

# X3D and VRML Sound Components

Mikko Pohja

HUT, Telecommunications Software and Multimedia Laboratory

mikko.pohja@hut.fi

## Abstract

*This paper introduces the VRML and X3D virtual reality APIs and especially their Sound components. The paper discusses the VRML and X3D 3D sound specification in detail and describes what is required from X3D or VRML browser to render the 3D sounds as specified. Finally, there is a short review to VRML and X3D browsers.*

## 1 INTRODUCTION

Briefly, 3-dimensional (3D) audio system can position sounds all round a listener. This is what required when realizing a sound system, which corresponds with real world. The problem with these real world sound systems is that the loudspeakers, which always produce the sounds, cannot surround a listener totally. In stereo systems, we have two speakers. So, the sound can be recognized as coming from somewhere between them, but not anywhere else. By adding speakers, the sound can be placed better, but it will still miss the ability to describe perception of distance, which is important to have a real-world experience. (IASIG, 1998)

The problem is how to get the real-world experience with reasonable amount of speakers. The psychoacoustics researchers have studied this for decades. The main issue is how the binaural human hearing (i.e., hearing with two ears) works. In the real world, one can tell exactly where the sound is located through two signals received by the ears. 3D audio system should be able to produce those two signals for ears to give a listener a feeling of the real-world experience. The easiest way to realize that is to use a set of headphones. So, both ears can be fed with different signals. The real world experience can also be achieved with the loudspeakers, but it then requires more complicated system. (IASIG, 1998)

The term virtual reality refers to an artificial, three-dimensional world that is completely generated by a computer. In addition to having ability to create graphics for virtual

reality, the systems have to render sounds correctly, i.e. they must be able to produce 3D sound. Virtual reality worlds can be defined in various ways. To ease the work of authors, it has been defined numerous Application Programming Interfaces (APIs) through which virtual realities can be defined. This paper introduces how the 3D sound is handled in the two virtual reality APIs.

## **2 VIRTUAL REALITY MARKUP LANGUAGES**

Document markup is the process of adding codes to a document to define the structure of a document or the format in which it is to appear. It is a way of communication that has existed for many years. Before computerization of the world, there were copy editors writing instructions (e.g., font size, etc.) on manuscripts to identify the appearance of the documents. Nowadays, this is done by word processors. (Watson, 1992) This chapter introduces two standards defined for describing virtual reality. Both of them are markup languages.

### **2.1 VRML**

The Virtual Reality Modeling Language (VRML) was a first intention to describe 3D objects and worlds through markup language. It is intended to be a universal interchange format for integrated 3D graphics and multimedia and it is designed to use on the Internet, intranets, and local client systems, too (The VRML Consortium Incorporated, 1997). It is an alternative Application Programming Interface (API) into the 3D programmers.

The first version of VRML, VRML 1.0, was developed between 1994 and 1995. The first version of VRML described only static worlds, where user can move. So, it was obvious that new version would be needed. The VRML 2.0 was introduced at the Siggraph96 conference. The most important new features were sounds and moving objects. (Ashdown and Forestiero, 1998)

In 1997, VRML was standardized by International Organization for Standardization (ISO) and the International Electrotechnical Commission (IEC). The standard is ISO/IEC-14772-1:1997, also known as VRML97. There is no real difference in functionality between VRML 2.0 and VRML97. Changes were made only to the wording and layout of the specification document, to ensure that it satisfied the ISO requirements. (Ashdown and Forestiero, 1998)

## 2.2 X3D

X3D is the successor to the VRML. First of all, X3D is an Extensible Markup Language (XML) unlike the VRML. Being an XML language, it gives an opportunity to parse the X3D documents with the common XML parsers and also combine them with the other XML applications. In addition, X3D is enhanced in various ways compared to the VRML. It has advanced application programmer interfaces, additional data encoding formats, stricter conformance, and a componentized architecture that allows for a modular approach to supporting the standard (X3D Task Group, 2002).

X3D applications are 3D time-based spaces that contain graphic and aural objects that can be loaded over a network and dynamically modified through a variety of mechanisms (e.g., scripts). The semantics of X3D describe an abstract functional behavior of time-based, interactive 3D, multimedia information. On the other hand, X3D does not define physical devices or any other implementation-dependent concepts (e.g., screen resolution and input devices). (X3D Task Group, 2002)

The X3D system architecture is shown in Figure 1. Basically, it is a browser who operates the X3D applications. The browser should be able to parse the X3D application, create nodes from the document, and handle dynamic modifications of the document (e.g., scripting).

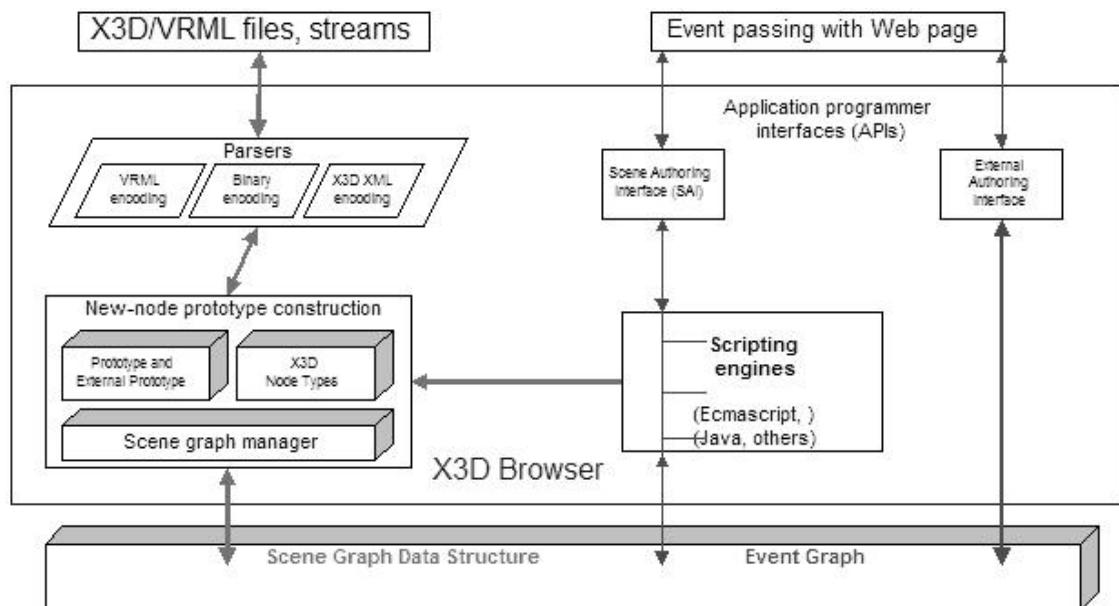


Figure 1. X3D system architecture. (X3D Task Group, 2002)

Always, each X3D application specifies world coordinates for the objects defined and included to the application. Also, they define and compose a set of 2D, 3D, and multimedia objects. In addition, X3D applications can contain hyperlinks to other sources, define behavior of the objects, and connect to the external modules or applications via programming and scripting languages. (X3D Task Group, 2002)

### 3 VRML AND X3D SOUND COMPONENTS

VRML and X3D sound components are basically similar. Although, the component concept is not part of the VRML specification, the functionality of the sound components is the same. Therefore the X3D sound is discussed in detail in this chapter. The specification defines how the browser should render the sound elements defined in the documents.

#### 3.1 Sound priority

Naturally, all the sounds defined in the document are supposed to be played during the presentation. However, it is possible that the browser has not resources to play all the sounds. So, the X3D specification defines which sounds are to be played if something has to be left out. The sounds are ordered according to the following criteria:

1. Decreasing *priority*.
2. For sounds with *priority* > 0.5, increasing ( $\text{now} - \text{startTime}$ ).
3. Decreasing *intensity* at viewer location ( $\text{intensity} \times \text{"intensity attenuation"}$ ).

The *priority* is the *priority* field of the Sound node, *now* represents the current time, *startTime* is the *startTime* field of the audio *source* node specified in the source field, and "intensity attenuation" refers to the intensity multiplier derived from the linear decibel attenuation ramp between inner and outer ellipsoids. (X3D Task Group, 2002)

The sort key 2 is used for high priority sounds in order to get new cues heard even when the browser is using all its resources to currently active high priority sounds. The sort key 3 is applied to sounds with low priority (i.e., *priority* < 0.5). Those sounds are not sorted by sort key 2. (X3D Task Group, 2002)

#### 3.2 Sound attenuation and spatialization

In X3D sound can be attenuated according the perception distance and localized, too. The sound node is rounded by two ellipsoids. Inside the smaller ellipsoid sound is not

attenuated and at the outer ellipsoid sound is attenuated to  $-20$  dB. Between the ellipsoids, attenuation varies linearly from 0 dB to  $-20$  dB. Outside the outer ellipsoid no sound can be heard. (X3D Task Group, 2002)

Basically, in the X3D document can define sound, which is located anywhere in the application's space. However, it is up to X3D browser if the sound can be rendered correctly. If browser is not capable to do that, it is assumed that browser at least support stereo panning of non-MIDI sounds based on the angle between the viewer and the source. The angle is obtained by projecting the Sound *location* in global space onto the XZ space of the viewer. The pan value is angle between Z-axis and the vector from the viewer to the transformed *location* scaled to range  $[0.0, 1.0]$  as shown in Figure 2. By stereo panning, it is given factors for the left and the right channels. The factors are also in the range  $[0.0, 1.0]$  and they can be obtained from the equations (1) and (2). (X3D Task Group, 2002)

$$\text{leftPanFactor} = 1 - \text{pan}^2 \quad (1)$$

$$\text{rightPanFactor} = 1 - (1 - \text{pan})^2 \quad (2)$$

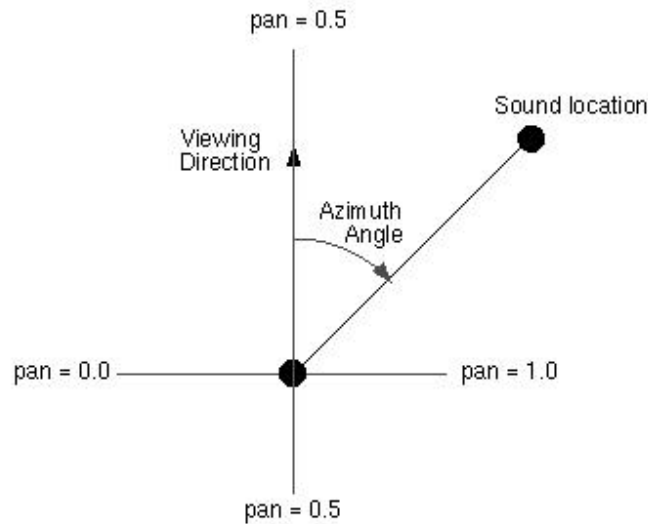


Figure 2. Stereo panning. (X3D Task Group, 2002)

When using stereo panning, the loudness of the sound is first modified according the intensity *field* value, and then distance attenuation. The final left and right output signals are obtained by applying leftPanFactor and rightPanFactor to that signal. (X3D Task Group, 2002)

### 3.3 Nodes

X3D contains two kinds of nodes for the sound. One is used to describe sound in an actual presentation and the other defines sound source. Both types have been derived from abstract nodes called *X3DSoundNode* and *X3DSoundSourceNode*, respectively. All the X3D nodes have fields through which they are described. In below, the sound nodes and their fields are discussed in detail.

#### 3.3.1 *AudioClip*

The *AudioClip* node is derived from the *X3DSoundSourceNode*. It specifies an audio data that can be referenced by the *Sound* nodes. Basically, the data can be in any format but it depends on browser if it can be presented. The browsers are expected to support at least the wave file format in the uncompressed PCM format and the MP3 compressed format. Also, it is recommended to support the MIDI file type 1 sound format. (X3D Task Group, 2002)

The fields of the *AudioClip* node are presented in Table 1. There are two general classes of field types: field types that contain a single value (where a value may be a single number, a vector, or even an image), and field types that contain an ordered list of multiple values. The single-valued field types have names that begin with SF. Multiple-valued field types have names that begin with MF. Multiple-valued fields are written as an ordered list of values. The type is specified after SF or MF. The access column indicates if field can be modified (in) or read (out) or both. Default and possible values are given if they exist. (X3D Task Group, 2002)

In the *description* field is given a textual description of the audio source. The description can be displayed when the sound is played but that is optional. The *url* field specifies the URL from which the sound is loaded.

The *pitch* field defines at which rate the sampled sound is played. The *pitch* value is a multiplier and it shall be greater than zero. It affects both the pitch and playback speed of the sound. For instance, if the *pitch* is set to 2.0, the sound is played one octave higher than normal and played twice as fast. For a sampled sound, the *pitch* alters the sampling rate at which the sound is played. (X3D Task Group, 2002)

A node generates *isActive* TRUE and FALSE events when node becomes active or inactive, respectively. These are the only times at which an *isActive* event is generated. In particular, *isActive* events are not sent at each tick of a simulation. The *isActive* field indicates which event has been fired last. (X3D Task Group, 2002)

The cycle of an *AudioClip* is the length of time in seconds for one playing of the audio at the specified *pitch*. The *AudioClip* node is active from *startTime* either to *stopTime* if

*startTime* is smaller than *stopTime* or when cycle finish if the loop field is FALSE. If the loop field is TRUE, the *AudioClip* is played until the *stopTime* regardless of the length of the cycle. If the *startTime* is equal to or greater than the *stopTime*, the node is active since the *startTime* is equal to or greater than the presentation's current time. (X3D Task Group, 2002)

Table 1. The fields of the *AudioClip* node. (X3D Task Group, 2002)

Field name	Type	Access	Default value	Possible values
description	SFString	[in,out]	""	
loop	SFBool	[in,out]	FALSE	
pitch	SFFloat	[in,out]	1	(0,∞)
startTime	SFTime	[in,out]	0	(-∞,∞)
stopTime	SFTime	[in,out]	0	(-∞,∞)
url	MFString	[in,out]	[]	[urn]
duration_changed	SFTime	[out]		
isActive	SFBool	[out]		

The *duration\_changed* event indicates that the *AudioClip* node's source has changed. Normally, this will only occur when the current *url* in use changes and the sound data has been loaded. The duration is the length of time in seconds for one cycle of the audio for a pitch set to 1.0. Changing the *pitch* field will not send a *duration\_changed* event. A duration value of "-1" implies that the sound data has not yet loaded or the value cannot be fetched for some reason. (X3D Task Group, 2002)

### 3.3.2 Sound

The *Sound* node describes sounds in the X3D worlds. It specifies the sound's location and behavior. The sound can be directed and it is emitted in an elliptical pattern. The pattern is formed by two ellipsoids, which also defines borders for level of loudness of the sound. The shape of the ellipsoids may be modified to provide more or less directional focus from the location of the sound.

The fields of the *Sound* node are presented in Table 2 and explained in more detail below.

The *source* field specifies the sound's source. The source can be either an *AudioClip* or a *MovieTexture* node. The *MovieTexture* node defines a time dependent texture map (contained in a movie file) and parameters for controlling the movie and the texture mapping. However, it can also be used as the source of sound data for a *Sound* node. In this special case, the *MovieTexture* node is not used for rendering and the movie format shall support sound. (X3D Task Group, 2002)

Table 2. The fields of the *Sound* node. (X3D Task Group, 2002)

Field name	Type	Access	Default value	Possible values
direction	SFVec3f	[in,out]	0 0 1	$(-\infty, \infty)$
intensity	SFFloat	[in,out]	1	[0,1]
location	SFVec3f	[in,out]	0 0 0	$(-\infty, \infty)$
maxBack	SFFloat	[in,out]	10	$[0, \infty)$
maxFront	SFFloat	[in,out]	10	$[0, \infty)$
minBack	SFFloat	[in,out]	1	$[0, \infty)$
minFront	SFFloat	[in,out]	1	$[0, \infty)$
priority	SFFloat	[in,out]	0	[0,1]
source	SFNode	[in,out]	NULL	[X3DSoundSourceNode]
spatialize	SFBool	[]	TRUE	

The *intensity* field defines the loudness of the emitted sound. The value of the field varies from 0.0 to 1.0. According to the specification, the *intensity* value is a factor, which scales the sound loudness. So, when set to 1, a *Sound* node emits audio at its maximum loudness (before attenuation), and a *Sound* node with an intensity of 0.0 shall emit no audio. Between these values (0.0 and 1.0), the loudness should increase linearly from a -20 dB change approaching an *intensity* of 0.0 to a 0 dB change at an *intensity* of 1.0. (X3D Task Group, 2002)

The value of the *priority* field is used to decide which sounds are played if some of them have to be left out because of the limits of the system resources or system load. The *priority* can have values between 0.0 and 1.0, with 1.0 being the highest priority. The algorithm to order the *Sound* nodes is presented in detail in Chapter 3.1. (X3D Task Group, 2002)

The *location* field contains the local coordinates of the sound emitter.

The inner and outer ellipsoids, which directs the emitted sound, are defined by the fields *minBack*, *minFront*, *maxBack* and, *maxFront*. The ellipsoids are defined by extending the direction vector through the location. The *minBack* and *maxBack*, and *minFront* and *maxFront* fields specify distances behind and in front of the location along the direction vector respectively. The ellipsoids have one of their focus at the *location* (the second focus is implicit) and intersect the direction vector at *minBack* and *minFront* or *maxBack* and *maxFront*. The organization of the *location*, *direction*, *minBack*, *minFront*, *maxBack* and, *maxFront* fields is shown in Figure 3. There is a controversy in the figure. The graph shows that outside the outer ellipsoid the sound is attenuated  $-20$  dB, but there is also written the text *no sound*. According to the text of the specification, there is no sound outside the outer ellipsoid. So, the graph should end to the outer ellipsoid. (X3D Task Group, 2002)

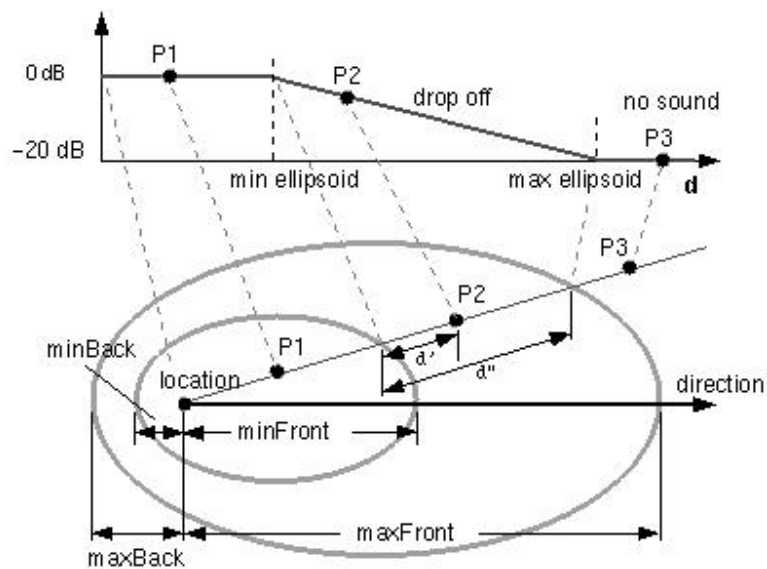


Figure 3. Sound node geometry. (X3D Task Group, 2002)

The values of the *minBack*, *minFront*, *maxBack* and, *maxFront* fields must be equal to or greater than zero. Also, the inner ellipsoid cannot be bigger than the outer one, but they can be at same size. Between the ellipsoids, the sound is attenuated linearly from 0 dB at the minimum ellipsoid to -20 dB at the maximum ellipsoid. The attenuation is obtained from equation (3).

$$\text{attenuation} = -20 (d' / d'') \text{ dB}, \quad (3)$$

where  $d'$  is the distance along the location-to-viewer vector, measured from the transformed minimum ellipsoid boundary to the viewer, and  $d''$  is the distance along the location-to-viewer vector from the transformed minimum ellipsoid boundary to the transformed maximum ellipsoid boundary (c.f., Figure 3). (X3D Task Group, 2002)

In X3D, it can be defined, if sound is heard from some direction or just in general from everywhere. That is specified in the *spatialize* field. If the field has value TRUE and the viewer is between the two ellipsoids, the viewer's and sound source's locations must be notified in rendering. The spatialization is explained in more detail in Chapter 3.2. Even if the *spatialize* field is FALSE, the loudness of the sound is still affected by ellipsoid dimensions and value of the intensity field. Also, if the sound source is multi-channel, it should be played as multi-channel, too.

#### 4 VRML AND X3D BROWSERS

In order to view and interact with a VRML or X3D world, a special browser program is required. There are number of VRML and few X3D browsers currently available. Most VRML browsers operate as a plug-in, where the 3D scene and user interface controls are actually displayed within the main Web browser window. This allows a VRML scene to be embedded within a HTML document. In addition, there are some stand-alone programs, which can operate VRML or X3D. (Ashdown and Forestiero, 1998)

There are several ways to move and explore the VRML and X3D worlds in the browsers. Although the methods of interaction are implemented differently on each browser, they are typically based on examine, fly, walk and click-and-see themes. In examine mode, the user can rotate an object or move it in relation to the viewpoint. A fly mode simulates moving through the scene, with mouse or keyboard input controlling speed and direction. Walk mode is similar to fly mode except the user's viewpoint will follow the terrain. Finally, some browsers implement a seek mode, where the user can click on an object with the mouse pointer and the viewpoint will move automatically towards it. The browsers also usually allow the user to switch between the viewpoints defined within the document. (Ashdown and Forestiero, 1998)

Two other common features in the browsers are kind of headlamp effect in which the 3D world is illuminated in front of the viewpoint and a method to allow the user to select an

object, usually by clicking on it with the mouse pointer, in order to follow a hyperlink or to activate a sensor. (Ashdown and Forestiero, 1998)

Although there are several browsers available, only few of them support sounds. Actually, there is currently not any X3D browser, which has support for the X3D Sound component. Some of the VRML browsers have support for the 3D sounds. The VRML browsers supporting VMRL 97 include:

- Cosmo Player (<http://ca.com/cosmo/html/player.htm>). Originally developed by SGI, now owned by Platinum. Version 2.1 is available for Windows and PowerMac machines.
- VRwave (<http://www.icm.edu/vrwave>) - the successor to VRWeb. This browser is being developed in JAVA at Graz University, Austria. Full source code for the browser is available.

The X3D documents can be transformed to the VRML documents through the Extensible Stylesheet Language Transformations (XSLT) document. That way the X3D documents with the 3D sound can be presented at the moment. Of course, the transformation requires software of its own, which is not normally included into the VRML browsers. So, the transformation has to be done beforehand by the user itself.

## 5 CONCLUSIONS

In theory, one can define a sound coming from any direction and any distance in X3D and in VRML. It is up to the browser, if it is rendered correctly. Referring to that, it can be said that the both languages can really be used to describe virtual realities what comes to the 3D sound.

However, there is one thing in the specifications, which may raise discussion. According to the specification, the sound is attenuated between the two ellipsoids from 0 dB to -20 dB. And, outside the outer ellipsoid, the sound cannot be heard at all. That obviously makes a step to the loudness in the boundary of the outer ellipsoid and is not definitely like in the real world. The authors should take this feature in account when defining the *Sound* nodes. If the sound is very loud, the above feature could be avoided by defining the outer ellipsoid bigger than the viewer is expected to explore in the 3D scene. That way he or she never meets the step in the loudness at the boundary.

The lack of the 3D sound support in X3D browsers obviously limits the use of X3D. Although, the documents can be transformed to VRML documents, the software for the transformation is always needed. That is not the problem for the advanced users, but will most likely limit X3D to be more common 3D sound API. However, the X3D browsers are developed all the time, so this absent should have been fixed in a future.

## REFERENCES

Ashdown, N. and Forestiero, S. 1998. "A Guide to VRML 2.0 and an Evaluation of VRML Modelling Tools," <http://www.agocg.ac.uk/train/vrml2rep/cover.htm> [Referenced 22.1.2003]

Interactive Audio Special Interest Group (IASIG), 1998. *3D Audio Rendering and Evaluation Guidelines*. <http://www.iasig.org/wg/closed/3dwg/3dl1v1.pdf> [Referenced 22.1.2003]

The VRML Consortium Incorporated, 1997. *The Virtual Reality Modeling Language International Standard ISO/IEC 14772-1:1997*. <http://www.web3d.org/Specifications/VRML97/index.html> [Referenced 22.1.2003]

Watson, D. G. 1992. "Brief History of Document Markup," [http://edis.ifas.ufl.edu/BODY\\_AE038](http://edis.ifas.ufl.edu/BODY_AE038) [Referenced 22.1.2003]

X3D Task Group, 2002. *Extensible 3D (X3D) International Standard ISO/IEC 19775:200x* [http://www.web3d.org/TaskGroups/x3d/X3DSpec\\_CD\\_Preview/index.html](http://www.web3d.org/TaskGroups/x3d/X3DSpec_CD_Preview/index.html) [Referenced 22.1.2003]