Distributed Virtual Reality Authoring Interfaces for the WWW: the VRSHOP case[1]

I. Varlamis, M. Vazirgiannis,
Dept of Informatics, Athens University of Economics & Business, Patision 76, 10434, Athens, HELLAS

I. Lazaridis
Information and Computer Science University of California, Irvine USA

M. Papageorgiou, T. Panayiotopoulos
Dept of Informatics, University of Piraeus, HELLAS

1 Introduction

Electronic commerce is emerging as an important domain of integration and enhancement of various technologies and research efforts. A trend in e-commerce applications is to provide potential customers/visitors with the ability to preview and "test" products before they buy them. Using a 3-dimensional virtual representation of products is a step forward towards a persuasive preview. It is also very important for users to be able to view products in their own environment. These issues pose several technical and integration challenges.

This work concerns the design and implementation of a system that allows the interactive design of a room, the definition of pieces of furniture and the placement of electrical and electronic appliances. The system works over the WWW offering web-users a tool for designing and visualizing their rooms. The users may define each object in the room, its size, position, and spatial relationships with the room or other objects. Certain integrity constraints are checked during this definition. The system can create on the fly a VRML representation of the specifications and render it to the client. The user may choose to alter and visualise the virtual-world or store it for further reference.

The system architecture follows the 3-tier scheme, where the client is independent of the scheme and architecture of the database, since communication is performed through an application that resides on the server. The three tiers of the system are:

- The database, which contains information on the various appliances and furniture, along with their virtual representations in VRML
- The application server, which is responsible for communicating with the database and manipulating its contents, in order to serve the clients' requests.
- A Java applet on the client side, which offers a highly interactive interface and communicates with the application server to provide the user with an on the fly created 3D visualisation of the room based on VRML.

The VRML based room description, and all the relevant files needed for the creation of the room (images and VRML files of the various furniture and appliances of the room), are downloaded in the client machine in a single compressed file in order to reduce the amount of data transferred. The user may store the compressed file in his/her machine. In this way the Java security constraints are safely overcome.

The contribution of this work is summarized in the following:

- Declarative authoring model for creating virtual worlds putting emphasis to 3D spatial relationships and related spatial integrity constraints.
- Integration and extension of cutting edge technologies in an e-commerce context.

[1] This work has been partially supported by the ESPRIT/VRSHOP project

- Dynamic on the fly creation and modification of virtual reality worlds in a distributed environment.
- A generic, platform-independent approach for storing data on the client side without violating the security aspects of Java.

The system conveys a generic approach to distributed creation and update of virtual worlds as a means of interaction and information dissemination in an e-commerce context. Compared to existing CAD software, our system has several advantages:

- It employs web technologies (VRML, Java), thus meeting the needs of companies that wish to offer web services and promote their products over the web.
- It enables companies to create a virtual representation of various products ranging from furniture and home equipment to clothing products and products that can be found on supermarkets' shelves. This system can cover the needs of all the industry areas that offer products having physical representation.

2 Model

The VR-Shop data model aims at the representation of: i) various entities positioned in a room, ii) the spatial relationships between these entities, and iii) the constraints these entities must satisfy in order to produce a coherent and presentable world.

Essentially, we used our experience in generalised spatiotemporal modelling [EGSV99] to address the simpler issue of developing an authoring tool for rooms. We will briefly describe the main aspects of our work in this field and then present how it relates to the actual model of the room.

2.1 General Data Model for Spatial Composition

A spatial composition is an arrangement of distinct spatial entities ("objects") in a specified position and with a specified orientation within the geometric space of an application (in our case a three dimensional room).

Geometric space is a 3-D Euclidean space using a universal right-hand co-ordinate system. Thus, its three coordinates define every point in this "application space".

An *object* has certain properties: *geometry type, dimension metrics and appearance*. These fully describe the object itself, i.e. without taking into account its positioning within the application. An object-centred local co-ordinate system (LCS) is also defined for each object. The object is rotation- and translation- invariant in this LCS.

Let A be an object of geometry G. Its dimensions are:

$$D^A = (d_1, d_2, ..., d_n) \in D^G \subseteq \Re_+^n \qquad \text{(Eq. 1)}$$

D^A is, an n-size vector of positive values. D^G is the space of allowable dimensions for geometry G. For instance the geometry "Sphere" has dimension space D^{sphere}=[0,∞], and geometry "Sphere with a radius at least 5m" has dimension space $D^{sphere>5}$=[5,∞]. Geometry "Box" has dimension space $D^{box} = \Re_+^3$, since a box is represented by the length (a positive number) of its three sides.

A geometric type may be defined as follows:

Definition 1:

Let ? be an object and O_{xyz}^{A} its object-centred LCS. The function G: $\Re^3 \times D^G \rightarrow \{0,1\}$ defines a geometric type. Object A is of geometric type G if and only if $\forall$ P(x,y,z) the following formula applies:

$$G(x,y,z,d_1,d_2,...,d_n) = \begin{cases} 1, if\ P \in A \\ 0, if\ P \notin A \end{cases} \qquad \text{(Eq. 2)}$$

The above offers a formal definition of objects' geometry. An object is of a certain geometric type G and is fully defined by an n-size vector of dimensions. We must also point out that the term *geometric type* in this context does not convey a concrete ordinary type such as "Sphere", "Polyhedron" etc. On the contrary it is a more generic entity, for instance we may define a geometric type "Human" with a set of dimensional metrics denoting such features as arm or leg length etc.

2.1.1 Simple Spatial Relationships

In order to place the object correctly within the application space, we must define the position of all its points in the Universal Co-ordinate System (UCS). As we have noted, these are fully defined with regards to the object's local LCS (by its geometric type and dimensions). Therefore we must simply define the relationship between the LCS and the UCS to find the position of an object in the UCS.

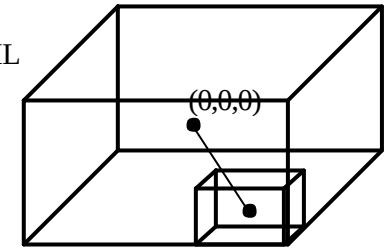
The spatial relationship between two objects consists of the following three constituents: scale, translation and rotation. If $X = (x \quad y \quad z)^T$ are the co-ordinates of a point in the universal system, and $X' = (x' \quad y' \quad z')^T$ in the local system, the relationship is defined as follows:

$$X' = RSX + T \qquad \textbf{(}\text{Eq. 3}\textbf{)}$$

where R is a 3×3 matrix handling the rotation of the LCS with reference to the UCS, S is a 3×3 matrix the scaling of the LCS with reference to the UCS and T is a 3×1 matrix representing the translation of the LCS with reference to the UCS.

The Simple Spatial Relationship (SSR) between two objects in the 3D space is consequently defined by the triple (R,S,T) with R, S, T as defined above.

To give an example, when we create a room, a box is created in VRML having its metacenter at the point (0,0,0) of the UCS. The position in the UCS of the metacenter of a desk that stands in a room corner must be calculated taking into account the room's dimensions, the size of the desk, its rotation and its distance from each of the walls.



2.1.2 High-Order Spatial Relationships

The SSRs described above have the following drawbacks:

- They do not make use of the geometric characteristics of the objects involved, as they only take into account their LCSs.
- They are cumbersome to use, since the user has to specify a set of parameters for the matrices R,S,T.
- They do not express high-level spatial relations that may facilitate the world definition process. Such relations cover topology (e.g. overlap, adjacency, etc.), direction (above, below, etc.) or even distance relationships.

To tackle the above problems, high-order spatial relationships must be defined. Such relationships are semantically richer and retain the functionality and completeness of the SSRs.

Our task, in essence, is to replace the parameters that are necessary for defining the R,S,T of the simple spatial relationship with an alternative set of parameters, which are: i. simpler to use, ii. tailored to the application domain and iii. more meaningful (allowing for faster intuitive understanding of the underlying relationship by human beings).

In the next section, we will present the data model for the VR-Shop system, which aims at a declarative and high-level description of a room content in terms of doors windows, furniture and domestic appliances.

2.2 VR-Shop Spatial Data Model

The VR-Shop Data model includes *objects, spatial relationships* between objects and *constraints* stemming from the real world limitations that the model must address. There are five different object classes. Each of them refers in essence, to the geometric types mentioned earlier. Given a vector of dimensions for each instance of the class (i.e. for each actual object in the room), we fully define its size properties. Each object is also placed in the room by an additional vector of placement data.

2.2.1 Size and Placement Data

Object classes and size-related attributes are presented in Table 1.

Object-Type	**Attributes**
Room	Length, Width, Height
Door	Width, Height
Window	Width, Height
Furniture Item	Length, Width, Height
Domestic Appliance	Length, Width, Height

Table 1. Size related attributes of object classes

Another issue is the placement of the objects in the context of the room. As a consequence, a set of primitives that declaratively describe the relative placement of the objects has to be defined. The placement-related attributes, which describe the spatial relationship of the object with respect to others in the spatial composition of the room, are presented in Table 2.

Object-Type	**Attributes**
Room	<none>
Door	OnWallName, Distance, FromWallName
Window	OnWallName, Distance, FromWallName, HeightFromFloor
Furniture Item	Distance1, FromWall1, Distance2, FromWall2, OrientationAngle, HeightChoice, [FromItem | Height]
Domestic Appliance	Distance1, FromWall1, Distance2, FromWall2, OrientationAngle, HeightChoice, [FromItem | Height]

Table 2. Placement-related attributes of object classes

From the above table it is clear that no placement information is needed for the room, since its co-ordinate system is considered as the UCS.

The position of Doors and Windows is defined first by a wall identifier (on which the door/window is on) and a distance value representing the distance of the door/window from a wall next to the wall in consideration. For window objects, an additional distance is required, that is the height with reference to the floor. Here we have to stress that only the distance from one wall is sufficient to uniquely define the position of the window/door on the wall. To give an example, the position of the front door can be fully specified by the wall it is on (e.g. front) and the distance of the left edge of the door from the left wall (e.g. 2m).

Furniture Items and Domestic Appliances require the following attributes for their full specification:

- Position, i.e. their geometric centre on a point in the ground plane defined by the required distances (Distance1, Distance2) from two related walls (FromWall1, FromWall2).
- OrientationAngle, which refers to a counter-clockwise rotation of the object around an axis, that is vertical to the room floor passing through its geometric centre.
- The height information (HeightChoice). There are several choices here. Either the object (to be precise the bottom surface of the object) is at a specific height from the floor or the object is "on the floor" or hanging "from the ceiling" or it is "on" another item (the user must select which).

2.2.2 Constraints

The use of high-level declarative predicates (*OnWallName, Distance, FromWallName* etc.) defined in the VR-Shop data model provides expressive power but also allows for inconsistencies. Thus, we have to consider the related integrity constraints. The constraints arise both from the geometric configuration of the objects and/or from the attached semantics (e.g. some objects are pieces of furniture while others are appliances). An implicit constraint is that all dimensions and distances must be positive numbers.

Explicit referential constraints checked in our design are the following:

1. All object dimensions must be less or equal than the dimensions of the room.
2. The physical dimensions and position/orientation of objects must not allow any part of them to be outside the room (i.e. all objects must be inside the room).
3. The intersection of any pair of objects must be either empty or at most part of a surface. Objects cannot intrude into one another, since furniture and appliances are considered as solid objects.
4. Each object must have at least one common surface with another object or with the room (there are no objects suspended in mid-air).
5. Furniture items may be placed on other furniture items, but not on domestic appliances.
6. When an object is removed from the room, all objects that are directly "on" that object must revert to a consistent position. Our solution was to place all such objects "on the floor".
7. Cyclic placements of objects "on" each other, either direct or indirect, are not allowed. The following placements are thus prohibited (arrow signifies "on" relationship) (Figure 1):

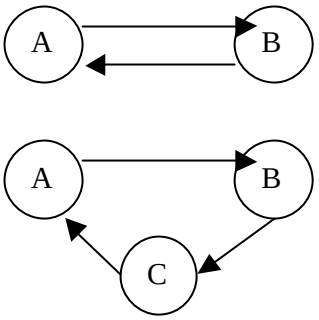


Figure 1. Disallowed object placements

The above list of constraints is not complete. Some semantic constraints have been avoided while others (like #5) have been included. For instance we might have disallowed the placement of certain devices (e.g. "Refrigerator") on others (e.g. "VCR"). However we viewed that the freedom to perform such actions must be given to the user, who –in most cases– will avoid such inconsistencies.

3 Mappings to VRML

VRML defines a set of objects useful for implementing 3D graphics. Each scene in VRML comprises a set of entities arranged in a hierarchical structure. These entities represent the various objects in the scene and their attributes and are called **nodes**. Nodes of a VRML scene can be included in the VRML file that describes the scene, or can be stored on separate files and referenced by their URL.

Since the core of the VR-SHOP system is the creation of VRML worlds and their distribution over the network, in this section we elaborate on a scheme for mappings between our high-level declarative model and VRML. This mapping enables the 3D interactive visualisation of a room description as defined in our model. Special attention has been paid to the amount of information transferred across the network. This demand leads us to consider reducing the size of the VRML source code using techniques that avoid code repetition for nodes that have similar characteristics. To achieve this, it is essential to have parametric generic objects, which will enable the instantiation of real objects with different size, colour per face, image per face, position and orientation.

Having in mind these requirements, we define a mechanism to parameterise a set of prototype nodes that describe a generic object and create copies of those nodes specifying different parameters for each copy. Intending to repetitively create objects of the same type based on a parameterised data structure, we exploit the ***PROTO node*** feature of VRML 2.0[VRML98]. This feature allows the extension of the built-in VRML2.0 nodes with a set of new nodes, using a combination of the existing ones. The definition of the generic node is stored as a VRML file. The parametric VRML nodes are instantiated in separate VRML files that reference the latter - without having to include the entire definition.

The mechanism allowing many VRML nodes to share the same PROTO files is called EXTERNPROTO. EXTERNPROTO is a VRML primitive that enables prototype nodes' sharing. It operates in the same way as the PROTO, but is designed to use external files for the definition part of the prototype. This allows generic models to be saved in separate files (PROTO files), which can then be referenced by any other VRML file that contains an instance of the model. For example, the generic model of a chair can be defined in a PROTO file. Whenever a VRML world contains a chair, with specific attributes (e.g. dimensions, colour etc) it can simply reference the external PROTO file and modify its parameters. This solution is the most appropriate in our case since it offers both *modular objects* and *reusability* [VRML98]. In the following we will describe how the PROTO node is used on the mapping of objects.

In the VR-SHOP system, the objects that may participate in a scene are mainly electrical appliances and furniture. As a first step, the objects are grouped into categories, depending on their shape (e.g. both the washing machine and the VCR have the shape of a Box). For each category, a VRML file is created, containing the description of the abstract object, using the PROTO definition. A prototype is defined within a VRML file by using the keyword PROTO. The PROTO declaration consists of three parts: the name of the new node type, a declaration of the parameters of this new node, and a definition of the node itself. The definition of the node can consist of any legal VRML2.0 built-in nodes and may also include any previously defined prototypes.

For instance, in the PROTObox.wrl (see APPENDIX A.) we describe a Box -using 6 faces- with its parametric fields (size, position, and orientation, backColor, backTexture, etc.)

The Box's definition corresponds to a bounding box of size 1, 1, 1, centred at the position (0, 0, 0). These design conventions have been made in order to permit control of two aspects: the object's size and the object's position.

We will present an example of a VRML file that describes a cooking top (length= 80 cm., height= 1.20cm., depth= 80cm., colour= black, frontTexture= "cookingtop1599.jpg", upTexture= "platesMK41,3.jpg"), using the PROTOtype VRML file of the BOX. As presented in detail in APPENDIX A, a new VRML file (CookingTop.wrl) is created. The first part of this file contains the PROTO syntax and the second part specifies the parameters that this new node will take. The key difference is that the EXTERNPROTO syntax defines the prototype name and declaration. The actual definition of the prototype is loaded from the specified URL.

The visual result of the above instantiation appears in Figure 2.

Figure 2. The CookingTop.wrl file in the Cosmo Console

The definition of the prototype has been moved to a separate file, which can be included in any VRML file. Following the same methodology we have produced a list of prototype files for each device and furniture category. Some of them are:

- *Electrical appliances*: Refrigerator, Washing Machine, Laundry, Air Conditioner, Microwave, Fan, Cooking top
- *Electronic Appliances*: Mini Hi-Fi, Television, Video TV, Speaker, VCR, DVD, CD Player
- *Furniture*: Room, Couch, Door, Sofa, Window, Desk, Table, Library, 3 types of Chairs, Basket, Lamp, Night table.

Having all these objects in a modular form, we can create our new parametric objects and further on to build dynamically a room. In the next section the authoring procedure for a room is presented.

4 Authoring tools

Since we divided the design phase of the world into two stages, the device definition and their positioning in the world, we decided to provide two tools, one for each stage. The first tool will be used to create the actual appliances based on a few prototype appliances, already defined in the PROTO files. The second tool permits the specification of the room (dimensions, colours etc) and the positioning of the appliances.

4.1 3D appliances specification tool

The specification of the 3D appliances and furniture is performed on an editor, built in Microsoft Visual Basic 6.0. This editor is used to provide a 3D-visualisation for the appliances and furniture that are stored in a database of prototypes. The tool implements a 2-tier architecture in which the

prototype objects and the specific attributes of each parameterised object are stored in the database tier, and the user interface and 3-D visualisation of prototypes and modified objects are implemented in the application tier. This local application uses an ODBC interface to connect to the database.

As already mentioned, the first step of the objects' creation process is the definition of a set of PROTO files that represent the basic object categories (e.g. box, TV-set, etc.). The user creates new appliances or furniture based on these prototype objects and stores their description in a separate VRML file. The derived objects' attributes, along with the location of the generated VRML file, are stored in the database. The editor enables: selection of the prototype appliances or furniture, modification of its characteristics, and creation of a specific device (e.g. a specific TV set model). The VRML file corresponding to this device is stored on the server and a link to it is created in the database. During the design process, the user sets the values of various parameters (dimensions, colour per face, image per face, background colour) and it is possible to preview the device, before saving it.

This application is customised to the needs of a large company that sells electrical and electronic appliances, thus our effort is focused more on the design of prototypes for electrical appliances than for furniture. Therefore we will further refer to this specific tool as "appliances creation tool".

4.2 Room specification tool

A major contribution of the system is the intuitive interface provide for designing 3D configurations using a 2D interface and moreover based on natural spatial relations (such as above, below, next to etc.). Another important contribution is the ability to create 3D preview on the fly during the user's session exploiting the parameters provided. The design and preview capabilities offered by the system are illustrated in the following.

After the device specification, a tool is needed for arranging them in the room. Apart from the placement parameters we need to configure the size parameters of some objects (furniture) or even the colour or surface textures.

The specification of the room is considered as a three steps process: i) definition of the room shape, dimensions and colour, ii) placement of doors and windows and iii) placement of any other object inside the room.

In the current version of the room editor, the room is considered to be rectangular; so only three parameters (length, width and height) are required. In addition, the colour of the walls and the colour of the floor can be chosen from a list of predefined colours. Then the four vertical walls are respectively named as: front, left, back, right so the reference to each wall becomes easier.

For the placement of doors and windows we need to define the wall that the door is on, as well as the distance from the wall next to this door. In the case of the window, the user must also define the distance from the floor. The width and height of doors and windows is also needed.

After the doors and windows are placed, one must define the domestic appliances and furniture of the room. The list of the available objects is retrieved from the database. In the case of furniture, the dimensions (length, width, height) of each object need to be defined. The domestic appliances are assumed to have specific dimensions that cannot be modified. The position of each object is specified using a set of parameters:

- The minimum distance of the object from a pair of adjacent walls.
- The rotation of the object in the vertical axis.
- The position of the object in the vertical axis.

The position of an object in a room is more easily defined by setting the minimum distance from the nearest pair of walls. In this way, one can easily put an object at the back-left corner of the room by selecting 0-distance from both left and back walls.

In order to minimizing the complexity of the placement process, only rotation around the vertical to the floor axis is permitted for objects. Additionally, rotation over other axes is very rare for the case of domestic appliances. The rotation is measured in degrees over the front wall and is positive when the rotation is counter-clockwise.

Finally, in order to define the position of an object in the vertical axis, we define four "level" options: i) on floor, ii) over another object, iii) at a height from floor and iv) on the ceiling. One can choose any of these options. In the second choice the user must also select the underlying object from the list of objects placed in the room. When the object is positioned at a certain distance from the floor, this distance must also be defined.

Although the previously described room model is complete enough to describe a usual room with its contents, the input of all the parameters needed for the description may become tiresome. To avoid this, we added some graphical capabilities in the user interface by forming a 2D drawing space that can be visualized in any WWW browser, see Figure 3. The user can set the room dimensions by dragging a rectangle in a drawing area. The placement of doors, windows and other objects can be done by simply clicking somewhere in the drawing area.

In order to assist the data input process, some parameters are initiated with default values (e.g. the door height is set by default to 2m, but the user can alter this value at any time).

The 2D representation area facilitates the room design since it offers an easy and powerful interface for describing the arrangement of objects in the room. Through the use of the 2D graphical tool room design can be performed much faster with no loss in precision (i.e. parameters revision is always possible).

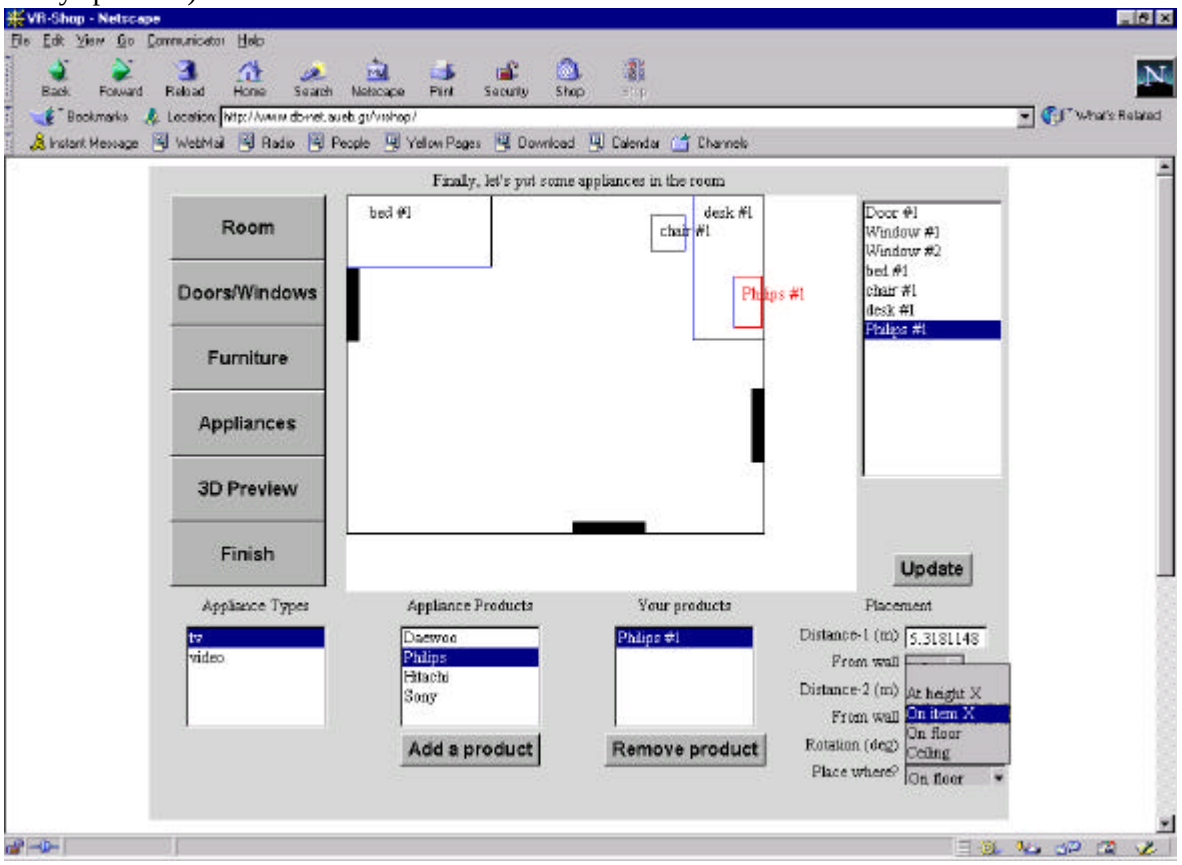


Figure 3. The 2D interface for interactive placement of objects in the room

During the design phase the user is able, ***at any time***, to preview the room's state in 3D VRML representation. This 3D preview is very helpful since it gives to the user the ability to check for any potential design errors, which are not visible in the 2D ground plan of the room. The design errors usually refer to the placement of objects on the vertical axis. Although the various positions of objects in the vertical axis (over an object, on floor, at height X, on ceiling) are denoted using different colours in the 2D plan, the 2D preview can still produce some errors.

5 Dynamic distributed VRML generation

The central idea of the proposed system is the provision of an authoring tool for distributed creation of 3D worlds, which can be used across the WWW. This tool integrates a database that contains information on available appliances. The generation of the VRML scene files can be performed:

- on the user's browser, whenever the user asks for a preview,
- on the server, when the user finishes designing and asks to store the scene locally,
- on the machine that runs the appliances creation tool.

The main modules of our three-tier system are explained in the following.

5.1 Appliances editor

As mentioned above this module aims at the definition of specific appliances based on generic appliances models. The application stands between the user and the database, allowing the user to retrieve or update information in the database. Even though it is not web-based, the application works in a networking environment, thus allowing many editors to remotely connect to the database and create or update appliances' representations.

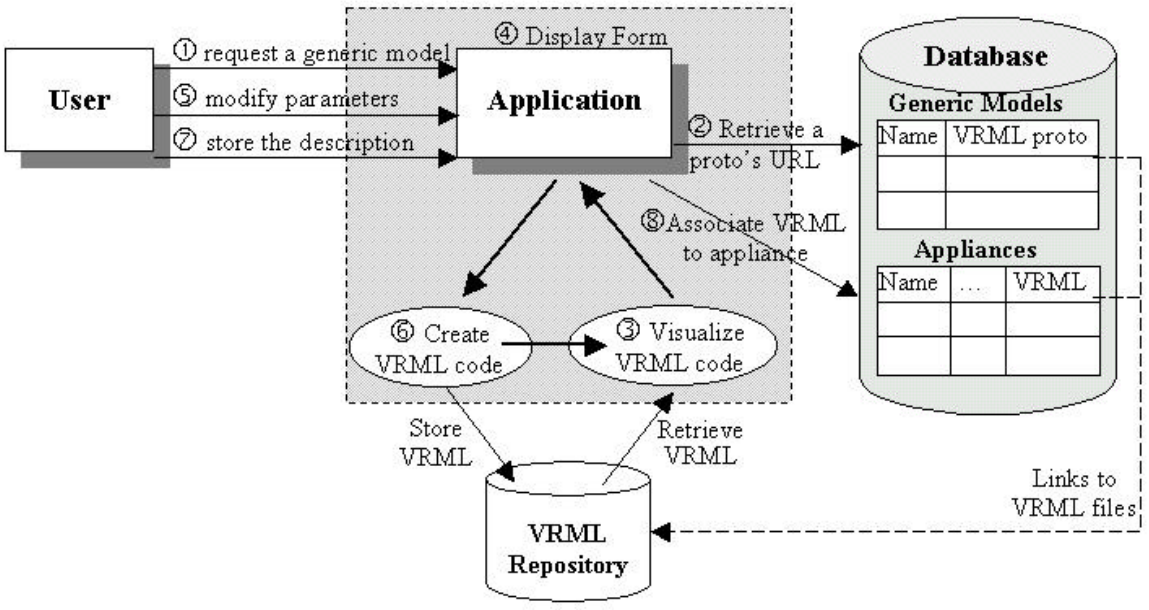


Figure 4. Appliance definition module architecture

Initially, the application displays the generic model selection screen, where all the categories of electrical and electronic appliances are listed. The steps, as depicted in Figure 4, are:

1. The user selects a generic model e.g. VCR
2. The application retrieves the location of the VRML file of the model from the database.
3. The application visualises the generic model (on an embedded VRML browser).
4. The application displays a form with all the modifiable parameters of the generic model.
5. The user gives the dimensions of the object, the colour and texture of each surface.
6. The application refreshes the visualisation of the object each time a parameter is changed.

7. The user chooses to store the specific VCR description on the server, gives a path and filename and associates the description to a specific VCR in the database.
8. The application stores the file on the specified path and updates the database.
9. The application redisplays the generic model selection screen for a new selection.

It must be noticed that the VRML file stored for each appliance contains only a reference to the prototype file instead of the whole file. This minimizes the amount of data stored on the server since for a set of objects of the same type (that use the same prototype) only one copy of the PROTO code and the corresponding references to this are stored on the server.

5.2 Room authoring tool

Similar to the objects' creation tool, the world-authoring tool creates a bridge between the user and the database. The main difference between the appliances creation tool and the world-authoring tool is that the latter is available over the web. This difference introduces new requirements that concern to the amount of data transferred across the network, the security factors that must be considered. Additionally, since this tool is to be used by non-expert users the interface must be simple and easy to use.

A requirement for a web-based application is to limit the amount of data transferred in each transaction. To cover this, in the VR-SHOP system, only the parameters needed for the creation of the VRML code are transferred between the server and client instead of the whole new VRML file. The applet on the client machine has all the logic required to combine the scene objects using the received parameters and compose the complete world.

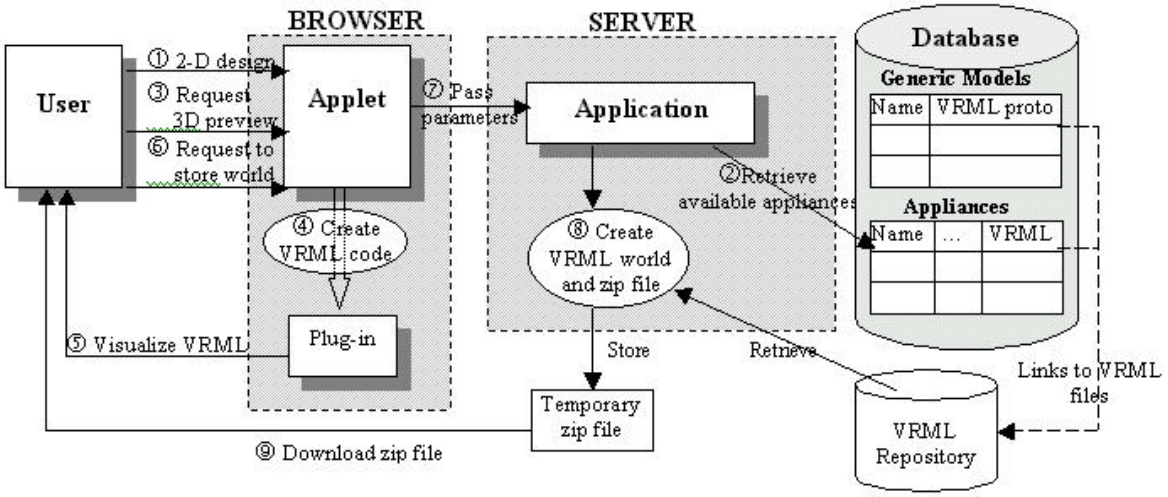


Figure 5. The architecture of the Room authoring tool.

Since the application is used over the web, it is necessary to make it work on a web-browser. Our authoring application requires high-level interactivity with the user, drawing and graphics visualisation capabilities, pre-processing of the user's input and communication with the database through a server application. Plain HTML does not meet these requirements. Therefore, it is necessary to use a language that will move part of the application logic and capabilities on the client side. This is possible using Java applets, or ActiveX components written in Visual Basic or Visual C++. Both applets and ActiveX components are simple programs that can be incorporated into a web page and cover the previously mentioned requirements. Compared to ActiveX components that work only on Windows client machines, Java applets have the advantage of platform independence. They can be executed on any client platform having a Java-enabled web-browser. The use of Java offers an additional advantage since Java provides the interfaces needed to access a VRML browser. The 3D-preview of the room is generated on-the-fly by the Java applet

and the result is displayed via a VRML browser plug-in. This communication between the applet and the VRML browser is performed using a set of Java classes, and will be described in section 5.2.1.

In order to fully exploit the advantages of Java in terms of architecture distribution, a three-tier architecture is used with an applet on the user browser, a Java application running on the server and a database server accessed only by the application. The applet and application communicate using sockets for sending and receiving data and the application connects to the database using the JDBC-ODBC bridge driver. This architecture is the most appropriate for this case since the processing load is shared between the client and the server and the data transfer is limited to only the most essential.

Before explaining in more technical terms the dynamic generation of the VRML world either in the server or the client we describe the way that the authoring tool works, as it is illustrated in Figure 5. The various tasks performed can be divided in two groups, those that refer to the interaction between the user and the application and those that handle the communication between the application and the database.

5.2.1 User-application communication

Interaction between the user and the application occurs in three distinct processes:

- When the user designs the room using both the 2D drawing area and the parameters' specification panel. (Fig. 5, step 1).
- Whenever the user selects to preview on the browser the 3D-visualisation of the room. (Fig. 5, step3)
- When the user decides to store the room visualisation on his/her local machine (Fig. 5, step 6).

In the first case the Java applet draws the user interface and handles the user input. The user can design the room, position the appliances in it and modify their parameters, through the application's interface (see Figure 3). During the design process, the applet requests from the application the list of available appliances, which are stored in the database, and presents them to the user.

In the last two cases the main action is the creation of the VRML file that represents the world. Although both processes concern VRML file creation, they have essential differences that are illustrated in the following.

In the case of the 3D preview, which may happen multiple times during the design process, all the processing is performed on the client machine by the applet. Whenever the user requests a 3D preview, the applet takes all the parameters concerning the room dimensions and the objects positioning and creates a VRML string. This string describes the room and contains references to the absolute URL of the VRML files that represent the objects in the room. The VRML string generated by the applet is passed to the VRML browser using the ***vrml.external*** set of classes and the ***createVrmlFromString*** method [VRML98]. This method takes as an argument the VRML string and sends it to the VRML browser. The result is shown in Figure 6.

The VRML browser we used is SGI's Cosmoplayer V2.1 plug-in. In order to load the VRML plug-in on the browser for the first time, we make a reference to a VRML file that contains an empty world. The following statement must be included into the HTML page:

```
<A NAME="preview">
<embed src = blank.wrl
border=0 width = 700 height = 350>
```

The loaded VRML plug-in is then accessed by the Java applet as described above.

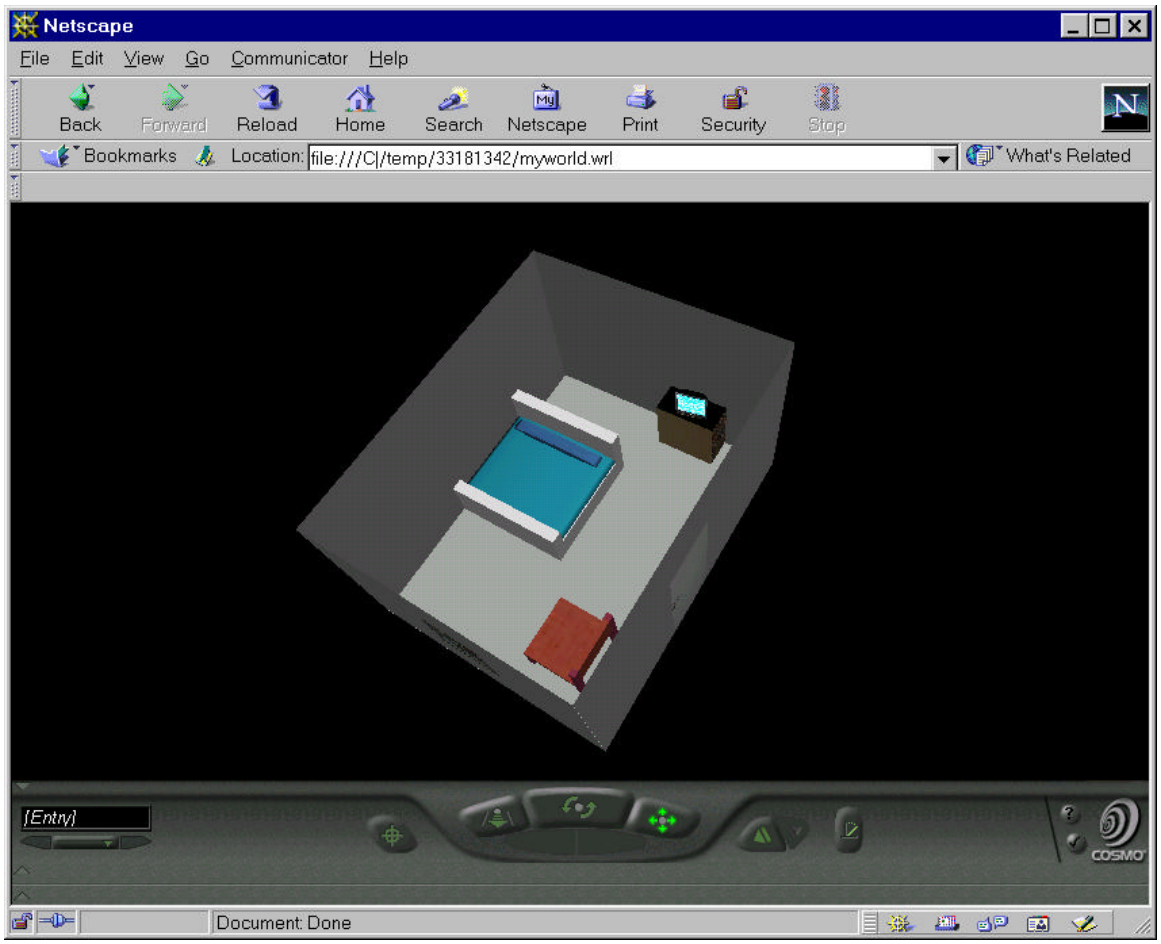


Figure 6. The visualisation of the room during authoring

In the second case, when the user requests to store the world, a file creation and packaging process happens on the server side. This process overcomes many security problems that will be explained in detail in the following section. In opposition to the 3D-preview case, data are transferred between the application and the applet. These data are limited only to a vector of the parameters needed to describe the room and the position of objects. The vector is sent to the server (Fig. 5, step 7) and the VRML file of the world is temporarily created on the server. This file and all the necessary images and PROTOs are compressed in a temporary zip file (Fig. 5, step 8) that the user can download (Fig. 5, step 9). This process happens once, when the user finishes the design of the room, so the amount of data transferred through the whole authoring process is significantly reduced.

5.2.2 Application – Database

Communication between the application and the database occurs when requesting the objects that are available for placement in the room and the links to the respective VRML files (Fig. 5, step 2).

5.2.3 Local storage of the produced world

An essential requirement in the creation of an authoring tool is the ability to store the resulting document. Similarly in our application users should be able to store the resulting world on their machine. Security restrictions of Java [JS01] do not allow applets to write anything to the client's hard disk. In order to overcome this security problem without harming the security rules that web

applications set, we use a secure bypass method that allows the users to download the room description only if they decide to.

With the use of Java's threads, the server application is able to serve multiple clients' requests creating a different processing session for each request. Whenever a user connects to the server (by entering the URL of the page that contains the applet) a new thread of the application starts on the server. When the user finishes the room design and decides to store the final version of the room locally, all the parameters needed for the creation of the VRML file are transmitted through the socket from the client machine to the server. The application that runs on the server creates a VRML file similar to the one listed in Appendix A. The only difference is that the referenced URLs are relative, since the prototype and image files will be in the same directory as the main VRML file and the reference to them will be simply their filename.

In the next step the application stores the main VRML file on the server. This file and the referenced files are packaged in a compressed file. This is done using the ***java.util.zip*** set of classes and the ***ZipOutputStream*** object that permits the creation of a fully compatible zip file. The main VRML file is deleted from the server.

The URL address of the temporary zip file is returned to the applet on the other side of the socket. The applet redirects the user's browser to the zip's URL, using the ***showDocument*** method of the applet class:

```
getAppletContext().showDocument
(new URL(url), "_self" );
```

The effect of this command is that the user is prompted to "open" or "save" the zip file locally. By choosing "save", the zip file that contains the files needed for the VRML representation is downloaded to the client using the HTTP protocol.

When the user disconnects, the server thread that serves this connection deletes the zip file from the server and exits.

6 Related Work

There are several tools (e.g. AC3D), which produce enormous code for simple geometry (using more complex geometries, with thousands of 3Dcoordinates, to define it). SCORE is a distributed object oriented system for 3D real time visualisation [MDG98]. It achieves a reusable design that can be used to develop other virtual 3D buildings where spatial navigation is required. In this work emphasis is given to hypermedia and spatial navigations. However, the system lacks methodology and a clear model for spatial navigation.

In [DiM98], the concept of virtual environment (VE) is introduced. The main feature of this and other similar systems ([CrK94], [DO97], [NB95], [BV95]) is its navigation means, which is spatial, and its interactivity facilities. VEs are described through three fundamental elements: context (for structuring the environment), object (which can be passive, reactive or active and movable or non-movable), and users that can activate objects, collect or move them. Our approach considers only the context element but it supports neither active objects nor avatars. Although this work is not tightly coupled to our approach, it is very inspiring for future work. With the help of the VR-Shop platform, the idea of a virtual store, where clients can walk through its corridors and click on the selves to order for products, becomes feasible. Its up to the creator of the prototype objects to add behaviour or interaction to these objects, which then will be inherited by each specific object.

In [VT98] an approach for specifying virtual worlds is described. Spatiotemporal objects are treated as integrated entities and thus apart from modelling their geometrical and positional

characteristics (in an open and application-independent format), it also demonstrates how to relate them to any non-spatial attributes. Temporal evolution of objects is also taken into account. The requirement for maintaining a continuous view of an object through time is satisfied by linking two successive and explicitly stored states. Then, the state of an object at any point can be extracted by interpolating into the period formed by these states.

Finally, the work of architectural CAD and house decoration software must be mentioned. The differences between our system and other room designing software packages [2020], [KCDw], [VW], can be summarized in the following:

- The VRSHOP system runs over the web and requires only a Java enabled web browser on the client machine. Most designing systems are not available on a web environment, thus cannot be used for an e-commerce application
- The set of available furniture and appliances is not fixed. VRML descriptions for the new appliances can be easily created using the "appliances specification tool". This enables the customisation of the application to the needs of any company that wish to create and maintain VRML presentations of their products.
- The rooms created by the users can be stored locally, but there is also the ability for the company to maintain the room description in the database. This allows users to review and modify their rooms anytime and companies to collect useful data concerning the users' preferences in home decoration.
- Although the VRSHOP system is designed to supply the needs of home appliances selling companies, its architecture is open to modifications. Cloth retailers for example can use the same architecture to promote their products. Clothes of different colours, designs and sizes can replace furniture and appliances and a virtual human body can replace the room. Users of such applications will be able to give their body's measurements, then to choose the clothes they like and finally have a preview of their choices.

Industrial interest for 3D virtual models is indicated by [MVM]. The system is simple enough so that it can be used by non-expert users and is based on parameterised models as models as VRSHOP does. Nevertheless it is still not clear whether virtual models can be widely accepted from wide user communities on the WWW.

7 Conclusions

In the recent past, efforts have been made towards the creation of environments that enable easier access to 3D visualisation. There is a strong demand for frameworks that fulfil these needs even if the area that covers is limited. Our effort could be classified in this category. The distributed authoring model is designed to export code as independent-modular and as simple as it can be.

Attempting to promote the products of a large home appliance vendor, we designed and implemented a system for the WWW enabled interactive design of a room, definition of pieces of furniture as well as placement of domestic appliances. The system allows the company staff to easily create virtual representations for every product in the database and offers the company's web-customers to preview the positioning of the things they intend to buy for their room. The users may define for each participating object the size, the position, and the spatial relationships with the room or other objects. Certain integrity constraints are checked during this definition. The system can create on the fly a VRML representation of the specifications and render it at the client. The user may choose to alter and visualize the world or store the world for further reference.

An important contribution of the VR-Shop system is that it offers to remote users a system for designing their world according to their preferences. This offer is very attractive to web users, since they do not have to buy a complicated and expensive CAD program. The system can be proved very useful for clients that wish to form a better opinion before buying anything and very profitable for companies aiming to promote their products.

The system is based on a 3-tier architecture, where the client is independent from the underlying database scheme and architecture. The database server contains information on the various products, along with links to the VRML descriptions of these products. The application server is responsible for the communication with the database and manipulation of its contents in order to serve the clients' requests. On the client side, a Java applet offers a highly interactive interface and communicates with the application server to provide the user with an on the fly created 3D visualisation of the room based on VRML.

The VRML based room description and all the relevant files needed for the creation of the room (images and VRML files of the various furniture and appliances of the room) are sent to the user in a compressed format. The user may store the compressed file locally. In that way the Java security constraints are overcome safely. The outcome of our work is summarised in the following:

- a declarative authoring model for creating virtual worlds putting emphasis to 3D spatial relationships and related spatial integrity constraints
- a graphical interface that utilizes the authoring model to facilitate the description of virtual scenes, especially virtual rooms,
- the dynamic creation and modification of virtual reality worlds in a distributed environment.
- a generic, platform-independent approach that permits storage both at the server and dient levels without violating the security aspects of Java
- a widely supported format, such as VRML, for describing virtual worlds.

Further work will concentrate on extending the room model towards a generic approach for interactive 3D worlds. This effort will define interactive spatiotemporal scenarios in terms of:

- Complex 3D objects (any complex object can be defined through a set of elementary ones connected via a set of spatial relationships)
- ECA-like rules where E is the triggering event, C the spatiotemporal conditions that must be fulfilled and A the set of actions connected by temporal intervals. The actions will be focused on object appearance and disappearance as well as on the geometric transformations than can be applied to it. Such transformations include: translation, rotation, scaling.
- Visualisation of products that have no physical representation (e.g. interest rates, stock market information, elections results etc). In this case users should be able to choose among different virtual representations available, and to navigate over information showcases.
- Creation of an authoring model and a supporting tool that will allow the quick development of interactive scenarios and worlds. Such extension will cover the needs of 3D animation companies, whose work is based in re-using their objects and actors database in many scenes.

REFERENCES

[2020] 20-20 Design. www.2020office.com

[CrK94] J.F. Cremer, J.K. Kearney, “Scenario authoring for virtual environment”, Proc. Of IMAGE VII Conf., Tucson, AZ, June 12-17 1994.

[DiM99] A. Diaz and R. Melster. "Patterns for modeling behavior in virtual environment applications", Second Workshop on Hypermedia Development: design patterns in hypermedia (in conjunction with Hypertext 99). Darmstadt, Germany, February 1999.

[DoH97] J. Dollner and K. Hinrics, “Object-oriented 3D modeling, animation and interaction”, in The Journal of Vizualization and Computer Animation, vol. 8:33-64, 1997.

[EGSV99] M. Erwig, R. H. Gueting, M. Schneider, M. Vazirgiannis, "Spatio-Temporal Data Types: An Approach to Modeling and Querying Moving Objects in Databases", GeoInformatica Journal, Kluwer Publishers, to appear, 1999.

[JS01] Frequently Asked Questions - Java Security, http://java.sun.com/sfaq/

[KCDw] KCDw kitchen design web site, www.KCDw.com/3d.htm

[MDG98] R. Melster, A. Diaz and B. Groth. "SCORE - The virtual museum, development of a distributed, object-oriented system for 3D real-time visualisation". Technical report 1998-15, TU Berlin, Germany.October 1998.

[MVM] My Virtual Model web site, http://www.mvm.com

[NB95] Marc Najork, Mark Brown , “Obliq-3d: A high Level, Fast Turnaround 3D Animation System”, IEEE Transactions on Visualisation and Computer Graphics, vol 1, no 2, June 1995.

[VRML98] The VRML Repository, available at: http://www.embl-heidelberg.de/vrml/

[VT98] A. Vakaloudis, B. Theodoulidis. "The storage and querying of 3D objects for the dynamic composition of VRML worlds". Chorochronos Workshop, Aalborg, Denmark. June 19 - 20, 1998.

[VW] Virtual Worlds, room planning and visualisation software, www.luk.net/virtual.

APPENDIX A. Sample VRML code.

```
#VRML V2.0 utf8
EXTERNPROTO Room [
field SFVec3f size
field SFVec3f position
field SFRotation orientation
field SFColor backColor
field MFString backTexture
field SFColor leftColor
field MFString leftTexture
field SFColor rightColor
field MFString rightTexture
field SFColor frontColor
field MFString frontTexture
field SFColor floorColor
field MFString floorTexture
] "www.serverpath.PROTOroom.wrl"
EXTERNPROTO Desk [
field SFVec3f size
field SFVec3f position
field SFRotation orientation
field SFColor cupColor
field MFString cupTexture
field SFColor sideColor
field MFString sideTexture
field SFColor frontColor
field MFString frontTexture
] " www.serverpath.PROTOdesk.wrl"

EXTERNPROTO Door [
field SFVec3f size
field SFVec3f position
field SFRotation orientation
field SFColor color
field MFString texture
] " www.serverpath.PROTOdoor.wrl"

Room  {
       size 4 2 4
       floorTexture "www.serverpath.wood.jpg"
       backColor 0.6 0.6 0.7
       frontColor 0.6 0.6 0.7
       leftColor 0.6 0.6 0.7
       rightColor 0.6 0.6 0.7
       }
Desk {
       size 1.5 0.7 0.7
       orientation 0 1 0 0.785
       position -1 -0.65 -1
       sideTexture
" www.serverpath.wood1.jpg"
       cupTexture
" www.serverpath.wood1.jpg"
       frontColor 0 0 0
       }
Door {
       position -2 0 0.5
       orientation 0 1 0 1.57
       size 1 2 1
       texture
" www.serverpath.door.jpg"
       }
```

Example VRML code of a room

```
#VRML V2.0 utf8
#PROTObox.wrl

PROTO BOX [    field SFVec3f size 1 1 1
                     field SFVec3f position 0 0 0
               field SFRotation orientation  0 0 1 0
                field SFColor backColor 1 1 1
                field MFString backTexture []
                field SFColor leftColor 1 1 1
                field MFString leftTexture []
                field SFColor rightColor 1 1 1
                field MFString rightTexture []
                field SFColor frontColor 1 1 1
                field MFString frontTexture []
                field SFColor dnColor 1 1 1
                field MFString dnTexture []
                field SFColor upColor 1 1 1
                field MFString upTexture []
                   ]
{ BOX ‘s definition}
```

PROTOtype file of a modular Box

```
#VRML V2.0 utf8
#CookingTop.wrl

EXTERNPROTO CookingTop [
               [    field SFVec3f size
                      field SFVec3f position
                      field SFRotation orientation
               field SFColor backColor
               field MFString backTexture
               field SFColor leftColor
               field MFString leftTexture
               field SFColor rightColor
               field MFString rightTexture
               field SFColor frontColor
               field MFString frontTexture
               field SFColor dnColor
               field MFString dnTexture
               field SFColor upColor
               field MFString upTexture
               ] "PROTObox.wrl"

Viewpoint{ fieldOfView 0.2}
NavigationInfo{ type "EXAMINE"
speed 1.0
headlight TRUE }
Background { skyColor 0.5 0.5 0.7}
CookingTop {
size 0.8 1.2 0.8
leftColor 0 0 0
rightColor 0 0 0
dnColor 0 0 0
backColor 0 0 0
frontTexture "cookingtop1599.jpg"
upTexture "platesMK41,3.jpg"
      }
```

Instantiating the PROTObox.wrl file

In a sidebar

The evolution of VRML

1994 - VRML 1.0 (http://www.vrml.org/VRML1.0/vrml10c.html)

In 1994, the proto-VRML community selected the Open Inventor ASCII File Format from Silicon Graphics, Inc. as the basis of VRML. The Inventor File Format supports complete descriptions of 3D scenes with geometry, lighting, materials, 3D user interface widgets, and viewers. It has all of the features that developers need to create highly interactive 3D applications, as well as an existing tools base with a wide installed presence. A subset of the Inventor File Format, with extensions to support networking, is the foundation of VRML.

1996 - VRML 2.0 (http://www.vrml.org/VRML2.0/FINAL/)

In 1996, the official VRML 2.0 Specification was released at Siggraph 96 in New Orleans. The VRML Architecture Group finalized the formal announcement of the VRML 2.0 Specification, and requested for a binary format that can be compressed and an external authoring interface to the VRML 2.0 specification, and initiated the formation of the VRML Consortium.

1997 - VRML97 (http://www.vrml.org/technicalinfo/specifications/vrml97/)

VRML allows to compose files together through inclusion and to relate files together through hyperlinks. Hierarchical file inclusion enables the creation of arbitrarily large, dynamic worlds. Therefore, VRML ensures that each file is completely described by the objects contained within it.

2000 - VRML200x (2000) (http://www.vrml.org/TaskGroups/x3d/)

The X3D Task Group is designing and implementing the next-generation Extensible 3D (X3D) Graphics specification. They are expressing the geometry and behaviour capabilities of the Virtual Reality Modelling Language (VRML 97) using the Extensible Mark-up Language (XML).

3D Visualisation Technologies

VRML

VRML is a script language that allows the fast creation of virtual worlds. It is not a standalone language and needs a browser/plug-in (developed in Java3D or Open-GL) to display results.
http://www.vrml.org

Java3D

Java3D offers great capabilities in creating 3D visualisations of objects. It is a powerful platform that can be used even for developing VRML browsers. It may do this through intermediary platforms like Direct3D or OpenGL.
http://www.j3d.org

OpenGL

OpenGL is a cross-platform standard for 3D rendering and 3D hardware acceleration. The software runtime library ships with all Windows, MacOS, Linux and Unix systems.
http://www.opengl.org

Direct 3D

Direct3D is an API from Microsoft Corporation, for providing new software features to developers, so that new and existing features of the PC can be exploited much better than is possible presently. The Direct3D API is part of DirectX and is the component that helps to integrate 3D into Windows applications. Direct3D is used to develop real-time, interactive, 3D applications.
http://www.microsoft.com/directx/default.asp

Related Links

- Web3D Consortium's Web3D Repository: The first and largest database of links to browsers, tutorials, tools. (http://www.web3d.org/vrml/vrml.htm)

- Web3D Round-Up: Top 3D tools and content developers demonstrate latest Web3D technology and applications in a fast-paced, exciting format to merciless, participative audience. (http://www.web3droundup.org/)
- Cosmo Software: The official site to download Cosmo Player. (http://www.cosmosoftware.com/download/)
- Web 3D Consortium related organisations page. Contains info and links to many organisations that practice VRML and offer virtual services (http://www.vrml.org/resources/related.htm)