X3D-UML: Enabling Advanced UML Visualisation Through X3D

Paul McIntosh Margaret Hamilton Ron van Schyndel

School of Computer Science and Information Technology
RMIT University
GPO Box 2476V, Melbourne, VIC., Australia, 3000

paul.mcintosh@internetscooter.com,{mh,ronvs}@cs.rmit.edu.au

Abstract

The Unified Modelling Language (UML) has become a commercially accepted standard for visualising software systems. Much of this success can be attributed to the computer aided software engineering (CASE) tools, which enable the UML to be effectively integrated into the software development life cycle. The UML is designed to be tool independent, however, in practice it would not be possible to manage the complexities of developing a large software system with "pencil and paper" UML diagrams. The UML is also designed to be extensible, allowing advanced use of visualisation such as 3D, however, in practice this extensibility is limited by the CASE tool used. With X3D-UML we present a means of using eXtensible 3D (X3D) as a visualisation medium for UML, enabling both standard and advanced visualisation to occur independent of CASE tools. We discuss the means, problems and benefits of transforming existing software system information into UML within an X3D environment.

CR Categories: D.2.2 [Software Engineering]: Design Tools and Techniques—Computer-aided software engineering (CASE); I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Virtual Reality; D.1.7 [Programming Techniques]: Visual Programming

Keywords: UML, X3D, Web3D, Software Visualization, VRML, X3D-UML, Virtual Reality Modeling Language, Unified Modeling Language, XML, JavaML

1 Introduction

The computer environment for which software is developed and executed is continually changing. Increases such as in network and processor speed enable software systems to grow larger and more complex. More often than not, the software development platforms used to create the software, are the same or similar, and make use of the very same increases. Network speed increases mean that software engineers can more easily collaborate worldwide. Processor speed increases mean that CASE (Computer Aided Software Engineering) tools are able to become more advanced and better aid the software engineer in the software creation process.

3D graphics hardware capability is also increasing continually, however use of these advances does not appear as readily in software development environments. For example, 3D visualisations are not an everyday software abstraction found in software engineering tools. Little research has been done into how to effectively integrate computer graphics, such as 3D, into software engineering processes. If we are to determine whether software visualisation should also evolve inline with increases in computer graphics capability, we must first provide a means of transforming current software processes into the 3D medium. In this paper we propose a combination of X3D and UML, which we term X3D-UML, which demonstrates how software 3D abstraction can occur as readily as current 2D and textual abstractions. Furthermore X3D-UML, by utilising Web3D technology, provides an entry point into advanced software visualisation which is not limited to 3D alone.

1.1 The State of 3D Software Visualisation

If we consider one use of 3D, computer games, we can think of this as one form, although contrived, of existing 3D abstract data manipulation. It is clear, by the popularity of such games, that people not only currently complete complex tasks in this environment, but enjoy the experience. Although such fantasy environments may not directly suit software development, why are not similar 3D interfaces, in any form, being as commonly used?

Previous research has shown that there are practical advantages to using 3D for software visualisation. For example Ware, Hui et al. [1993] demonstrated that the use of the third dimension can allow software engineers to view complex systems more accurately. Dwyer [2001] also demonstrates its use can provide a means for understanding complex software system architecture.

Franck, Sardesai et al. [1995], Feijs and Jong [1998], Fishwick [2000] and Maletic, Marcus et al. [2001] researched various forms of 3D software visualisation, based on spheres, boxes and other metaphors. Gil and Kent [1998] and Gogolla, Radfelder et al. [1999] have proposed 3D Unified Modelling Language notation. However software engineers rarely use 3D environments in their every day work although the notations, evidence of benefit and graphics capability are available.

1.2 Barriers to the Adoption of 3D Software Visualisation

Some reasons as to why 3D visualisation may not have been adopted are suggested in this section. These are not conclusive but do give an outline of how our concepts of X3D-UML have evolved.

Computer Graphics – Computer graphics capabilities have been a major limitation, since any form of 3D graphics is limited by current hardware capabilities. It is apparent that these hardware restrictions have also applied to software visualisation.

Evolution - One possible reason given for lack of 3D use is that software engineering is an evolving discipline. For new concepts to be adopted they need to evolve from existing accepted practices and tools. Software engineers, like rock climbers, ensure firm footholds in what is around them before reaching for the next new thing. As 3D is still a new concept, it also needs to be introduced without the need for software engineers to completely let go of traditional practices (such as the "all powerful" command line interface). Even if we ignore human factors, without evolution, we cannot leverage existing tools, which add value and also evolve. As in the example of html adoption, everyday text editors enabled basic text-based editing to first occur and then some evolved into specialised html editors.

Open Standards – Another possible reason is that there are no open standards in the area of 3D software visualisation, and therefore 3D software visualisation cannot be readily produced and shared. This is further complicated by the fact that standards generally are created in response to a need, so such standards will not evolve unless 3D visualisation is first used. The issue of openness is also important as far as the ability of software engineers to trust the technology and alleviate fears of being tied to a proprietary system, which is out of their control.

Perceived Return on Investment – Advances in software engineering which require a large amount of effort, are generally balanced against perceived return on investment (whether it be in time or money). 3D requires a large amount of effort, Walsh and Bourges-Sevenier [2000] state "the field of computer science that deals expressly with creating, manipulating and navigating computer content in three dimensions – is difficult. Extremely difficult.". These difficulties of 3D software visualisation hinder exploration into its benefits. Furthermore, without empirical evidence of benefits, it is difficult to justify further research or even use.

Software Changes – A software system is ever changing during the software development life cycle. If effort is to be expended in creating a software visualisation, that visualisation has to also adapt to changes in the software. This adds complexity to an already complex process of producing the initial 3D visualisation.

Desktop Computer Limitations – Any visualisation is most valuable at the point where it is most needed, on the desktop of a software engineer, where it can provide an extension to the current work flow. Unless the visualisation can work within a standard software engineer's environment, which typically has no specialised graphics capability, and integrate with current tools, then there can be little or no benefit.

Although the reasons above are pure conjecture, Diehl [2002] also summarised similar ideas when referring to the future directions of software visualisation research, "Software visualization will be doomed to stay an academic endeavor, if we do not succeed to integrate it into working environments and thus into the work flow of programmers, designers and project managers. To facilitate such integration existing standards must be adopted or extended..."

1.3 Integrating Web3D and Software Visualisation Standards

This paper presents a possible solution, X3D-UML, designed to address all the issues discussed above. Our solution is based on integrating an existing open standard in Web3D visualisation (X3D) and software visualisation (UML). Our solution does not rely on specialised graphics hardware and the test system, used for the results in this paper, is a standard notebook computer. Our solution uses existing software information to generate its visualisation, which both allows evolution from existing practices and maintains a consistent visualisation throughout the software development life cycle. Finally, as X3D defines content using standardised text, effort expended in creating 3D software visualisations can be more easily shared, reused and refined by researchers and engineers.

In Section 3, we address the history and open standard of UML. In Section 4, we discuss the suitability of X3D as a 3D visualisation web standard to be applied to software. We present the elements and the advantages each standard brings to Web3D software visualisation. In Section 5, we discuss the issues involved in transforming current working environments into X3D-UML. Finally, in Section 6 we present working examples of X3D-UML, summarising this and future research in Sections 7 and 8.

2 UML

The UML (Unified Modelling Language) is a trademark of the OMG (Object Management Group, Inc - www.omg.org). The UML, as described by Booch, Rumbaugh et al. [1998], "is a graphical standard for visualizing, specifying, constructing, and documenting the artefacts of a software-intensive system'.

The UML is not only flexible, allowing new applications, but has experienced commercial acceptance. Commercial acceptance allows the products of this and subsequent research to be applied to a large variety of real production systems.

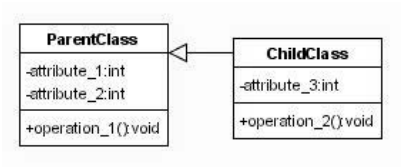

Figure 1 - Example of UML Visualisation. This diagram, produced by a CASE tool (Poseidon for UML - www.gentleware.com), demonstrates how a class relationship can be visualised in UML.

2.1 History

Booch, Rumbaugh et al. [1998] state that the UML effort started in October 1994 and the first specification 1.1 accepted in November 1997. The success of UML can be measured by the proliferation of UML tools, with Objects by Design [2004] listing more than 100. The number and variety of these tools also provides evidence of the flexibility of the UML to be used in different situations through the use of such tools.

2.2 UML for 3D Software Visualisation

As discussed above, the UML has gained wide acceptance in the software engineering workflow and this is the main reason why we chose the UML. Ironically, another major reason that the UML was chosen, is its difficulty to implement. By using an accepted standard, we can test the ability to render "real" software visualisation, rather than something contrived to suit the environment. By using the UML as a benchmark we gain greater

understanding of the problems of software visualisation in Web3D environments.

There are further advantages, in that the UML provides flexibility to expand its concepts from the standards given. Booch, Rumbaugh et al. [1998] give explicit reference to this when referring to the nine common diagrams provided by the UML, stating that "To fit the needs of your project or organization, you can create your own kinds of diagrams to view UML elements in different ways"

Booch, Rumbaugh et al. [1998] also give explicit reference to 3D, in the same section, noting "In practice, all the diagrams you'll create will be two-dimensional, meaning that they are just flat graphs of vertices and arcs that are drawn on a sheet of paper, a whiteboard, the back of an envelope, or on a computer display. The UML allows you to create three-dimensional diagrams, meaning that they are graphs with depth, allowing you to "swim" through a model. Some virtual reality research groups have already demonstrated this advanced use of the UML".

The research referred to in this statement was that of Gil and Kent [1998]. Gil and Kent also note that to test the use of UML in this way some form of 3D environment would need to be developed. They suggest a prototype in VRML could be created, however this has not been pursued to date.

The UML does not limit the use of 3D to diagrams. Stereotypes allow the creation of new visualisations which give greater meaning than a standard type. For example, if a series of classes represented a hardware input type, we could use text to create a stereotype "<<hardware input>>". We could also create a visualisation of that hardware input, and display this instead of the traditional class "box".

In practice, although the UML does not limit the use of advanced visualisations, it is likely that the UML CASE tool will. Although it may be possible to write integrations into some UML tools, the initial effort in creating the 3D visualisation would still be present and serve as a barrier.

Since the comments of Booch, Rumbaugh et al. [1998], the UML specification has evolved, with UML 2.0 expected to be completed in late 2004. Of particular interest is the emerging use of XML (Extensible Markup Language), such as the Object Management Group [2002] XML Metadata Interchange (XMI) Specification and the Object Management Group [2003] Unified Modeling Language: Diagram Interchange Specification. These technologies allow, through the use of XML, UML information to more readily be transferred between tools. The use of XML also allows the information to be more readily transformed into new forms, such as 3D.

3 X3D

Extensible 3D (X3D) is a trademark of the Web3D Consortium (www.web3d.org) and is the next generation of VRML (Virtual Reality Modelling Language). The Web3D Consortium [2004] defines X3D as "a software standard for defining interactive web- and broadcast-based 3D content integrated with multimedia". X3D allows users to download, display and interact with 3D content via a web browser plug-in or standalone viewer.

3.1 History

X3D is the next generation of VRML, based on XML. VRML is now 10 years old, being introduced at the first international World Wide Web Conference in 1994 by Mark Pesce and Tony Parisi.

In August 2004, the Web3D Consortium announced that the X3D specification was approved by the International Standards Organization (ISO) and formal publication is expected in October of 2004.

The use of VRML for software visualisation has been explored previously. As one example, Maletic, Marcus et al. [2001] used VRML for "Ismovision" (pictured figure 2) and demonstrated a language "COOL", that maps C++ to a VRML visualisation.

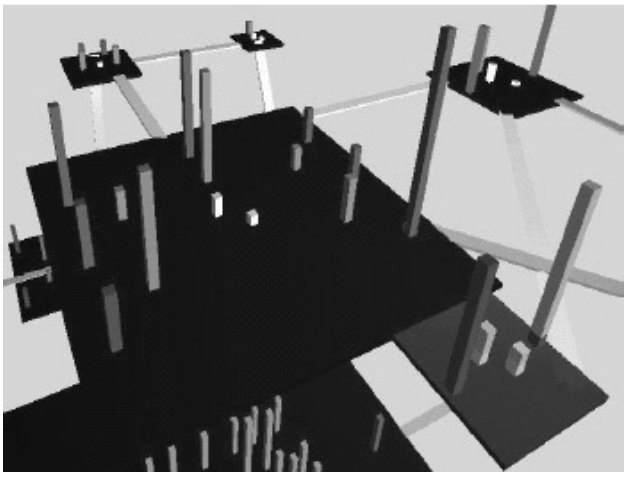

Figure 2 - Imsovision (IMmersive SOftware VISualizatION), by Maletic, Marcus et al. [2001]. This demonstrates a C++ software system displayed in a VRML world based on a language COOL.

3.2 X3D for 3D UML Visualisation

The benefits of X3D for 3D web content are apparent. With respect to software visualisation, some of the benefits are outlined below. This list, from the feature list defined in the Web3D Consortium [2004] X3D specification, demonstrates that X3D captures functionality far beyond just 3D. X3D provides a feature rich modelling environment which should enable advanced software visualisation from both a development perspective, to understand the system from a design point of view, and runtime perspective, to understand the system as it executes.

3D graphics – X3D provides a rich set of modelling features, designed for creating 3D worlds, which can also be applied to software visualisation. Application of 3D can be as simple as adding depth to a UML diagram or as complex as modelling the software system, its mechanical interfaces and the context environment (e.g. modelling an air traffic control system, the control panel, the airport, aeroplanes and the users of the system).

2D graphics – As most current software development involves 2D, such as text editors, manuals and computer screens, some of these aspects can be directly translated and displayed as 2D in a 3D world. Similarly standard 2D UML diagrams could be directly rendered.

Animation – The ability to move objects in real-time allow features such as automatic "force directed" layout of diagrams, which Dwyer [2001] has shown to provide benefit for understanding complex software system architecture. Other examples of use could be a form of visual debugging similar to that proposed by Jacobs and Musial [2003], or a means of modelling environment behaviour (as with the air traffic control example). Animated UML, using VRML, has already been demonstrated by Thaden and Steimann [2003] as a means of intuitively teaching object-oriented program execution.

Spatialized audio and video – Video could be applied to projecting actual output onto a 3D UML stereotype of an "interface" class, to depict the computer screen runtime object. Audio could be applied in terms of data sonification. For example, audio sounds could be applied to class objects in digital telephone exchange software, allowing telephone technicians to detect faulty call switching by sound, in the same way they did in the past for mechanical telephone exchanges.

User interaction – This can be utilized to allow the user to manipulate the software modelled during development, in real-time. If modelling a complete runtime environment, user interaction can be used to model the interfaces to the software system before they exist.

User-defined objects – This allows the definition of new objects through a prototyping mechanism. This allows a library of UML objects to be created and, because a object's interface gives a point of abstraction, it is then possible to experiment with different visualisations based on the same information.

Scripting – This feature allows intelligence to be added to nodes and views. Furthermore, scripting will allow integration with existing software development tools such as compilers, debuggers and existing UML tools.

Networking – X3D is designed to enable shared components (i.e. user created nodes and content) and worlds (the combination of the components into a view). This feature should allow software engineers to collaborate on software both by providing a collaboration environment and by sharing work.

Physical simulation – X3D provides features such as Humanoid animation which, may be, very valuable in (HCI) Human Computer Interaction research. For example a UML "Use Case" may specify the way a user interacts with a system through a sequence diagram, through the same specification, humanoid animation may be able to bring that sequence definition to life.

Individually the above features can be seen as adding benefit to software visualisation. Combined they represent a powerful set, that if functioning as specified, would allow many combinations of features to be experimented with.

The X3D specification is designed to be platform and implementation independent, enabling it to be used across a wide variety of development environments. Furthermore many 3D tools can export content to VRML and X3D, allowing complex visualisations to be created and imported.

4 X3D Issues for Software Visualisation

In heading towards X3D-UML, the reality of virtual reality revealed some well known issues plus others more specific to software visualisation..

4.1 The Browser Minefield

Despite 10 years of maturity the VRML/X3D technology still struggles to deliver to expectations. Mark Pesce [2004], VRML co-inventor, noted that even now "potentially useful projects are hamstrung at a fundamental level because the basic VRML 'player' … hasn't matured much (to be brutally honest, at all) in the last decade." This was very much our discovery while investigating if UML could be rendered at all in X3D. Many of the advantages noted previously for X3D, are not only advantages in theory for software but in theory for all X3D visualisation. It should be noted that what is presented in this paper is not a definitive implementation but an exploration into what is possible in today's environment.

The first problem is that, depending on the computer system, many browsers just don't work. Problems can vary from the browser crashing before doing anything, to blank screens or partially rendered worlds with no error message to indicate why.

The second problem is that if they do work, they have an incomplete implementation, since the X3D specification has only just been accepted. Further to the problem, different browsers implement different aspects of the specification.

The result is a browser "minefield", where getting to the other side, is a process of careful trial and error, using different browsers and different X3D features. The problems were found to be more evident for software visualisation due to the use of features that are not commonly used in traditional 3D rendering.

Fortunately the browser technology is under active development and the X3D technology is rapidly improving. Our research has found that the visualisations presented in this paper were not possible, with any browser on our system, till late 2004.

For this research, we found that the browser, which was best suited to the system that we used, was BS Contact 6.2 (www.bitmanagment.de). This did not implement features such as 2D or Java and reported errors during loading of some worlds. Despite this, it provided the most complete set of features and was the most predictable plug-in found. However, we note that other browsers have since released upgrades and may have different behaviour on different systems.

4.2 Describing 3D Software Visualisation with X3D

As stated previously, X3D is designed for defining interactive web- and broadcast-based 3D content. It is not designed with software visualisation in mind and when applied in this way some limitations and idiosyncrasies were uncovered.

X3D has a limitation on user defined types. With user defined types, an X3D content creator, can combine X3D nodes to create new types. For example a user defined "chair" could be created by combining box nodes. However, there is no "software" node defined in X3D or a node where we can directly store and retrieve source code style text. There is a script node, but this not directly readable in the same way a text node would be.

X3D, as with VRML, comes with primitive types, such as box, sphere and cone. At first glance these appear ideal for a quick and easy visualisation (and are included in many VRML "helloworld" examples). However, for software visualisation using them introduces limitations because they do not have the ability to

change at runtime. For "real-time" UML software visualisation, the objects will change in many more ways than options provided by transform nodes, which control the size and shape of primitive nodes.

5 Transforming Java into X3D-UML

In adding 3D visualisation to the software development current workflow, we use the source code itself as the means of generating the visualisation. Source code is designed for explicit program specification, and by creating such a close coupling between the source code and the visualisation, we ensure that as the source code changes, so does the visualisation. Furthermore we provide an evolutionary path to the current workflow. The software engineer can continue to work editing source code, as before, but now has the option to view the code as UML through X3D.

The issue of how to extract the information from the source code, that we wish to visualise, was found in JavaML. JavaML was created by Badros [2000] and uses the compiler itself to generate an XML marked up version of the code. By starting with JavaML we can easily extract information regarding class, attribute and operation names.

Badros [2000] demonstrated that JavaML can be transformed into HTML through the use of XSLT (eXensible Stylesheet Language: Transforms). Polys [2003] demonstrated that XSLT can also transform one form of data, CML (Chemical Markup Language) into an X3D visualisation. Hetherington and Scott [2004] show us that X3D can be extended through the use of XSLT, such as adding the dimension of time to a scene. By starting with an XML based representation, we have also found that we can transform source code from JavaML into X3D-UML using XSLT.

X3D-UML itself is a user defined library which makes use of the X3D prototype semantics. The initial "X3D-UML.x3d" definition file contains merely a standard UML class view "X3DUML2DClassview" whose prototype has fields for name, attributes and operations. Based on these, the prototype generates the UML visualisation, with bordered compartments that display the elements.

Finally the software system can be represented by an X3D world which includes, through the X3D inline feature, all the class X3D files. Through these steps a complete software system can be transformed from a textual representation into standard UML, existing in a fully immersive Web3D environment.

The steps taken in transforming Java code into X3D-UML are summarised below and graphically depicted in Figure 3.

1) Java source code is transformed into JavaML

2) XSLT is used to extract information from JavaML and to generate an X3D file, which contains information pertaining to the UML visualisation.

3) SoftwareSystem.x3d contains the classes of a system inlined into an X3D world. Once loaded the class visualisations are displayed as defined within X3D-UML.x3d

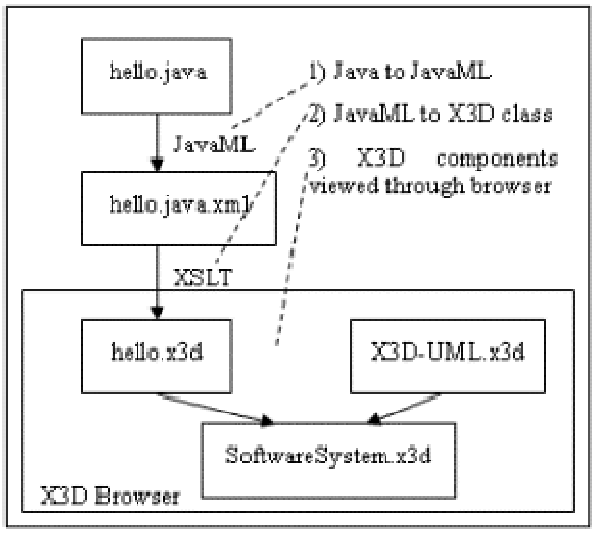

Figure 3 - Transformation from Java to X3D-UML

At this point it is important to note that each step is independent and open. The transformation process is designed to allow evolution of visualisation outside the control of any one tool. Java could be replaced with another programming language or another abstract representation, as long as the appropriate information can be extracted. Different aspects of information can be visualised through changing the XSLT file, and different visualisations can be applied to the prototype. At the SoftwareSystem.x3d level, users are open to changing layout and introducing additional functionality.

6 UML Visualised in an X3D Environment

On the other side of the browser minefield we now see how source code can be visualised, in a standard way, through X3D-UML. If we first look at a trivial example of a single class (as show in Figure 4), it is instantly recognisable as UML. The only visual clue to its new environment is the perspective view given when rotated.

The class displayed has all the benefits of any other object represented in X3D. If we consider the box primitive, it can be viewed via the web, actions can be associated to it when it is touched, it can be displayed in a multi-user environment, different users can view it from different perspectives and it can be animated. These and other benefits are also available to this UML class object, through no, or relatively minimal, programming effort.

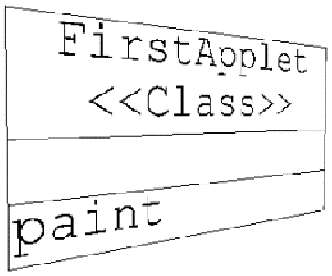

Figure 4 – FirstApplet.x3d displayed through an X3D browser.

The major benefit though, as with the UML, is when it comes to comprehending large software systems. Even without any form of UML representation, it would not be difficult to understand the function of a single class, with only one operation. However, an X3D-UML class, like the box primitive, can be combined with other nodes to create a more complex system. One benefit of this new environment may be to render entire software systems in one scene, and for this to occur we need to first show that it is possible to render systems as large as those found in reality.

Our research found that in version 6.2 of BS Contact, with the performance option set to "Use textures for text" we were able to render such scenes. Examples of a large existing software system can be found in figures 5 and 6. Here we can view the complete Java 3D classes, from the Sun's Java 3D Core API code base (j3d-core Project Home [2004]). We were able to generate this visualisation directly from the source code and demonstrate that, without the aid of scene optimisation techniques, over 700 UML classes can easily be rendered and navigated effectively using a Web3D environment. Prior to version 6.2 of BS Contact, our experiments showed that only at most 200 UML classes could be rendered and navigated effectively, the number now is more than 2000.

Although only simple UML is used (as yet no associations have been visualised) and only simple X3D features, it is clear that web based projects such as Java 3D would benefit from UML, through X3D-UML visualisations. Through a web browser plug-in we are able to see the complete project as UML, without needing to download the code and import it into a specialised tool.

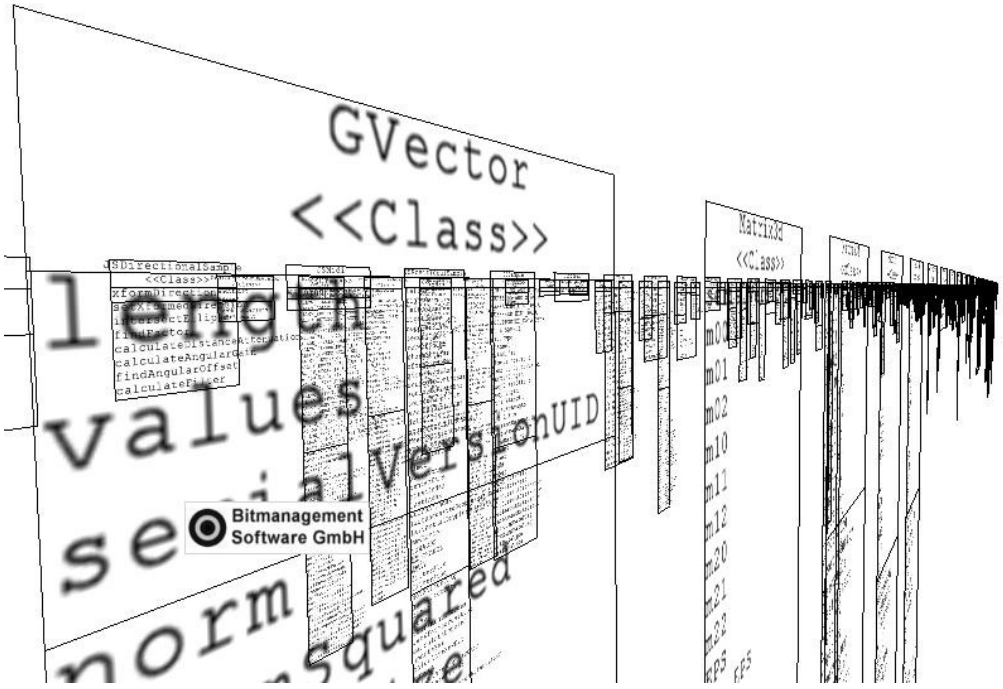

Figure 5 – Classes from the Java 3D packages shown as UML through an X3D browser. Classes are grouped in planes based on the package (directory structure) in which they are found. GVector (front) is from the vecmath package, the other planes show j3d-core-utils and j3d-core classes.

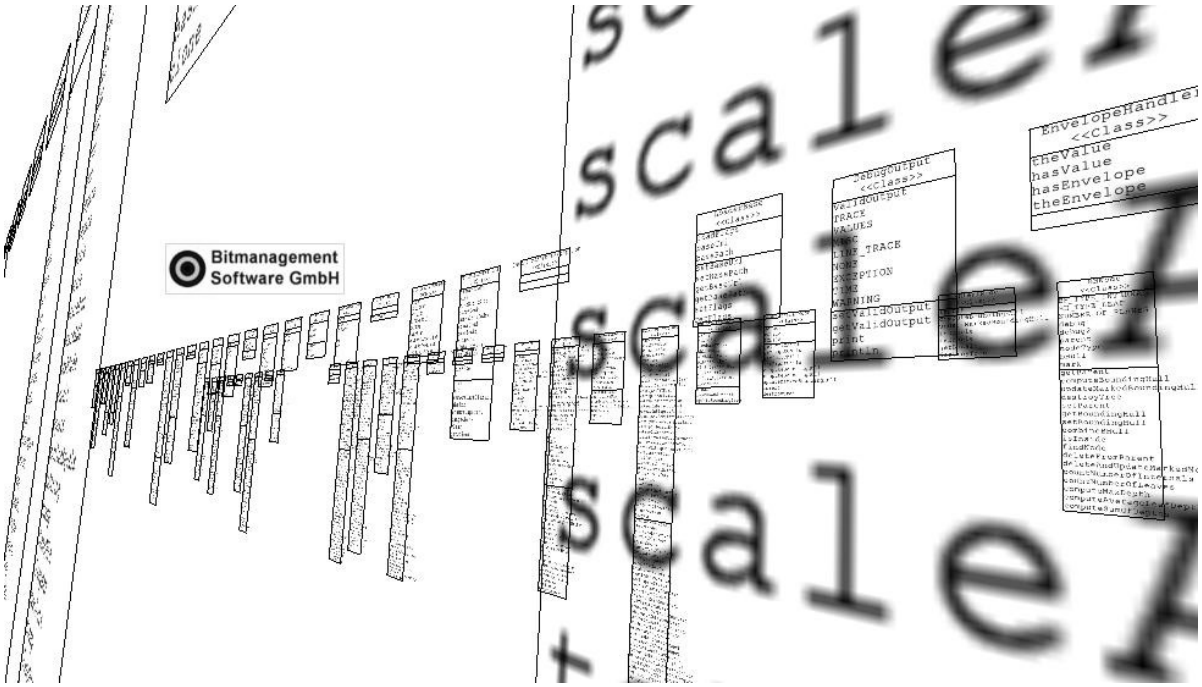

Figure 6 – Classes from the Java 3D package. This is the same scene as shown in figure 5, but viewed from a different location.

7 Summary

We have demonstrated that very recent advances in the support for X3D have enabled advanced UML visualisation to become a reality within this 3D environment. X3D scenes usually model the real world, where hundreds of unique text nodes are unlikely to be found. Despite the unique properties of the UML, we find that X3D is capable of effectively modelling such content.

Furthermore, we demonstrate that X3D is able to effectively render large software systems, as UML, of the sizes found in real systems. We are able to visualise a software system of many hundreds of classes without any optimisation techniques, such as partial scene loading.

Not only is X3D capable of presenting the data, it is able to do so in a way that will facilitate exploration into advanced UML visualisation. We have demonstrated that XSLT can be used to transform one form of XML source code information into X3D-UML. XSLT can therefore be used in a similar fashion with other XML data such as XMI. As the X3D-UML visualisation is merely a user defined X3D type, there is no restriction to redefining the visualisation to explore new areas.

Finally X3D provides a modelling environment with many more features than those provided with traditional UML visualisation. Although we have only demonstrated traditional UML in a 3D space, we are now able to evolve the visualisations to take advantage of the many new features available. The visualisations possible are only limited by the X3D specification and the capabilities of the browser.

Further research is planned into the possible benefits of advanced UML visualisation in the key strength areas of X3D. As X3D is a Web3D technology we plan to explicitly research the areas of networking and 3D. More complete UML notation will be created and its use tested in both a collaborative online Web3D environment (many people sharing the same view remotely) and an immersive 3D environment (many people sharing the same view in the same location).

The artefacts of this (and future) research will be made available via the website www.x3d-uml.org.

8 References

BADROS, G. J. 2000. *JavaML: a markup language for Java source code*. Proceedings of the 9th international World Wide Web conference on Computer networks : the international journal of computer and telecommunications networking, Amsterdam, The Netherlands, North-Holland Publishing Co.

BOOCH, G., J. RUMBAUGH, et al. 1998. *The Unified Modeling Language User Guide*, Addison-Wesley.

DIEHL, S. 2002. *Software Visualization - Chapter 5 - Future Perspectives*. Software Visualization International Seminar, Dagstuhl Castle, Germany, Springer.

DWYER, T. 2001. *Three dimensional UML using force directed layout*. Australian symposium on Information visualisation, Sydney, Australia, Australian Computer Society, Inc.

FEIJS, L. and R. D. JONG. 1998. "*3D visualization of software architectures*." Communications of the ACM 41(12): 73-78.

FISHWICK, P. A. 2000. *3D behavioral model design for simulation and software engineering*. Proceedings of the fifth symposium on Virtual

reality modeling language (Web3D-VRML), Monterey, California, United States, ACM Press.

FRANCK, G., M. SARDESAI, et al. 1995. *Layout and structuring object oriented software in three dimensions*. Proceedings of the 1995 conference of the Centre for Advanced Studies on Collaborative research, Toronto, Ontario, Canada, IBM Press.

GIL, J. and S. KENT. 1998. *Three dimensional software modelling*. Proceedings of the 20th international conference on Software engineering, Kyoto, Japan, IEEE Computer Society.

GOGOLLA, M., O. RADFELDER, et al. 1999. *Towards Three-Dimensional Animation of UML Diagrams*. UML'99 - The Unified Modeling Language. Beyond the Standard. Second International Conference, Fort Collins, CO, USA, Springer.

HETHERINGTON, R. E. and J. P. SCOTT. 2004. *Adding a fourth dimension to three dimensional virtual spaces*. Proceedings of the ninth international conference on 3D Web technology, Monterey, California, ACM Press.

J3D-CORE PROJECT HOME. 2004. *https://j3d-core.dev.java.net/*. 2004.

JACOBS, T. and B. MUSIAL. 2003. *Interactive visual debugging with UML*. Proceedings of the 2003 ACM symposium on Software visualization, San Diego, California, ACM Press.

MALETIC, J. I., A. MARCUS, et al. 2001. *Visualizing Object-Oriented Software in Virtual Reality*. Ninth International Workshop on Program Comprehension (IWPC'01), Toronto, Canada, IEEE, Inc.

OBJECT MANAGEMENT GROUP. 2002. *OMG XML Metadata Interchange (XMI) Specification.*, Object Management Group, Inc.

OBJECT MANAGEMENT GROUP. 2003. *Unified Modeling Language: Diagram Interchange - version 2.0.*, Object Management Group, Inc.

OBJECTS BY DESIGN. 2004. *UML Products by Company http://www.objectsbydesign.com/tools/umltools_byCompany.html*. 2004.

PESCE, M. 2004. *VRML: The First Ten Years http://www.3d-test.com/interviews/mediamachines_2.htm*, 3DTest. 2004.

POLYS, N. F. 2003. *Stylesheet transformations for interactive visualization: towards a Web3D chemistry curricula*. Proceeding of the eighth international conference on 3D Web technology, Saint Malo, France, ACM Press.

THADEN, U. and F. STEIMANN. 2003. *Animated UML as a 3d-illustration for teaching OOP*. ECOOP 2003 - Object-Oriented Programming. Proceedings of the 17th European Conference, Darmstadt, Germany, Springer.

WALSH, A. E. and M. BOURGES-SEVENIER. 2000. *Core Web3D*, Prentice Hall PTR.

WARE, C., D. HUI, et al. 1993. *Visualizing object oriented software in three dimension*. Proceedings of the 1993 conference of the Centre for Advanced Studies on Collaborative research, Toronto, Ontario, Canada, IBM Press.

WEB3D CONSORTIUM. 2004. *Extensible 3D (X3D) - ISO/IEC FDIS (Final Draft International Standard)*, Web3D Consortium.