who have an emotional style of buying, by providing a more immersive, interactive, and visually attractive experience, (iv) it can even satisfy social needs, by allowing customers to meet and interact with people (e.g., other customers or salespeople). However, to become more important in the future, 3D e-commerce has still to face major challenges (e.g., browser compatibility issues, navigation and usability issues,…), and needs to be provided with proper tools to develop effective Web sites and use them. For example, while for 2D e-commerce both specific research about adaptivity (e.g. [1, 2, 9, 11, 12, 13]), and commercial developer tools for building adaptive Web sites (see [9] for a review of them) are available, the relevant issue of personalization in 3D e-commerce has yet to be faced. From this point of view, providing the application developer with tools for personalization (as it is typical of successful 2D sites) is an important aspect to make 3D solutions more viable.

For this reason, to show an example of how the general AWE3D architecture can be employed in a practical case study, we choose an e-commerce scenario. In the following, we first briefly introduce the chosen 3D world, then we discuss in detail the specific choices taken for each module of the AWE3D architecture in order to implement the considered example, with a particular emphasis on VRML coding solutions.

3.1 The 3D Store World

The VRML world we choose as an example represents an architectural 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, and putting them in the cart, which is also represented in 3D (see Figures 2 and 3). Besides shelves, customers' attention towards products is sought by exploiting special rotating display spots in prominent places, advertisements on the wall, and audio messages. Moreover, the store is populated by *walking products*, an interface element we proposed in [6] to help users in finding products in 3D stores. Walking products (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 (s)he is looking for a specific type of products, (s)he has just to follow any WP of that type and will be quickly and easily lead to the desired destination. The specific path followed by the WP to accompany the customer to his/her destination is chosen by taking also into account the merchandising strategy of the store. If the customer wants to stop along the way and take a look at other products, s(he) can count on the fact that WPs will be always available to lead her/him to the original destination. For a more detailed discussion of WPs, and a comparison with other navigation aids, see [6].

Figure 2. An adaptation of the 3D store.

Figure 3. A second adaptation of the 3D store.

3.2 The Usage Data Sensing Module in the 3D Store Example

In the considered example, the specific data collected by the *Usage Data Sensing* module by monitoring customer actions are:

- *Seen Products*. While the customer wanders around the store, (s)he voluntarily or involuntarily looks at the products which fall in her field of view. Products which have been seen by the customer are tracked by the module.
- *Clicked Products*. When the customer wants to know more about a product, (s)he clicks on it to get the product description. The event is detected by the module.
- *Cart Products*. The product description allows the customer to put the product in the shopping cart for a possible later

purchase. The module tracks which products have been put in the shopping cart.

- *Purchased Products*. If a product in the cart is later purchased by going to the checkout counter, this action is detected by the module.

Seen Products and *Clicked Products* data are acquired through sensors associated to the PROTO describing a product, while *Cart Products* and *Purchased Products* data are handled in the PROTOs describing the cart and the checkout counter, respectively. As an example, the VRML code for the *Product* PROTO is given in Figure 4. A touch sensor (called *Click*), a visibility sensor (called *Visible*), a proximity sensor (called *Near*), and one application-dependent script (called *ProcessProductEvents*) are included in the body of the PROTO with proper DEF declarations. The *Click*, *Visible*, and *Near* sensors respectively detect when the product is clicked, when the product is inside the avatar's field of view, and when the user is near enough the product. When a sensor is activated by user's actions, it notifies the script through a proper ROUTE declaration (see the last three lines of the PROTO). When the script receives the event associated to the *Near* or the *Visible* sensors, it checks whether both the *Near* and *Visible* sensors are simultaneously active: if this is the case, it sends (through its *productSeen* eventOut) the unique identifier of the product (*ProductID*) to the *SendUsageData* script. When the *ProcessProductEvents* script receives the event associated to the *Click* sensor, it sends the *ProductID* to the *SendUsageData* script. The PROTO interface (see first lines of the PROTO) comprises: (i) the two eventOut for connecting to the *SendUsageData* script (the routing from specific instances of products to the *SendUsageData* script is performed at instantiation time, and will thus be discussed in Section 3.5), and (ii) a number of fields that respectively allow one to specify which 3D model has to be used for the product (*Product3DModel*), what is the unique identifier of the product (*ProductID*), and what are the parameters (*proximitySize* and *visibilitySize*) for the visibility and proximity sensors (in general, the size of the visibility box should be minimal, but contain the whole product, while the size of the proximity box should also take into account the minimum distance at which the user is able to clearly see the product).

3.3 The User Model Database in the 3D Store Example

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

- Demographic data (e.g. gender, year of birth, product categories of interest among those available in the store), which the customer can enter through an HTML form.
- User preferences about the store (e.g., presence of audio and music, preferred music genre, preferred store size and style), which are also entered or modified by the user through the HTML form.
- Usage data, recorded by the *Usage Data Recorder* module, and exploited to dynamically update the user model.
- the *product interest ranking*, which tries to order product categories according to customer's interests.

In particular, 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.

To determine the product interest ranking, an initial value is determined in two different ways: (i) the above mentioned HTML form allows the customer to fill fields about his/her products of interests: if (s)he chooses to do it, the information is used to initialize the ranking, (ii) if the customer does not provide product interests in the HTML form, some *User Model Update Rules* in the *Personalization* module try to predict interests by using demographic profiles (e.g., a customer in the 16-24 age range is very likely to be interested in the latest models of cellular phones). Then, regardless of the quality of the initial value, product interests will be continuously updated by other *User Model Update Rules* which exploit usage data: each purchase, cart insertion, and click of a product increases (with different weights) the level of interest in the corresponding product category.

```
PROTO   Product [
        field MFNode  Product3DModel Shape {}
        field SFInt32 ProductID 0
        field SFVec3f proximitySize 0.0 0.0 0.0
        field SFVec3f visibilitySize 0.0 0.0 0.0
        eventOut SFInt32 ProductSeen
        eventOut SFInt32 ProductClicked
]

{
        Group {
                children [
                        Transform {children IS Product3DModel}
                        DEF Click TouchSensor  { }
                        DEF Visible VisibilitySensor {
                                size IS visibilitySize
                        }
                        DEF Near ProximitySensor {
                                size IS proximitySize
                        }
                        DEF ProcessProductEvents Script   {
                                 field SFInt32 PID IS ProductID
                                 eventIn SFBool IsClicked
                                 eventIn SFBool IsVisible
                                 eventIn SFBool IsNear
                                 eventOut SFInt32 seen IS ProductSeen
                                 eventOut SFInt32 clicked IS ProductClicked
                                 url "ProcessProductEvents.class"
                        }
                ]
         }

        ROUTE Click.isActive   TO  ProcessProductEvents.IsClicked
        ROUTE Visible.isActive TO ProcessProductEvents.IsVisible
        ROUTE Near.isActive    TO ProcessProductEvents.IsNear
}
```

Figure 4. PROTO of a product in the 3D e-commerce example.

3.4 The Personalization Module in the 3D Store Example

In this section, we present some examples of rules we introduced in the *Personalization* module to perform adaptations in the 3D store.

Simple rules are given by the direct associations between the user preferences about *size* and *style* of the store and specific 3D models for the virtual building.

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, e.g. 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 in the *Personalization* module:

- *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 Figures 2 and 3 show different possible allocations of shelf space.
- *DisplaySpot(X)* is false if product *X* is displayed only on its shelf (together with other products of its category), while it is true if product *X* is displayed also in a separate display spot in a prominent place.
- *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)*. *Web3D 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 in it cannot be increased).

We now examine some specific rules, and how they relate to the information recorded in the user model (we will formulate rules in a intuitive format that does not require familiarity with the Jess language). 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 (*seen(X)* is the recorded number of times a product has been seen, *ProductInterest(X)* is the rank in the product interest ranking, *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 case when 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, e.g., if a customer buys a computer and has never purchased a printer, (s)he could be soon interested in a printer. The rule can be expresses as follows (*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 increasing 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 uneffective merchandising strategies, e.g. 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.

Many researchers in the Adaptive Interfaces community, e.g. [3, 8], stress that adaptive systems must reconcile adaptivity with stability in the user environment, otherwise users can be confused if the organization of the environment is constantly changing. Therefore, 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 general criterion we are using is to keep the experience of returning to the 3D store consistent with the familiar experience of returning to a known real-world store: (i) the store layout and style remain essentially the same (these parameters are indeed under user control, and are not changed autonomously by the *Personalization* module, unless the user explicitly modifies its preferences about *size* and *style*), and (ii) a limited number of changes concern what products are displayed, and how the attention of the customer towards those products is sought.

3.5 The VRML World Creator Module in the 3D Store Example

The personalization choices received by the *VRML World Creator* concern:

- *Store Layout and Look*. The preferred size and style information provided by the customer are used to choose store layout and look of the 3D representation of the store. For example, the stores in Figures 2 and 3 show two different sizes and styles available. In this way, the customer can visit a 3D store which is closer to the ones (s)he chooses in the real world (or safely experiment with stores she would like to try in the real world, but avoids, e.g. for emotional reasons such as fear of judgment). Store Organization. This concerns product placement, product space, and banners.
- *Set of WPs*. This concerns how many WPs are in the store and which products are associated to them.
- *Audio*. Unlike real stores, the chosen voice messages can be targeted to the specific customer, both in the promoted product, in the type of voice, and in the choice of words (e.g., a teenager and a elder customer prefer very different kinds of message style, voice, and emphasis).

Every above mentioned choice is satisfied by the *VRML World Creator* by retrieving one or more PROTOs from the *VRML Content Database* and properly instantiating them into the VRML file of the personalized VRML world. As an example, we consider a product instantiation: the product PROTO has been already illustrated in Figure 4, while Figure 5 shows the VRML code added by the *VRML World Creator* to obtain an instantiation of a specific product. More specifically, the considered code defines an instance of a product called *Product001*, associating it with a 3D model representing a small box of biscuits (which is retrieved from the *VRML Content Database*), assigning it the value 1 as *productID*, and defining the size of the sensors. The two ROUTE declarations connect the outputs of the script inside the *Product001* instance to proper inputs (called *IDSeen* and *IDClicked*) of the *SendUsageData* script.

```
DEF     Product001 Product {
        Product3DModel  Inline {url "SmallBoxOfBiscuits.wrl"}
         ProductID 1
         proximitySize 1.0 1.0 1.0
         visibilitySize 1.0 1.0 1.0
}

ROUTE Prod1.ProductSeen TO SendUsageData.IDSeen
ROUTE Prod1.ProductClicked TO SendUsageData.IDClicked
```

Figure 5. Instantiating a Product PROTO in the 3D e-commerce example.

4. CONCLUSIONS AND FUTURE WORK

In this paper, we have both proposed a general architecture for adaptive 3D Web sites and shown its application to a 3D e-commerce case study.

We are currently performing some experiments with users employing the described 3D e-commerce example, and some preliminary results have been presented in [7]. In general, the reactions of users to the system are mostly favorable. We plan to proceed with the evaluation, following the indications given by [5]. More generally, our future work comprises two sets of goals: one refers to the general architecture, while the other is specific to its application.

With respect to the general architecture, we intend to investigate the impact of extending it to multi-user worlds. From this point of view, it is interesting to note that a multi-user 3D world can conflict with personalization aspects, limiting the possibilities of adapting the site. For example, if multiple users have to travel together and interact in the same VRML world, the personalization of the features of that world cannot target anymore the specific profile of a single user. Trying to find the best compromise which maximizes the match with the different user models can be a possible solution, but it would not be easy to implement, considering that the set of users could continuously change.

Moreover, we would like to test how the EAI can be used to improve usage data acquisition and adaptation. With the EAI, one has complete control on all the nodes of the VRML world. Thus, the EAI should be useful both when VRML sensors are impractical or inadequate to record complex events (e.g., tracking detailed user paths followed in the 3D world), and when some forms of on-the-fly adaptation of VRML content are needed (for example, we are considering recording the average speed at which the customer is able to move in the 3D world to get an estimate of her 3D navigation ability and tailor the animation of navigation aids - such as WPs - to it).

With respect to the 3D e-commerce scenario, we will concentrate on how the effectiveness of the 3D store can be augmented with features which would be very difficult or impossible to obtain in a

real store (e.g., automatic rearrangement of products in shelves according to different criteria, such as function, price, brand, chromatic similarities, or user model). Second, we will explore how to give more interactive control to the customer over adaptation aspects (e.g., to switch background music on and off, and to determine music genres preferences, an interactive 3D jukebox can be included in the store). Finally, a more long term goal is to explore the possibility of significantly extending the abilities of WPs, enriching them with further user assistance functionalities besides that of leading customers to specific parts of the store. These extended animated characters would be closer to store assistants, capable for example of addressing simple customer questions, taking the customer to any product shelf, and then performing product presentations.

Finally, we plan to test our AWE3D architecture in other domains: in particular, we are working at two different visualization projects (one in Medicine and one in Physics).

5. ACKNOWLEDGMENTS

Manuela Casanova played an important role in the implementation of the work described in this paper.

Our research activities on Internet technologies are partially supported by the Friuli Venezia Giulia region, under Regional Law 3/98.

Our research activities on visualization are partially supported by the Italian Ministry of University and Research (MURST), under the COFIN 2000 program.

Finally, we would like to thank one of the anonymous reviewers for the valuable comments which helped us improve the paper.

6. REFERENCES

[1] Ardissono, L., and Goy, A. Tailoring the interaction with users in electronic shops. In Proceedings of UM99: 7th International Conference on User Modeling (Banff, Canada, 1999), Springer Verlag, 35-44.

[2] Ardissono, L., Goy, A., Meo, R., Petrone, G., Console, L., Lesmo, L., Simone, C., Torasso, P. A configurable system for the construction of adaptive virtual stores. World Wide Web Journal, 2 (1999), 143-159.

[3] Boyle, C., and Encarnacion, A.O. Metadoc: An Adaptive Hypertext Reading System. User Modeling And User-Adapted Interaction, 4 (1994), 1-19.

[4] Brusilovsky, P. Adaptive hypermedia. User Modeling and User Adapted Interaction, 11(1-2) (2001), 87-110.

[5] Chin, D. Empirical Evaluations of User Models and User-Adapted Systems. User Modeling and User Adapted Interaction, 11(1-2) (2001), 181-194.

[6] Chittaro, L., and Coppola, P. Animated Products as a Navigation Aid for E-commerce. In Proceedings of the CHI2000 Conference on Human Factors in Computing Systems (The Hague, NL, 2000), Extended Abstracts Volume, ACM Press, 107-108.

[7] Chittaro, L., and Ranon, R. Interfacce in Realtà Virtuale per siti di e-commerce: linee guida, ausili alla navigazione, personalizzazione. Proceedings of the SIGCHI ITALY Symposium, 7th Conference of the Italian Ergonomics Society (Florence, I, 2001), SIE, 71-77.

[8] Dufresne, A. ExploraGraph: Improving interfaces to improve adaptive support. In Proceedings of the AIED 2001: 10th International Conference on AI in Education (San Antonio, TX, 2001), IOS Press.

[9] Fink, J., and Kobsa, A. A Review and Analysis of Commercial User Modeling Servers for Personalization on the World Wide Web. User Modeling and User-Adapted Interaction, 10(3-4) (2000), 209-249.

[10] Friedman-Hill, E. Jess, the Expert System Shell for the Java Platform, 1995-2001, http://herzberg.ca.sandia.gov/jess/

[11] Kobsa, A., Koenemann, J., and Pohl W. Personalized Hypermedia Presentation Techniques for Improving Online Customer Relationships. The Knowledge Engineering Review, 16(2) (2001), 111-155.

[12] Joerding, T. User Modeling for Electronic Catalogs and Shopping Malls in the World Wide Web. In Proceedings of the 1st Workshop on Adaptive Systems and User Modeling on the WWW (Chia Laguna, I, 1997), http://fit.gmd.de/UM97/Joerdung.html

[13] Joerding, T. A Temporary User Modeling Approach for Adaptive Shopping on the Web. In Proceedings of the 2nd Workshop on Adaptive Systems and User Modeling on the WWW (Banff, Canada, 1999), http://wwwis.win.tue.nl/asum99/joerding/joerding.html

[14] Perkowitz, M., and Etzioni, O. Adaptive Web Sites. Communications of the ACM, 43(8) (2000), 152-158.