Stylesheet Transformations for Interactive Visualization:

Towards a Web3D Chemistry Curricula

Nicholas F. Polys
Department of Computer Science
Virginia Polytechnic Institute and
State University (Virginia Tech)
US 540.961.2951
npolys@vt.edu

ABSTRACT
Recent Standards specifications offer important but underemployed techniques to maximize access-to and distribution-of information for real-time 3D visualization over the web. This paper describes and evaluates such techniques to transform structured data such as Chemical Markup Language (CML) to different forms and contexts for Web3D delivery using Extensible Stylesheet Transformations (XSLT), Extensible 3D (X3D), and VRML97. Standards design approaches offer a number of advantages: data durability, data interoperability, and an ecology of tools to be deployed for production and delivery. As we demonstrate, these techniques allow developers to port data between multiple representations and formats, to leverage the separation of the presentation (reference) from the content (referent), and the ability to define 'high-level' markup tags for application-specific needs. By defining a set of XSL Transformations, we are able to generate multiple views and interaction schemes with the same data set; each one 'personalized' for different applications and different levels of expertise.

Categories and Subject Descriptors
D.3.3 [**Programming Languages**]: X3D, VRML, XML, XSLT, DOM, CML, Java

General Terms
Standardization, Languages, Design,

Keywords
Information Visualization, Interactive 3D Graphics, Molecular Chemistry, Education

1. INTRODUCTION

International standards organizations serve the computing community by developing and specifying open platforms for digital data exchange- they provide the evolving nexus for Integration and Interoperability. By building to industry standards, organizations can lower their costs for software and data integration, maximize their data re-use and all the while guarantee author reliability and user access beyond market and political vagaries. Such internationally recognized standards are manifested as widely deployed web technologies such as HTML, SGML, and the XML family produced by the World Wide Web Consortium (W3C) [18] and Virtual Reality Modeling Language (VRML) and Extensible 3D (X3D) produced by the Web3D Consortium (Web3DC) [14].

This paper examines important but underemployed techniques for 3 or 4-dimensional visualization of XML data with X3D and VRML. Through diligent and cooperative work, the X3D Task Group and the W3C have converged on an XML encoding for interactive 3D scenegraphs – X3D. X3D is a componentized and improved successor to VRML97 for describing and programming Web3D virtual environments. By using the XML encoding of X3D, developers can leverage the entire XML family of technologies and get a suite of tools for data portability as XML is founded on the distinction between content and presentation [7, 9, 15]. We believe that the full impact of the convergence between Web3D and XML is not yet fully understood, and intend this project to show the power of open standards and technologies.

In this project, multiple visualizations of the same data are be achieved through Extensible Stylesheet Transformations (XSLT) technologies [18]. Using a structured data representation from the Chemical Engineering community, Chemical Markup Language (CML) [3], we will demonstrate a set of web-ready methodologies for processing, publication and display that can be applied to any XML dialect or structured data source.

1.1 Extensible Markup Language (XML)

The World Wide Web Consortium's (W3C) metalanguage codification of XML has opened new and powerful opportunities for information visualization as a host of structured data can now be transformed and/or repurposed for multiple presentation formats and interaction venues. XML is a textual format for the interchange of structured data between applications [18, 7, 15]. The great advantage of XML is that it provides a structured data representation built for the purpose of separating content from presentation. This allows the advantage of manipulating and transforming content independently of its display. It also dramatically reduces development and maintenance costs by allowing easy integration of legacy data to a ***single data representation which can be presented in multiple contexts or forms*** depending on the needs of the viewer (a.k.a. the client). Data thus becomes 'portable' and different formats may be delivered and presented (styled) according to the application's needs.

Another important aspect of XML is the tools that it provides: the DTD and Schema. The Document Type Definitions (DTD) defines 'valid' or 'legal' document structure according to the syntax and hierarchy of its elements. The Schema specifies data types and allowable expressions for element's content and

attributes; they describe the document's semantics. Using any combination of these, ***high-level markup*** tags may be defined by application developers and integration managers. This allows customized and compliant content to be built for use by authors and domain specialists. These tags could describe prototyped user-interface elements, humanoid taxonomies, or geospatial representations [14]. Developers can describe the valid datamodel for their application by using the DTD and Schema, share it over the web, and standardize it amongst their community.

XML can be as strict or as open as needed: content or fragments of content can be 'well-formed' and still processed with most XML tools. Typically, data validation is at author time, but it can be done at serving, loading, or runtime if needed. From an XML-compliant source document or fragment, logical transformations (Extensible Style Sheet Transformations (XSLT)) can be applied to convert the XML data and structure to another XML document or fragment. A series of such transformations may be applied ending with a presentation-layer transformation for final delivery target style, content-type integration, and display. [7, 15, 18]. Common XSLT design patterns have been described [7]; those used in the first phase of visualization evaluation were: fill-in-the-blank, navigational, rule-based, and computational.

1.2 Extensible 3D (X3D)

The Web3D Consortium's next-generation successor to VRML is X3D which, like XML, moves beyond just specifying a file format or a language like VRML or HTML. It is a set of objects and interfaces for interactive 3D Virtual Environments with bindings defined for multiple profiles and encodings and collected under a standard API [13, 14]. The X3D specification describes the abstract performance of a directed, a-cyclic scenegraph which can be defined by an XML binding using DTDs and Schema [14]. Also, rather than defining a monolithic standard, the X3D specification is modularized into components which make up 'Profiles'. Profiles are specific sets of functionality designed to address different application domains from simple geometry interchange or interaction for mobile devices and thin clients to the more full-blown capabilities of graphical workstations and immersive computing platforms.

The X3D Task Group has provided a DTD, Schema, an interactive editor, and a set of XSLT and conversion tools for working with X3D and VRML97. Publishing techniques fro Web3D are discussed in detail in section 2, but since native X3D browsers are not yet widely deployed, most of the power of the transformation and composition of X3D happens 'behind the scenes' and then presented with a VRML97 viewer. The official file extension for X3D files is `.x3d` and the official MIME type for X3D files is defined as: `model/x3d`. A comparison of X3D encodings is shown in Table 1.

Table 1. X3D Encoding Comparison

VRML-style encoding	XML encoding
Nodes	Elements (tags)
fields	Un-writable attributes (readable)
eventOuts	Un-writable attributes (readable)
eventIns	Un-readable attributes (writable)
exposedFields	Writable, Readable attributes
ROUTES	Elements (tags)

VRML-style encoding File Header

```
#VRML V3.0 utf8
PROFILE Interactive
COMPONENT Scripting:1
META "key value string" "value string"
Group { … }
```

XML encoding File Header

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE X3D PUBLIC "-//Web3D//DTD X3D 3.0//EN"
"http://www.web3d.org/specification/x3d/x3d-3_0.dtd">
<X3D version="3.0" profile="Interactive">
<head>
 <component name="GeoData"/>
    <component name="Nurbs" level="2"/>
     <meta name="description" content="X3D scene header example"/>
  </head>
<Scene> … </Scene>
</X3D>
```

In addition to the suite of diverse low-cost tools XML technologies provide for transformation and presentation, other bonuses of using the XML encoding and XML tools for X3D is that it could integrate with Scalable Vector Graphics (SVG), Synchronized Multimedia Integration Language (SMIL) for example [18, 14].

1.3 Chemical Markup Language (CML)

CML is a new approach to managing molecular information. It has a large scope as it covers disciplines from macromolecular sequences to inorganic molecules and quantum chemistry. CML is new in bringing the power of XML to the management of chemical information. CML contains a number of kinds of information about a particular compound, from its physical atomic structure and elemental makeup to its water solubility, melting point, atomic weight, and various spectral descriptions. These are manifested by a variety of datatypes in tags describing structural, numerical, and meta data about the chemistry of the molecule. Dr. Peter Murray-Rust, Dr. Henry Rzepa, Dr. Michael Wright, and the other developers of CML have provided a DTD, Schema, and API toolkit (CMLDOM). In addition, they provide Jumbo, a Java-based application to load and view a number of chemical formats in CML (CIF, MDL, MOL2, PDB, XYZ JMOL) and write them out to any of those formats. [3]

Of immediate interest for Web3D Visualization was the top-level <molecule> tag and its children the <atomArray> tag with 2D or 3D Euclidean coordinate children naming the atoms and describing the atom type and location, and the <bondArray> tag describing the bonds between the various atoms. Structural and geometrical information was not the only interest for Web3D display since CML carries much more: <formula><float><string> each describing important compound data such as the formula,

physical properties, and official and alternate names respectively. In addition, <chimeral: spectrum> tags are included in some compound files which describe the measured infrared and ultraviolet/visible spectra exhibited by the molecule.

2. OVERVIEW OF WEB3D PUBLISHING

Publishing data for multidimensional visualization over the web has shown 3 principal requirements which are met by open standards: a structured yet flexible data format to describe and deliver real-time 3D objects and environments, scalable pathways to integrate and distribute intellectual property to common platforms (WWW, CD, DVD), and interoperable and reusable content which is not bound to proprietary tools and systems. When architecting such systems, developers take into account the data source and production workflow (logical transformations) and consider the needs of target platforms and the delivery mechanisms (presentation transformation): hardware, software, connectivity, runtime integration. Using this information, application scene-graphs are designed and built to one or more of the multiple Profiles specified by X3D: Interchange, Interactive/MPEG-4, Extensibility, Immersive (VRML97), or Full [12]. X3D provides authors the power to add and integrate new nodes and application components; and allows points of interoperability with W3C and MPEG media.

Web3D publishing and content management for open standards systems can be divided into 3 basic kinds: the export or publication of a VRML file which can be run in a web browser or embedded in an HTML page (with internal and external ECMAscript or Java scripting), the compositing of VRML parts from a database and delivered from a webserver (e.g. PHP, MySQL, Perl, Java) as above, and more recently with X3D, the transcoding/transformation of XML data to X3D for delivery or other Web presentation as VRML, or HTML formats (e.g. Perl+XML, Java+XML, Apache Cacoon). This last method, also the topic of this work, uses Extensible Stylesheet Transformations (XSLT) to easily convert the scene-graph representation between encodings and formats. The X3D Task Group of the Web3D Consortium has developed a set of XSLT stylesheets and hundreds of translated content exemplars included on the Web3D Consortium X3D Software Development Kit CDs (2000-2002) [14].

2.1 Producing Web3D Visualizations

We divided the scene into 'blocks' which described the various functional parts of the scene. These were:

1. Define environment & locations
2. Define user interface & Viewpoints
3. Model objects & Prototypes
4. Define interactions
5. Organize and collect scene
6. Optimize scene with post-production tools

The degree of 'granularity' & 'encapsulation' in the scene abstraction will impact the data representation, its production, as well as its storage and retrieval- design is 80% of production!

The result of this approach was an implicitly structured X3D 'Document' describing our various CML scenes in the form of:

Content type

Header
PROTOs & EXTERNPROTOS
World set (Backgrounds, ProximitySensor)
HUD & User Interface
Scripts
Environment & Populus set (molecular geometry)
ROUTES

This allowed a modular structure to the scene that could be build from any number of applications or databases to the final target presentation.

2.2 Publishing Web3D Visualizations

A number of web, print [13], and conference resources detail the production and publication techniques of VRML97, so we will not address those here. These techniques are usually described for use with the common and capable web browser plug-ins [1, 11, 12]. A number of standalone browsers and application tookits for VRML97 also exist such as OpenWorlds [11] and Xj3D [14] both of which are X3D capable. Still, given the widespread web support of VRML97, we considered VRML97 the default presentation layer format for our Web3D project.

Recent work worthy of mention for our CML X3D project are the XML capabilities of X3D that work before it is delivered to the user. The X3D Task Group has provided a number of these, such as XSLT style-sheets for the transformation of: X3D -> VRML97, X3D -> HTML, and X3D -> wrap/unwrap. Also, courtesy of the National Institute of Standards Technology (NIST), a translator application for VRML97 -> X3D data migration that has been made freely available and been integrated into a number of Web3D editing tools including X3D Edit and White Dune. [14, 16]

XML/XSLT approaches to data management and web presentation can be divided into 2 kinds: the back-end production of a file archive, and the serving of a source document in response to a 'live' (networked) transformation request (see Fig. 1). Given server overhead, bandwidth, and delivery contraints, periodically auto-generating a web-ready content archive may be appropriate. These approaches use X3D source files and directories (with naming conventions) with scripted XSLT to produce framed HTML, VRML, and X3D document trees complete with linked with chapters, titles, and embedded views of the source file. The generated document trees can be organized and hyperlinked for navigation with a web browser [14]. The auto-generation can be done with straightforward batched XSLT Java [9] or Perl [6] scripts. These content publications can then be served over the web or distributed on CD or DVD. The second approach is using XSL Transformations 'on the fly' using common web server software such as Apache Cacoon [17] or Perl and the XML Gnome Libs [6]. This approach can provide custom or database-driven presentations of the source data with a proportionate server overhead.

Kim and Fishwick [8] demonstrated the power of the content/presentation distinction when they used XML, Schemas, and XSLT to render their XML descriptions of dynamic, physical systems to different visual and system metaphors they call *rubes*.

Dachselt et al. [4] demonstrated an abstracted, declarative XML and Schema to model Web3D scene components and especially interfaces. The systems described in these papers in addition to the display conventions of molecular viewing software influenced the design and development of this project.

Sufficiently impressed with the power of XSLT and other XML technologies in the Web3D publication pipeline, we embarked on our demonstration CML-X3D XSLT implementation described below (section 3.2). XSLT programming is relatively straightforward and a number of free tools are available so we used this phase as a highly-iterative development cycle where each 3D scene-graph architecture and visualization mode (rendering) was evaluated with respect to other chemical visualization applications and kinds of display interactivity they used. We developed 3 versions of the cml2x3d.xsl style-sheet each adding the necessary X3D nodes and logic to achieve the display and interactive context of the visualization. We also used the X3D to VRML97 XSLT stylesheets provided by the X3D Task Group to transform these X3D files to final VRML97 presentations.

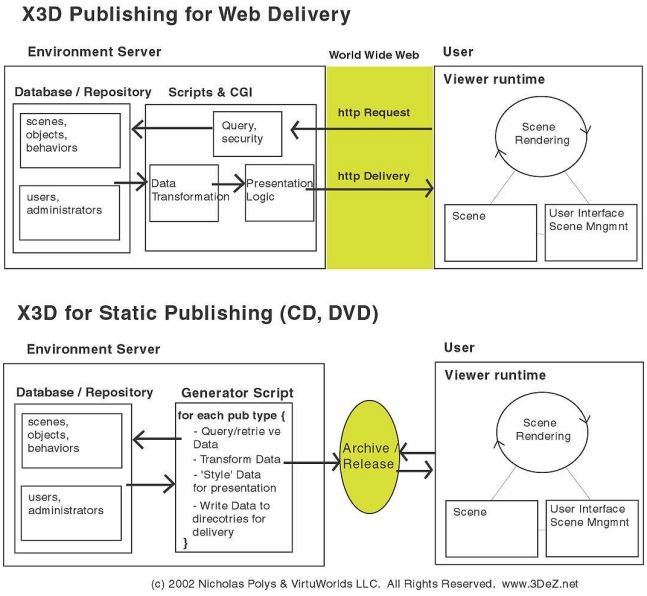

Figure 1. Overview of Web3D Publishing

3. EXTENSIBLE STYLESHEET TRANSFORMATIONS FOR INTERACTIVE 3D CHEMISTRY

Numerous developer resources exist for the World Wide Web Consortium's (W3C) XSLT specification [2, 7, 15, 18] so we will focus on the design issues specific to CML and X3D. However, a review of the typical XSL Transformation process is in order:

1. An XSLT engine parses the source XML document into a tree structure its various nodes
2. The XSLT engine transforms the XML document using pattern matching and template rules in the .xsl style-sheet
3. Template elements and attribute values replace matched element/attribute patterns in the source document in the result document.

First, for each of our XSLT files, we specified the XSL attributes of the output method, encoding, media-type, and cdata-section-elements, and the DOCTYPE system for our result X3D document. Next, we matched the root element of the source document and wrote out our <X3D> XML header information to the result document. This manifested a blank X3D <Scene> (as in Table 1) into which we could apply our other template rules into an X3D node structure. The root-element template also supplied 3D environment-specific X3D nodes to the result document such as <ProtoDeclare>s, <Background>, <PointLight>, <NavigationInfo>, entry <Viewpoint>, and any required <Inlines>. When transforming CML to X3D, we assume 1 Angstrom = 1 Meter for our coordinate units.

3.1 Design

As a design approach, we began with the minimum number of nodes required to represent the basic CML molecular geometry (atoms and bonds) since that was the first target for simple visualization in 3D. However, for our purposes of maximum visualization (seeing as much of the CML information as possible), efficient representation was best achieved through the use of the PROTO node, Text node, and others which required the VRML97 Profile or higher. Story-boarding, scene-design, and prototyping where crucial to this process. It is conceivable that other, future applications may only use the Interchange or Interactive profile depending on the kind of data they are exchanging or interactivity they require.

For our transformations, we defined X3D <ProtoDeclare> prototype nodes for each chemical element the style-sheet might encounter as an <atom> CML tag. We did not cover all cases, but defined a color, atomic radius, and element label for: Carbon, Hydrogen, Nitrogen, Oxygen, Flourine, Silicon, Phosphorus, Sulphur, Chlorine, Bromine, Iodine, and a default 'unknown'. We exposed the X3D atom prototype's position and transparency fields for future event routing. Next declared as X3D <ProtoDeclare>s were: an <IndexedLineSet> to draw chemical bonds described in the CML <bond> tags, and a set of 3 <Text> node headings for the display of titles, formulae, and other annotations. Encapsulating these objects as PROTOs meant that we could change the ***displayed form*** of the instantiated objects relatively easily; for example using some other geometry besides a colored sphere for atom or white lines for a bond (see Fig. 2).

After we determined satisfactory display transformations for the molecular geometry, we then added more robust examples of data display and model interaction. These examples changed the presentation's ***interactive context***. These included more advanced capabilities for interaction with: billboarded text, transparency sliders, measuring planes, gimbals, and drag sensors. Each version used its own XSLT stylesheet to generate the different X3D environments from the same CML source file. These transformed 3D environments provided varying degrees of information and interaction richness; the basic result could be used to see the fundamental structure of the molecule, another result provides a level of manipulation that may be appropriate for a high-school chemistry class, while the last provides more powerful affordances tuned to expert or collegiate use.

3.2 Implementation

After assembling all our scene resources for the X3D result, we next addressed the top-level <molecule> element in the source CML document directly in another template-match. According to the CML DTD [3], any <molecule> tag may contain a <formula>, <atomArray>, <bondArray>, or <date> tag and any number of <string> or <float> children tags. Since the <molecule> tag contains some attributes of interest, we began with a <ProtoInstance> of our "title_text" Proto. We declared an <xsl:variable name="$fullname"> whose selected value is the concatenation of the <molecule title=""> and the context's child <formula> information. This string was then used as the 'txt' attribute filed for the title_text prototype and transformed to an arbitrary textual information plane, <Billboard>, or HUD. A XSLT for-each structure then recurses annotation positioning and font size (as "ano1_text" and "ano2_text" <ProtoInstance>s) as it extracts the CML molecule's <float> and <string> children tag and attribute information to the display plane.

We continue defining XSLT templates to evaluate the geometrical properties of the CML compound as in the <atomArray> and <bondArray> tags. When we match the CML <atomArray> tag, we begin an X3D <Group> and recurse for-each <atom> testing if it contains 2D or 3D coordinate information and recording its ID in an <xsl:variable>. Addressing elements and attributes for variables and conditionals in XSLT is done through Xpath expressions (see the code or the resources cites for more). If the <atom> is has 3 dimensions, we concatenate those into an <xsl:variable> describing the atom's position; and instance it with a DEF'd <ProtoInstance> of the proper chemical element. If it only has 2 dimensions, we assume z = 0 in our position variable and similarly instanced a proper and positioned, DEF'd atom. Each atom was instantiated in the result X3D scene-graph with a unique name as well as exposedFields for position and transparency animation control. The </Group> was then closed, completing atom translation processing. The design of variables and prototyped node objects is certain to be application-specific; ours were the result of the design process described above.

Next, we wrote a template to process and translate the CML <bondArray> tag. This template was also written enclosed in an X3D <Group> tag. Within this context, the CML <bond> tag commonly describes the ID (or DEF) name of the bond, between which 2 atoms the bond exists, and the 'order' of that bond (e.g. 1, 2, 3) [3]. For-each bond order, we initialized a set of <xsl:variable>s which retrieved the bonded atoms' coordinates (via an Xpath expression) and supplied those as a Vector3FloatArray exposedField to the named "line" <ProtoInstance>. This template is simplistic in that higher order bonds are always drawn offset to the first bond with a positive slope computed by mean, regardless of quadrant. This arithmetical fix could be achieved with more enumerated cases in the XSLT. We note that in the application of chemical engineering, other bond representations and information would be relevant. This is a fertile area for future work and collaboration with scientific and educational end-users.

3.3 Delivery

We targeted VRML97 for the final presentation format as a number of robust viewers exist for web distribution and the X3D Task Group provides a free, open-source pathway for conversion [14, 1, 11, 12]. We wrote a series of XSLT style-sheets each adding more user-interface widgets to manipulate and view the molecule and its physical data (see Fig. 3). Given an X3D document in the XML encoding, users can view it in an X3D-capable browser directly, or developers can apply other XSLT style-sheets not only to transform the data to different X3D Profiles or presentational formats (VRML97, HTML, POV, PDF, etc), but also 'style' the data according to the user-target and the associated template rules.

Once the XSLT style-sheets are designed and written, the final presentation files can be auto-generated from the source XML with Java or Perl scripts [6, 9, 2]. The resulting document trees can then be served from a website or otherwise published on a CD-ROM or DVD. The CML Source files for a number of compounds, XSL Transformations files, X3D, and VRML97 result files for this project are browsable at: http://www.3DeZ.net/X3D/CML. The other delivery method using our XSLT stylesheets is for the server-side ('on-the-fly') transformation of the XML representation using software such as Apache Cacoon, and Perl [17, 6].

A natural extension of our stylesheets would be to deploy them for the server-side customization of display form and interactivity context rather than archival release. For example, a student visiting an online course has shown or specified that 3D exploration tasks with an expert interface are preferred and they need a specific geometry type to represent molecules and bonds in 3D. Using Cookies, a user database, and a geometry and user interface repository in conjunction with XSLT, the system may serve them an integrated 2D/3D graphical user interface with full interactive control and modify the rendered colors and geometries of the content according to this preference. This personalized delivery of a single data representation again illustrates the power of XML and XSLT in regards to data portability and re-use.

4. RESULTS AND FUTURE WORK

This project demonstrates the conceptual and technical power gained through the separation of presentation and content and the use of web standards. In the field of chemistry, the XML dialect of Chemical Markup Language provides a structured and manageable format for the conversion and sharing of common molecular data files. When considering the re-use of a single representation for multiple presentation forms and contexts such as the delivery of multimedia web curricula, an online lab, or an educational CDROM, the portability advantages of XML become clear: increased distribution for lower production and maintentance overhead. By building our application with the XML encoding of X3D, we have shown flexible transformation paths for CML to real-time 3D visualization using common toolkits and languages such as XSLT, Perl, and Java. These same rule-based and computational approaches may be applied to visualize virtually any XML data in real-time 3D.

There are a number of fertile areas for future work with CML, XSLT, and X3D. Respectively:

- Expound user requirements and formalize CML -> X3D templates to address visualization needs of educators, chemical engineers, and researchers. The stylesheets could then be applied to the NIST WebBook [10] or other chemical data repository for example.
- Work is progressing on componentized taxonomies and mark-up for 3D User Interfaces (UIs) [4, 5] and this is also a prime area for future research and development. This would entail working with VR HCI researchers and accessibility experts to develop a descriptive and framework for composing user-interfaces in information-rich 3D worlds. This would also include improving and formalizing widget-component libraries.
- Explore and evaluate serialization and performance issues for runtime control through the Scene Access Interface and the Document Object Model (DOM) scripting and integration.

5. ACKNOWLEDGMENTS

Many thanks to Dr. Don Brutzman and the X3D Task Group for continued guidance during the evolution of this application and X3D, Dr. Murray-Rust who welcomed and oriented me to CML, as well as Dr. Doug Bowman and the other faculty at Virginia Tech who reviewed this manuscript.

6. REFERENCES

[1] Blaxxun Contact VRML Browser [Win] http://www.blaxxun.com

[2] Brown, Martin. XML Processing with Perl, Python, and PHP, Sybex, San Francisco, 2002

[3] Chemical Markup Language (CML)

http://www.xml-cml.org
http://www.ch.ic.ac.uk/rzepa/chimeral/

[4] Dachselt, Raimund; Hinz, M., Meissner, K.,. "CONTIGRA: An XML-Based Architecture for Component-Oriented 3D Applications", *Proceedings of the Web3D 2002 Symposium*, ACM SIGGRAPH 2002

[5] Figuroa, Pablo, Green, M., Hoover, H.J. "InTml: A Description Language for VR Applications", *Proceedings of the Web3D 2002 Symposium*, ACM SIGGRAPH 2002

[6] Gnome XML & XSLT Libs for Perl http://www.gnome.org

[7] Kay, Michael. XSLT Second Edition, Wrox Press, Birmingham UK, 2001

[8] Kim, Taewoo and Fishwick, Paul. "A 3D XML-Based Customized Framework for Dynamic Models", *Proceedings of the Web3D 2002 Symposium*, ACM SIGGRAPH 2002

[9] McLaughlin, Brett. Java & XML Second Edition, O'Reilly, Cambridge, 2001

[10] NIST WebBook http://webbook.nist.gov/chemistry/

[11] OpenWorlds Software : Viewers and Tooklits http://www.openworlds.com

[12] Parallel Graphics' Cortona VRML Browser http://www.parallelgraphics.com [Win, Mac, …]

[13] Walsh, Aaron and Sévenier, Mikael. Core Web3D. Prentice-Hall, Upper Saddle River, NJ, 2001

[14] The Web3D Consortium

Specifications: Extensible 3D (X3D), Virtual Reality Modeling Language (VRML- ISO/IEC 14772:1997)

http://www.web3d.org/fs_specifications.htm

X3D TaskGroup: http://www.web3d.org/x3d.html

Software Development Kit: http://sdk.web3d.org

Xj3D Open Source X3D/VRML toolkit:

http://www.web3d.org/TaskGroups/source

[15] White, Chuck. Mastering XSLT, Sybex, San Francisco, 2002

[16] White Dune X3D-VRML authoring/animation toolkit for Linux + UNIX w/ input devices

http://www.csv.ica.uni-stuttgart.de/vrml/dune/

[17] Williams, Clifton. "Network Application Server using XML to Support Distributed Databases and 3D Environments" Master's thesis Naval Postgraduate School. http://theses.nps.navy.mil

[18] The World Wide Web Consortium

Specifications:

Extensible Markup Language (XML): http://www.w3.org/XML

Extensible Stylesheet Transformations (XSLT):

http://www.w3.org/TR/xslt11