X3D® (Extensible 3D) Frequently Asked Questions (FAQ)

Version 1.17
Last modified: Thurs Aug 9, 03:20:01 PDT 2001

This document contains answers to a number of frequently asked questions about the Web3D technology known as **X3D**tm. The contents of this FAQ will evolve over time as the technology evolves, and as more questions are asked! If you have an entry that you would like to add, then please e-mail Martin Reddy. If you are not already familiar with the Virtual Reality Modeling Language (VRML), then you may also find Bob Crispen's excellent comp.lang.vrml FAQ a useful resource.

The Japanese version of the X3D FAQ, translated by Oga-san, is available from http://www.janit.com/, or directly from http://www.janit.com/netbranch/VirtualReality/faq.html.

The Korean version of the X3D FAQ, translated by Nexternet, Inc., is available from http://www.web3d.co.kr/, or directly from http://www.web3d.co.kr/x3d/faq/.

The Spanish version of the X3D FAQ, translated by Roberto G. Puentes Diaz, is available from http://www.confis.com.ar/x3d/(X3D)FAQes(1-17).htm.

Contents

Introduction

X3D is the next-generation open standard for 3D on the web. It is the result of several years development by the Web 3D Consortiums' X3D Task Group and the recently-formed Browser Working group. The BWG has worked closely with the X3D Task Group to create a new X3D Specification that meets both the Browser companies needs and the needs of the community at large. These requirements are:

- Compatability with existing VRML content, browsers, and tools.
- Extension mechanism to permit introduction of new features, quick review of advancements, and formal adoption of these extensions into the specification.
- Small, simple "core" profile for widest-possible adoption of X3D support, both importing and exporting.
- Larger, full-VRML profile to support existing rich content.
- Support for other encodings including XML for tight integration with Web technologies and tools.

- Architecture and process to advance the specification and technology rapidly!

These requirements were achieved by introducing a component-based architecture to support addressing extensions, incompatabilities, and bugs, and encodings as separate issues. A component represents a grouping of related features, such as a collection of related nodes, an extension to the event model, or new scripting support.

Instead of a large, monolithic specification which requires full adoption for compliance, a component-based architecture supports creation of different "profiles" which can be individually supported. These profiles are collections of components, and two examples of profiles are the small "core" profile to support simple non-interactive animation, and the "base" VRML-compatible profile to support fully-interactive worlds. Components can be individually extended or modified through adding new "levels", or new components can be added to introduce new features, such as streaming. Through this mechanism, advancements of the specification can move quickly because development in one area doesn't slow the specification as a whole.

General Overview

1. **What is X3D?**

   X3D is the next-generation open standard for 3D on the web. It is an extensible standard that can easily be supported by content creation tools, proprietary browsers, and other 3D applications, both for importing and exporting. It replaces VRML, but also provides compatability with existing VRML content and browsers. Existing VRML content will be played without modification in any X3D-2 browser, and new X3D-1 and X3D-2 content can be read in to existing VRML applications.

   X3D addresses the limitations of VRML. It is fully specified, so content will be fully compatable. It is extensibile, which means X3D can be used to make a small, efficient 3D animation player, or can be used to support the latest streaming or rendering extensions. It supports multiple encodings and APIs, so it can easily be integrated with Web browsers through XML or with other applications. In addition to close ties with XML, X3D is the technology behind MPEG-4's 3D support.

2. **What Does X3D look like?**

   The new spec is being finished and will be available for review soon on Web3D.org . Hopefully then people will have a better understanding of what is going on. Until then, here is a brief description, and a link to a working copy of the new spec index page:
   :http://www.openworlds.com/x3d/new/index.html.

   In simplest terms, X3D is VRML 97 broken down into components, with a mechansism to add new components to extend beyond VRML 97 functionality. X3D looks just like VRML. To turn a VRML file into an X3D file, you add the following comment lines:
   #X3D profile:base

   if your content has features which aren't standard VRML, you add a line like:
   #X3D component:streaming:1
   This tells the browser that this content requires the streaming functionality, level 1. This might be a collection of nodes to support streaming, or might be an API-level facility. If it is a

collection of nodes, this might trigger the browser to load in a world file which contains the EXTERNPROTO declarations of these nodes.

So that people creating content don't have to worry about listing or including dozens of components, Profiles are created which consist of many components. In this manner, you can specify one profile which can have enhancements in several functional areas. For example, the Base profile includes both new components (PROTO, Audio, etc.) and new levels of existing components (i.e. the Box node in the geometry component) over the Core Profile, but you only specify the profile, not the list of components; for example, '#X3D profile:base'

As browsers advance, components will be adopted into new profiles, so that the next browser profile may include components for NURBS, streaming, etc. This is the basic architecture.

Now because it is hard to fully import VRML, we wanted to make it easy for companies to import and export some level of X3D. This is why VRML has been grouped in components and profiles. Components group nodes or functionality, for example, the geometry component groups the VRML geometry nodes. Components have different levels, so geometry level 1 doesn't contain the Box node, but level 2 does, etc. As new geometry node types are added, new levels to that component get added.

A profile is a collection of components, so the core profile (X3D-1) consists of level-1 components which support geometry and animation. X3D-2 is the VRML97 profile which supports all of VRML 97 nodes plus the additional functionality of PROTOs and Scripts.

A company that makes an X3D-1 product knows that it can import content that is X3D-1 compliant, and that content it generates can be read in by X3D-1, X3D-2, and VRML97 browsers.

Notice we haven't mentioned XML. That is because XML support is not requried. Current VRML97 browsers are immediately X3D-2 compliant. This is a basic requirement of the specification. XML is an additional encoding, just like a binary encoding. XML encoding and related APIs are a powerful mechanism to integrate X3D with other Web-based technologies, and much work has been done in this area by the Task Group to insure that X3D will be supported by XML tools. Translators will also be available to translate content between encodings. Because of the scope of the encodings issues, encodings have been moved into a separate document.

In summary, all VRML content and tools will work right off the shelf with X3D. Plus X3D will now have a way to have non-VRML97 features such as Nurbs and GeoVRML supported as new native nodes in all browsers within the scope of the specification rather than as a proprietary extension. X3D also give a way of many companies easily supporting importing and exporting of X3D to whatever level they can, and making sure that they support it well instead of having spotty support. And it give a way of companies creating small, efficient X3D browsers which don't need the level of functionality that VRML provides, ala Shout3D. Foremost, it gives a way of the VRML 97 browser companies to extend their current VRML 97 browsers with new features that can easily and QUICKLY be incorporated into the spec rather than have them stay as proprietary extensions. And the optional XML encodings and support provide a mechanism for tight integration with other Web technologies.

3. **VRML is extensible too. Why is X3D more extensible than VRML's EXTERNPROTO mechanism?**

A component can contain many nodes (i.e. the Nurbs profile contains all of the related nurbs nodes). Also, a component can add other areas of functionality, such as a new scripting language support, or user-interface requirements, etc. A component can also simply be a collection of externprotos.

VRML has only the Externproto mechanism for extensibility, but no real mechanism for creating groups of functionality extensions. X3D's component, level, and profile mechanisms allows for this. And while the individual browsers can implement profiles by using protos and externprotos, it is not forced on the browser companies to do this.

Plus, components can be in terms of more than just nodes. It can be for entire functional areas. For example, we might decide we need in-line ECMAScript within the X3D file at some point. The component mechanism permits this kind of extension.

4. **What are X3D Components, Levels, and Profiles?**

   For X3D to truly be an industry-wide standard, the designers realized that different companies don't need or want to support every feature that X3D has to offer. For example, if a company wants to make a small, effecient 3D animation engine, it might not be interested in supporting features for geology rendering. Because of this, groups of features are encapsulated in what are called "components". A component is usally specific for one particular area of functionality (i.e. a "Geo" component for handling geographic data, or a "geometry" component containing a group of geometry "nodes", or a scripting component introducing the concept of script support).

   While components provide a means to introduce a collection of new nodes, X3D still supports extensions through Externprotos, protos, scripts, etc. In fact, component support can be implemented through use of these features.

   A profile is a grouping of components covering several different areas of functionality (i.e. a "Full" profile which handles all VRML97 nodes and functional areas). A profile can even contain the functionality of several profiles (i.e. the "Full" profile includes the functionality of the smaller geometry-based "Core" profile).

   Once a group of profiles is deemed important for inclusion across many applications, a new version of X3D can be created which includes by default a set of profiles. A new Version implies more functionality than a lower Version number.

   Companies can create X3D browsers, tools, importers, and exporters which support different Versions, profiles, and components. For example, a small player might be X3D-1 compatible. A full VRML97-compliant browser would be X3D-2 compatible. X3D-3 might include extra areas of functionality including NURBS, streaming, etc.

5. **How are new components and profiles created?**

   Companies can create new Components which their product supports and submit them to the X3D Board for approval. When a component is submitted, it contains a prefix for the company submitting the component, similar to how OpenGL extensions have a prefix for the company which created the extension (i.e. OW_) Components will undergo testing and review by the X3D board, the Web3D Consortium, and the community at large.

   Once a component is accepted and implemented by more than one company, the prefix

changes to EXT_. If the component is ratified by the board, it then gets the prefix X3D_.

The Board can deem that certain components are so widely adopted and important that they should be included in a new profile.

6. **Do companies need to support all of the Profiles, Components, and Levels?**

   No. Profiles and components exist so that companies need only support which profiles and componetns that suits their needs. By having profiles, their products can be sure that content they read will work in their application, and that content that they create will work in other applications that support their components or profiles.

   Many companies would not want to support a large, complex specification like VRML97. But X3D's modular structure means that they can start off by supporting X3D-1, and gradually add additional profiles as they see fit.

7. **Won't it be a problem having companies making dozens of components?**

   No. The process of having components adopted into the X3D specification provides the mechanism to keep X3D-compliant applications working together. Many new features will fall under existing components, thereby introducing new levels for those components.

   By having companies be able to develop components and submit them, X3D leverages industry-wide advances quickly and efficiently. It also guarantees that X3D grows and flourishes, and does not become technically obsolete as prior standards have become.

8. **Why should my company support X3D?**

   Supporting X3D gives many advantages for a company:

   - Foremost, even if your product uses a proprietary format, supporting X3D instantly gives you access to more tools, content, and compatability with other applications, all will minimal effort. You even get the best of both worlds, your own format PLUS industry compatibility!
   - Your product will benefit by having a competetive marketing advantage by being able to claim "X3D Compatabile!", and this will additionally provide an easy path to leverage industry-wide developments in X3D.
   - There are significant commercial and open-source movements for advancing X3D. This provides a path for your application not having to "reinvent the wheel" everytime new advances in the industry are made.]
   - X3D compatibility is easy! X3D-1 is simple to implement.
   - By supporing X3D, your company helps foster growth of the 3D industry as a whole! X3D acts as a unifying platform and unifying marketing banner under which the entire industry can grow.
   - X3D content is modular and reusable, saving development time and money.
   - X3D leverages VRML content, exporting, and tools. Packages exporting VRML such as 3DSMax already are X3D compatible.
   - X3D supports optional XML encodings for tight integration with other Web technologies.
   - X3D support also provides a path for MPEG-4 support. X3D-1 is the basis for MPEG-4's 3D rendering.
   - Because it is extensible and modular, a browser can support only the profiles it needs, so

companies can make small, efficient browsers to meet their individual needs.

Technical Overview

1. **Why bother with Extensible Markup Language (XML)?**

   XML was adopted as a syntax for X3D in order to solve a number of real problems:

   1. **Rehostability** - VRML 97 syntax is foreign to all but the VRML community. It is similar to the Open Inventor scene graph syntax on which it is based, and to some object notations. Nevertheless the dominant syntax in world-wide use is XML. Markup has proved to be the best solution to the long life-cycle problems of data archival and rehosting.

   2. **Page integration** - XML page-based integration goes directly to the problem of keeping the system simpler so more people can develop for web pages, in both content and implementation.

   3. **Integration with the next-generation web** - The members of the World Wide Web Consortium (W3C) are putting a lot of effort into the development of XML. Extensive XML support is expected in the version 6 releases Netscape Communicator and Internet Explorer. It looks like XML is going to be here for a while and so we need to get on the bandwagon to enable tighter integration with next-generation web technologies.

2. **Is there an accepted Document Type Definition (DTD)?**

   The X3D Task Group spent a number of months looking at various alternatives for representing the VMRL97 nodes in XML. This deliberation resulted in a single DTD (Document Type Definition) which has been agreed upon by the group and can be found from http://www.web3d.org/TaskGroups/x3d/translation/x3d-compromise.dtd.

3. **What about other XML-related technologies?**

   The X3D Task Group is looking at various other W3C standards that relate to XML. Technologies that are of particular interest to the X3D effort include: SMIL (Synchronized Multimedia), SVG (Scalable Vector Graphics), XHTML (the W3C's XML-ization of HTML), Schema, (enables complex datatypes in XML) and DOM (Document Object Model). The Web3D Consortium is a member of the W3C and by this virtue the X3D Task Group is able to closely monitor the progress in each of these areas, assessing the best way to proceed in each case.

4. **What tools are there to support X3D?**

   There are a number of useful tools related to X3D, including stylesheet conversions between VRML97 and XML-based editing suites for X3D files. Don Brutzman has put together a diagram showing the many of the available XML tools and their relationships for X3D. This PDF document can be found here.

5. **How can XML markup be integrated with a web browser?**

   Once you have created a set of XML tags then, depending how the browser uses XML, you

might incorporate these through one of the following schemes:

1. **Stylesheet**: The XML file is used directly. Internally, it will have a reference to a stylesheet language file such as XSL. The XSL file has instructions for displaying the XML tags.

2. **Data islands**: Values of XML are bound into another file (e.g., HTML) to named tags such as HTML tables, divs, etc. This means the XML will be included by value or reference into the document, and the values inside the tags are used by HTML tags. This is a Microsoft specific solution.

   (NOTE: 1 and 2 are the solutions sometimes referred to as page integration.)

3. **Support by plugin**: This is the form where an object tag is used inline to indicate where supporting code can be found, parameters to pass, etc. In addition, Netscape 6 is expected to provide support for "pluglets", which will offer tighter integration with the Web browser.

4. **Direct object support**: The Web browser provides native support for the tags by augmenting its object model to specifically handle these. Obviously, unless the browser is extensible by authors, the tags and their implementation will be hardcoded into the browser.

6. **How many three letter acronyms will I need to understand in order to use X3D?**

   Or alternatively: will X3D authors need to know how to write their own DTDs (Document Type Definitions), CSS (Cascading Stylesheets), XSL (Extensible Stylesheets), etc.? The short answer is no.

   All of this work is a responsibility of the X3D Task Group. The X3D specification defines a DTD for X3D that will have a one to one correspondence with VRML 97 nodes and fields. Authors use these defined tags and hence do not need to develop their own DTDs. Translators are being constructed to convert VRML files to X3D files so that any VRML modeling tools can continue to be used. Exemplar open-source software for parsing/importing/exporting X3D will be provided to encourage 3D toolmakers to easily add X3D import/export next to their VRML import/export.

7. **Where can I find out more about XML?**

   Here are few useful resources that should get you started:

   1. http://www.w3.org/XML/
   2. http://www.xml.com/
   3. http://www.oasis-open.org/cover/xml.html
   4. http://xml.about.com/

8. **What is componentization good for?**

   The following list enumerates some of the benefits that form part of the drive behind the componentization effort:

   1. **Small, lightweight core** - VRML 97 is a large and complex standard to implement. By

stripping VRML down to a small core set of functionality we make it easier for developers to implement X3D, reduce the complexity of the implementation, and hence improve the maintainability of this implementation.

2. **Extensibility** - Through the notion of extensions and profiles, it is possible to build added functionality on top of the Core. This enables new features to be easily added, or existing features to be replaced with alternative extensions.

3. **Re-engaging the Web3D Working Groups** - An extensible architecture allows the results of VRML Working Groups to be implemented on top of the Core browser. For example, there might be an H-Anim, EAI, Living Worlds, or GeoVRML extension.

4. **Reduced footprint** - This is useful, for example, in the space of set-top boxes where each feature has a cost. Here, the ability to use profiles to enable a browser with a smaller footprint (and corresponding smaller set of functionality) is an absolute requirement.

N.B. there are certainly a number of technology issues to be resolved here. For example, extensions will, in general, need access to the core and the underlying OS. This means that extensions will not generally be compatible with one another or any given core implementation. These are open issues which the X3D Browser Working Group and X3D Task Group are be looking into.

9. **Is componentization feasible?**

There are many complex issues surrounding the implementation of cross-platform, cross-browser componentization schemes. Here are some factors which appear to make componentization feasible for the X3D effort.

- Textbooks such as Szyperski, Clemens; "Component Software: Beyond Object-Oriented Programming"; Addison Wesley; 11/1997; 411 pages.
- The Bamboo project at NPS has shown that componentization is possible cross-platform and cross-OS even for C++ dynamically linked binaries. See http://www.npsnet.nps.navy.mil/npsnet/Bamboo.html.
- The Mozilla open source for Netscape is implementing a cross-platform COM which might also serve as a componentization mechanism (but no one has stepped up as a working coder/expert on this topic yet).
- A core kernel implementation already in place can certainly choose to install it's own properly compatible components/extensions.
- Some full-up VRML 97 implementations may choose to remain monolithic, still complying with VRML 97 profile (but also requiring full install).
- Use of a Document Type Definition to specify the XML tagset might also specify a tag or attribute mechanism whereby content authors can point to component implementations available to render newly defined tags. in other words: we can first use XML to define syntax for new nodes like Torus, and also let that XML include pointers to componentized Torus implementations.
- Conventions for defining and supporting and automating such mechanisms are very valuable and best developed by an open, diverse working group.
- Existence of Java3D open source and soon-to-arrive blaxxun C++ community source will let people try componentization schemes out in the light of day, so we all might learn and

improve.

10. **Is it true that Box, ElevationGrid, etc. nodes are being eliminated?**

    No, that is not true. X3D is not reducing functionality: it is partitioning it. It is true that these nodes will likely not be in the Core, but they will be implemented in an extension, such as the VRML 97 extension. The rationale is that Box, Sphere, Cone, ElevationGrid, etc. should not be included in the Core because they can easily be derived from lower-level primitives such as the IndexedFaceSet. This simplifies the core, making it smaller, easier to implement, and easier to maintain.

11. **Is X3D really just VRMLLite?**

    No. The notion of a stripped down version of VRML, affectionately termed VRMLLite, was discussed for a while on the VRML mailing list. The idea was to reduce the bloat in current browsers by stripping away non essential functionality. X3D does not strip any functionality - the notion of a Core is perhaps similar to the concept of VRMLLite, but the whole idea of the X3D Core is that it can be extended to prove further functionality.

12. **What nodes are in the Core?**

    A 3-day retreat was organized over 27-29 September 1999 to decide the subset of nodes that should appear in the X3D Core. Notes from this retreat can be found here. The list has been augmented slightly since then. The current subset is described in Table 9.2 (Nodes for conforming to the core profile) of the X3D Specification.

13. **What are X3D Versions and Profiles?**

    For X3D to truly be an industry-wide standard, the designers realized that different companies don't need or want to support every feature that X3D has to offer. For example, if a company wants to make a small, effecient 3D animation engine, it might not be interested in supporting features for geology rendering. Because of this, groups of features are encapsulated in what are called "profiles". A profile may be specific for one particular area of functionality (i.e. a "Geo" profile for handling geographic data), or can be more general covering several different areas of functionality (i.e. a "Full" profile which handles all VRML97 nodes). A profile can even contain the functionality of several profiles (i.e. the "Full" profile includes the functionality of the smaller geometry-based "Core" profile).

    Once a group of profiles is deemed important for inclusion across many applications, a new version of X3D can be created which includes by default a set of profiles. A new Version implies more functionality than a lower Version number.

    Companies can create X3D browsers, tools, importers, and exporters which support different Versions and Profiles. For example, a small player might be X3D-1 compatible. A full VRML97-compliant browser would be X3D-2 compatible. X3D-3 might include extra areas of functionality including NURBS, streaming, etc.

14. **How are new Profiles and Versions created?**

    Companies can create new Profiles which their product supports and submit them to the X3D Board for approval. When a profile is submitted, it contains a prefix for the company submitting

the profile, similar to how OpenGL extensions have a prefix for the company which created the extension (i.e. OW_) Profiles will undergo testing and review by the X3D board, the Web3D Consortium, and the community at large.

Once a profile is accepted and implemented by more than one company, the prefix changes to EXT_. If the profile is ratified by the board, it then gets the prefix X3D_.

The board can deem that certain profiles are so widely adopted and important that they should be included in the next version of X3D.

15. **Do companies need to support all of the Profiles and Versions?**

    No. Profiles and Versions exist so that companies need only support which profiles and versions that suits their needs. By having profiles, their products can be sure that content they read will work in their application, and that content that they create will work in other applications that support their Version or Profiles.

    Many companies would not want to support a large, complex specification like VRML97. But X3D's modular structure means that they can start off by supporting X3D-1, and gradually add additional profiles as they see fit.

16. **Won't it be a problem having companies making dozens of profiles?**

    No. The process of having profiles adopted into the X3D specification provides the mechanism to keep X3D-compliant applications working together.

    By having companies be able to develop profiles and submit them, X3D leverages industry-wide advances quickly and efficiently. It also guarantees that X3D grows and flourishes, and does not become technically obsolete as prior standards have become.

17. **What is an X3D Node Type?**

    A Node Type is an abstract node base for nodes with related functionality. Figure Object Hierarchy, shows the relationship and derivations of each node based on its abstract node type (names ending in "Node" in Object Hierarchy).

    - Example - GeometryNode, LightNodes, SensorNode, BindableNode, etc. as defined in the VRML 97 specification and reiterated in the X3D DTD.
    - Example - new Tetrahedron node definition would be derived from a GeometryNode type, since it can go anywhere in scene graph that a Box, Sphere or other geometry node can go.

18. **Have any profiles been defined?**

    Yes. Don Brutzman has produced a DTD for a GeoVRML Profile. This is based upon the 10 nodes of the GeoVRML 1.0 Specification.

19. **Is X3D written in Java?**

    No. X3D is a 3D scene graph specification. This specification can be implemented in any of a number of languages, including C, C++, Java, etc. As part of its deliverables, the X3D Task

Group intends to deliver the X3D specification along with two openly available sample implementations. Currently it looks like these sample implementations will be written in Java using Java3D open source, and in C++ using Blaxxun's Contact community source. Other implementations may include Shout (pure Java) and DRaW (FSG). This does not mean that X3D need only be written in Java or C++, or that X3D is restricted to being implemented as a Java applet.

20. **Will my VRML 97 browser work with X3D content?**

    X3D will use XML as a syntax for its file format. A standard VRML 97 browser will not have an XML parser and so will probably not generally work with X3D content. However, file converters can be written to convert an X3D file into a VRML 97 file for browsing in a VRML 97 browser; just as VRML 1.0 files are currently converted to VRML 97 files for browsing. Preparsers and translators are also likely to be produced which could serve as front ends to existing VRML 97 browsers.

21. **Will my VRML 97 content work in an X3D browser?**

    Yes: it is a requirement of X3D to be backward compatible with VRML 97 content.

22. **How does Fahrenheit relate to X3D?**

    The Fahrenheit architecture comprises three API layers: Fahrenheit Low Level (FLL), Fahrenheit Scene Graph (FSG), and Fahrenheit Large Model Visualization (FLM). X3D browsers could potentially use the Fahrenheit API as their rendering layer, just as current VRML 97 browsers normally use OpenGL or Direct3D for rendering. For example, OpenWorlds has an implementation of their VRML 97 browser using Fahrenheit and they intend to produce an X3D implementation using Fahrenheit also.

23. **How does XGL relate to X3D?**

    XGL is XML-based file format for representing 3D information based upon the OpenGL rendering library (http://www.xglspec.org/). The similarity with X3D is that they both use an XML format and are used to encode 3D data. However, XGL is aimed at a much lower level, representing only things like geometry, surface properties, and lighting. X3D is aimed at a much higher level than the raw geometry because it supports an event model, scripting, sensors, animations, and extensibility. The fact that XGL has adopted XML adds validation to X3D's choice to develop an XML format. Also, by adopting XML, X3D has the potential to support other XML languages through the use of XSLT, such as schema for weather, molecular, and 2D graphics data.

Process and Status

1. **Who is developing X3D?**

   X3D is being actively developed by the **Browser Working Group** and the **X3D Task Group**, both official task groups of the Web3D Consortium. Please read the Working Group pages to find out how to participate.

2. **Why should my company support X3D?**

Supporting X3D gives many advantages for a company.

1. Foremost, even if your product uses a proprietary format, supporting X3D instantly gives you access to more tools, content, and compatability with other applications, all will minimal effort. You even get the best of both worlds, your own format PLUS industry compatibility!

2. Your product will benefit by having a competetive marketing advantage by being able to claim "X3D Compatabile!", and this will additionally provide an easy path to leverage industry-wide developments in X3D.

3. There are significant commercial and open-source movements for advancing X3D. This provides a path for your application not having to "reinvent the wheel" everytime new advances in the industry are made.

4. X3D compatibility is easy! X3D-1 is simple to implement.

5. By supporing X3D, your company helps foster growth of the 3D industry as a whole! X3D acts as a unifying platform and unifying marketing banner under which the entire industry can grow.

6. X3D support also provides a path for MPEG-4 support. X3D-1 is the basis for MPEG-4's 3D rendering.

3. **Is there an X3D specification?**

   Yes, but it is under active development. The transition draft of the X3D specification is available at: http://www.openworlds.com/x3d/new/index.html. Originally, a number of companies put forward various proposals for X3D, including Shout Interactive, Blaxxun Interactive, DRaW Computing, New Objectivity, Sony, and Lucid Actual. These were developed and evaluated by the **X3D Task Group** and are now being integrated into a single coherent design in conjunction with the Browser Working Group.

4. **Did you say sample or reference implementation?**

   The X3D group and the Source Code Task Group will produce a sample implementation, not a reference implementation. Basically, a sample implementation is just one example of how you can implement something; whereas a reference implementation defines more precisely how you should implement something. Reference implementations are legal articles: sample implementations are not.

5. **Can I help with the development of X3D?**

   Sure! See http://www.web3d.org/TaskGroups/x3d/X3dTaskGroupCharter.html#Mail for details on signing up as an x3d-contributor or an x3d-reviewer. If your company is a Consortium member and planning on shipping an X3D software product, the Browser Working Group may be for you!

6. **Was there an RFC for X3D proposals?**

   Originally, the X3D process was to be managed by a single, small design team and this team was to make all design designs - i.e. there was no RFC (Request For Contributions). Then the

X3D Task Group was formed and contributions were opened up to anyone wishing to participate. After discussions on the www-vrml mailing list, it was decided to formalize the process for submitting proposals by issuing an RFC.

7. **Is there an Open Source X3D browser?**

   Yes. The Java3D Working Group is producing an Open Source implementation of X3D called Xj3D. A working prototype is available today and has been tested under Linux, Solaris, and soon Win32. Current status is that a VRML 97 compliant DTD is available with a working X3D prototype example. In addition, Blaxxun has released their Contact VRML 97 browser as Open Source and intends to use this for their X3D implementation.

   There are other open source VRML97 efforts that could potentially be used as the basis for an X3D browser, such as the work being performed as part of the OVAL project.

   Finally, if the Cosmo Player source code becomes available in the future, then this could be used as another base for developing an Open Source X3D browser.

8. **Does X3D need the CosmoPlayer source code?**

   No. Some background: the CosmoPlayer VRML 97 browser was originally written by SGI. SGI then spun off Cosmo Software, which was then bought over by Platinum. Platinum offered to provide the source code for the Cosmo browser as Open Source to the Web3D Consortium. Then Platinum was bought out by Computer Associates (CA). CA are currently continuing to look into that offer. X3D is not hanging on the hopes of getting the CosmoPlayer source code.

Contributors

The following people have contributed to the contents of this FAQ. Many thanks to you all!

- Don Brutzman, <brutzman@nps.navy.mil>
- Len Bullard, <clbullar@ingr.com>
- Richard Y. Choi, <web3d@dreamwiz.com>
- Paul J. Diefenbach, <paul@openworlds.com>
- Rick Goldberg, <Rick.Goldberg@eng.sun.com>
- Linda Hahner, <linda.hahner@familiartales.com>
- Adrian Mann, <asm@invetech.com.au>
- Chris Marrin, <chris@marrin.com>
- Eiji Oga, <oga@ruby.famille.ne.jp>
- Nicholas F. Polys <npolys@virtuworlds.com>
- Richard F. Puk, <puk@igraphics.com>
- Martin Reddy, <reddy@ai.sri.com>
- Sandy Ressler, <web3d.guide@about.com>
- Christopher K. St. John, <cstjohn@quik.com>
- Anthony Steed, <A.Steed@cs.ucl.ac.uk>
- Sherrie Thodt, <thodt@hawaii.edu>
- [your name here!]

Comments or additions to Martin Reddy.

This documents lives at: http://www.web3d.org/TaskGroups/x3d/faq/