A Decade of NML Networked Graphics

John A. Stewart  
CRC Canada  
3701 Carling Avenue  
Ottawa, Canada  
+1 613 998-2079  
Alex.stewart@crc.ca

Sarah Dumoulin  
CRC Canada  
3701 Carling Avenue  
Ottawa, Canada  
+1 613 990-5242  
Sarah.dumoulin@crc.ca

Sylvie Noël  
CRC Canada  
3701 Carling Avenue  
Ottawa, Canada  
+1 613 990-4675  
Sylvie.noel@crc.ca

ABSTRACT
This paper is a summation of a decade of support for X3D, human-computer Interaction, and networked graphics that occurred at the Networked Media Laboratory, Communications Research Centre, Canada.

Categories and Subject Descriptors

General Terms

Keywords
X3D, FreeWRL, CRC, NML, Multicast, Peer to Peer, Virtual Reality, Shared Virtual Worlds.

1. INTRODUCTION
As the 20th century came to a close, the Networked Media Laboratory (NML) group at the Communications Research Centre, Canada (CRC) held a series of formal and informal meetings to discuss where computer networking and human-computer interaction (HCI) would be in 5, 10 and 25 years. Research directions were laid out as to how the NML could direct research to keep CRC a relevant and capable player in the future of the Information Age.

2. PATH TO THE FUTURE
Through previous participation in the European Commission's 4th, 5th and 6th framework projects relating to distributed networking (projects Meccano\(^1\), MERCI\(^2\) and MICE\(^3\)), and participation in the Canarie-funded project "Distributed Interactive Virtual Environment (DIVE) over CA*net II\(^4\), the groundwork was laid for NML research activities relating to 3D computer graphics and distributed/augmented reality (AR). Knowledge gained from participating in these EU/Canarie projects led to the first full NML project: MVIP-II. Subsequent projects and HCI studies would be run both entirely by CRC and with external participation.

At that time, (circa 1999) the NML group reached the conclusion that human interaction with networks, and the computers and protocols themselves would substantially change. From that basis, the following areas of interest were mapped out for further research:

Computer Interface - Mice and keyboards would become depreciated; interfacing would be accomplished by other means such as touch, sound, and movement. The computer would fade from being a prominent visual part of daily life and become ubiquitous and invisible;

Software Usability - These new technologies would need to be usable by people everyday, as communication becomes increasingly facilitated via computer;

Computer Graphics - Graphics would become the de-facto visual interface to computer networks; keyboards, mice and ASCII character displays would disappear, and all visual network interfacing would be accomplished via highly interactive graphical interfaces;

Specific Graphics Rendering Issues – Transforming theory into reality would require continuous effort as computer rendering capabilities continue to evolve;

Networking Protocols - Ad-hoc interconnectivity and real-time interactions would become the norm. Internet Service Providers (ISPs) would become commodity brokers and all interesting networking tools would be enabled by applications running on client machines.

\(^1\) http://www-mice.cs.ucl.ac.uk/multimedia/projects/meccano/  
\(^2\) http://www-mice.cs.ucl.ac.uk/multimedia/projects/merci/  
\(^3\) http://www-mice.cs.ucl.ac.uk/multimedia/projects/mice/  
\(^4\) http://www.sics.se/dive/
We will discuss below the NML’s approach to these areas.

3. COMPUTER INTERFACE

Our approach to research in this area came from following the evolution of HCI. Starting with punched cards and tapes, HCI had moved to cathode-ray tube based displays and typing, then on to mouse-based computing. We believed computer interfaces would continue to evolve beyond mouse-based computing.

3.1 Novel Hardware Solutions

NML personnel investigated various hardware-based interaction techniques. The following is an abridged list of what was accomplished:

**Flying Chairs**: Trials with magnetic trackers and head-mounted displays (HMD) and physical walking/crouching/rotating were partially successful. Physical walking in a room with a HMD led to collisions with real objects within the room that were not visible in the displayed virtual world. This was of minor consequence, however, as electromagnetic interference and resolution of magnetic tracking systems led to small, cyclical movements of the displayed virtual world in the HMD, which would induce motion sickness in participants. Furthermore, sensitivity to electromagnetic sources severely limited possible locations for use of these technologies. An additional conclusion reached from the initial studies was that one was “tethered” to one location in the virtual world, making exploration beyond a limited scope difficult.

To overcome these problems, the NML developed two "flying chairs" (Figure 1) in which participants could fly through a virtual world on a virtual magic carpet. Navigation through the world was accomplished by using a joystick, while magnetic tracking allowed one to turn one’s head and look around (or down at the carpet one was flying on). Navigation with the joystick was bounded to keep the user’s avatar upright (no roll). Pitch was limited, or set to zero (no inversions when flying, or, if set to zero, one could not leave the ground). Yaw through 360 degrees was possible in all experiments.

This setup was ultimately successful, having been deployed in a transatlantic demonstration between Canada and Denmark.

**Gloves and Wind**: The NML group experimented with glove-based navigation and acquisition of objects in a virtual world to a limited degree. An experiment was also made with a wind machine: participants were queried as to whether physical objects in a virtual world (in one case, a small airplane) creating real breezes helped the feeling of immersion [10]. This will be further discussed in the next section.

**MIDI input, output**: Experiments with dedicated hardware (data gloves, magnetic trackers) highlighted the effort required to interface with and configure these devices. MIDI, a protocol usually associated with the production of music, proved an ideal local bus for sending and receiving information to external physical devices, and for controlling environmental sound effects [14].

Various MIDI devices were procured and connected to the virtual world system. An analogue computer was built in-house from a kit (Figure 2). This allowed easy interfacing of devices as it contained various modules. For instance: analogue voltage (0-10v DC) to MIDI, MIDI to analogue (again, 0-10v DC), and various filters, dampers and amplifiers.

Accelerometers were acquired, as it was thought that one could use such a device as a controller. Unfortunately, after preliminary investigations and interfacing to these chips at the NML, Nintendo released specifications for their upcoming "Wii" system, which was accelerometer, based, leading us to the conclusion that while our theories were correct, competing against Nintendo would not be productive. Work on the NML Accelerometer Input Device was halted.

FreeWRL was also integrated with Reason®, a professional MIDI music composition program, from Propellerhead Software. This interface enabled FreeWRL to be engaged both as a physical MIDI I/O device for Reason, and, to enable professional audio integration with FreeWRL.

Other simple MIDI devices, both in-house and purchased; were used as controllers for movement in a virtual world. An ultrasonic ceiling mounted grid that generated MIDI data was constructed in-house. This system would allow participation in a shared virtual world wherein identification and movement in the world was accomplished solely by body presence and location. Formal experimentation with this system was not

---

5 Propellerhead Software, Stockholm, Sweden; http://www.propellerheads.se
completed before the dissolution of the NML.

![Figure 2. An analogue computer used for MIDI interfacing](image)

### 3.2 Usability Solutions

As we continued to use our collaborative virtual environments, it became clear that there was a paucity of research investigating how to make these environments usable. Aside from the difficulty of creating easy ways to navigate and manipulate objects in 3D, there were the difficulties in communication produced by the use of avatars as intermediaries. NML decided to investigate ways in which to improve interactions for users in a virtual world [17].

The first study explored the impact of haptic stimulation on the feeling of immersion within a 3D world. While there are many types of haptic feedback, we chose to incorporate a simulated breeze into our world, with the goal of increasing users' sensation of being within the virtual world. For this purpose, the NML created an air cannon from easily-available parts. There were four experimental conditions in all: one with no haptic feedback; one where the air cannon was activated whenever the person moved through the world; one where the air cannon was activated whenever a virtual airplane (several of which were in the world) crossed paths with the person; and finally one where the air cannon was activated at regular intervals. Results showed that people felt most immersed in the 3D world when their own movement caused the haptic feedback. However, the presence of a breeze did not have an impact on people's ability to do the task (find a specific building within the virtual world). Finally, there was some anecdotal evidence that at least some of the participants felt that the breeze reduced their feelings of motion sickness [10].

The second study looked at the difficulty in identifying the current speaker in a 3D world. In a traditional social setting identifying the current speaker comes naturally as we use localization of the produced sound along with visual cues to easily locate the speaker. However, when the speaker is an avatar, these cues are either not present or not as strong as in real life. NML studied three different ways in which speakers could be made more easily identifiable: a radar-like display, a graphic “speech balloon” displayed over the head of the current speaker, and a HUD that displayed the name of the current speaker. This experiment found that although all three methods were equally good for speaker identification, a participant's ability to recall the conversation afterwards was negatively affected by the use of the HUD. We posited that the semantic task required of the user in reading the current speaker’s name interfered with their ability to recall the contents of the conversation [9].

As avatars became more complex, designers began adding emotions to avatars in virtual environments. In order to establish a baseline on which to do further experimentation, NML investigated the ability of users to identify emotional content displayed on static and animated avatar faces. We found that identification rates were similar for both static and animated faces, with users most easily able to identify sadness and happiness and having the most problems identifying disgust and the neutral expressions [8].

Elsewhere, the facial expressions of these avatars were being automated by various means (vocal cues, content cues, real time facial tracking) to remove the task of manually changing avatar expressions from the user - which tended to be used sporadically when provided [5,6]. However, the only research we could find that looked at the accuracy of automated emotional recognition found error rates were around 30% [11]. NML researchers wondered what the effects would be when errors occurred in presenting emotional content. Would people be less trusting of avatars that displayed emotions that were at odds with what they were saying? Would they be more forgiving of emotional content errors when dealing with an avatar face than they would be with emotional content errors presented by a photo-realistic human face?

NML’s study on the impact of incongruent emotional information found that conflicts between the emotion presented and the emotion expressed was recognized by participants. However, we found that as long as the emotional valence (emotional tone varying from negative to positive) was the same, people were much more forgiving of errors. Incongruent situations where the valence was incorrect (i.e. a sad emotion for a happy statement) resulted in avatars being judged as less trustworthy, less convincing and less sincere, as well as the emotion being judged less appropriate for the situation. Incongruent situations where the valence was correct (i.e. a sad emotion for an angry statement) were still judged as having a less appropriate emotional expression but were not judged any less trustworthy, less convincing, or less sincere than the congruent case. There were no significant differences between the judgment of avatar faces and photo-realistic faces [7].

### 4. COMPUTER GRAPHICS

Early work at CRC had shown the importance of standardized, open protocols for generating and manipulating 3D graphics. As part of the DIVE project, two VRML viewers were selected for display rendering: “Liquid Reality” by Dimension-X, and “FreeWRL”, a project from Harvard Ph.D. student Tuomas Lukka. During the short lifespan of the DIVE project, Liquid Reality was purchased by another software company, and was effectively removed from market. FreeWRL was open source, and thus could not be discontinued in such a manner.

---

[78x518 to 318x699]
Management of FreeWRL was transferred from Dr. Lukka to CRC, ensuring long term survival.

3D computer graphics gave an alternative method of distributing information over a multitude of differing displays. Using the ISO- standardized VRML set of protocols, the NML’s work in virtual reality effectively became platform independent.

Management of FreeWRL gave a number of benefits that fit within CRC’s mandate;
   - in-depth world-class knowledge of computer graphics design and rendering;
   - ability to prototype and thus actively participate in the ISO standardization process;
   - participation in W3C working groups to help direct the direction of development of the web;
   - keen understanding of past and present graphical systems, enabling better prediction of future trends.

The FreeWRL project also allowed the NML to fulfill, in various ways, CRC’s mission: "To be the federal government's centre of excellence for communications R&D, ensuring an independent source of advice for public policy purposes" [3]. NML’s contribution to CRC’s overall mission is outlined below.

**Open Source:** Management of FreeWRL was acquired from Dr. Lukka, thus starting what was in all probability the first international open-source project managed by the Canadian Federal Government, and most certainly the longest running and most successful open source project led by Canadian government employees. CRC, along with Sun Microsystems, produced open-source Software Development Kits (SDKs) in cooperation with the Web3D Consortium in 2003 and in subsequent years. The FreeWRL software, containing references to both CRC and Industry Canada, has been downloaded from various sites. While total download counts are impossible to ascertain because of the nature of the distribution of free software, download counts of well over 1 million copies are highly probable. FreeWRL was also distributed by other groups while still keeping the CRC and Industry Canada references; most notably by Apple Computer. Simple tracking of web hits via Clustrmaps show that FreeWRL did indeed have a worldwide presence (Figure 3).

**ISO and W3C Standardization Process:** FreeWRL renders the VRML/X3D series of ISO-standardized protocols, as defined and controlled by the Web3D Consortium and has been used in the conformance testing process by the Consortium [15]. By being an active player in Web3D, CRC gained knowledge and increased its reputation in international standardization bodies. Most recently, CRC was, via the Web3D Consortium, an active player in the W3C HTML5 working group, and had presented a Javascript based X3D viewer running in a prototype HTML5 browser at the 2009 W3C Technical Plenary.

**GPU programming:** It became evident in the early years of the NML that not only would graphics rendering become the norm for screen-based displays; but also that the cost of size of components needed to render graphics would be reduced while their capabilities increased. FreeWRL evolved to use hardware-based graphics rendering for many operations. Key among these were the implementation of the X3D Shader Component and hardware-based Occlusion Culling and Visibility Sensor operation. In general, there was a movement of increasingly large parts of the code base to run directly on the graphics hardware. NML was in the process of moving part of the code base that used older versions of the OpenGL Applications Programming Interface (API) to newer versions, such as OpenGL ES 2.0 and OpenGL 4.0. These newer APIs relied increasingly on GPU programming to render scenes. The move to these APIs would have allowed FreeWRL to run on today’s smartphones, most notably Apple’s iPhone. A port for the Apple iPhone and iPod touch was being completed at the time of the dissolution of the NML.

**Worldwide research-centric dialogue:** FreeWRL was used on many projects worldwide. Communications from users indicated that the code was in use at many universities and research institutions for rendering serious graphics, including machine-vision, micro-interaction projects (e.g., molecular bindings and nuclear decay) and the visualization of mathematical models. By supporting these various projects NML researchers engaged in dialogue regarding research aims with personnel at various institutes, such as CERN, NASA, and various other Research Institutions and Defence Agencies worldwide.

### 4.1 Specific Graphics Rendering Issues

Much effort has been spent in managing data within FreeWRL. Scene graph optimizations and threading support will be discussed here.

As originally written, FreeWRL fully traversed the scene graph multiple times per rendering loop. Such behavior is inefficient, as data is referenced and OpenGL calls processed even when no visible output would be apparent.

**Information Bubbling:** Scene graph tree pruning is necessary on models with a large number of individual nodes in order to optimize rendering speeds and CPU usage. It was decided to keep a linear list of X3D nodes, wherein each and every node knows the quantity and exact location of each parent. Additionally, nodes that can have children know the quantity and location of each child. Thus, we could reference all nodes linearly, and hierarchically as either parent-sibling, or sibling-parent.
Such information was used to provide intelligence to enable efficient traversal of the scene graph. Information regarding children that were sensitive, geometric, geometric with alpha, viewpoints, and currently visible was collected. Further work was envisaged that would have allowed automatic detection of static data, thus further increasing potential rendering efficiencies.

For instance, in the following scene graph:

```
DEF GR1 Group {
    children [
        Shape {
            Material {diffuseMaterial _ transparency 0.5}
            geometry Box {}
        }
        Shape {
            Material {diffuseMaterial _}
            geometry Cone {}
        }
    ]
}
```

The X3D Group node “GR1” would know that it had two child nodes – one solid and one with alpha transparency – no sensitive nodes, no lights and no bound viewpoints. Additionally, GPU-based Frustum culling would indicate whether any of the Shape children were currently visible. This information was used to selectively prune branches from the scene graph during each distinct phase of the scene graph rendering process.

Further work was under way that would have allowed streaming of vertex and material information for this node, as the scene-graph below this Group was by default static (no DEFs and no USEs, thus no ROUTES possible). Such information could be used to reduce GPU-CPU data exchange.

**GPU usage for Occlusion Culling:** GPU-based occlusion culling was enabled automatically if the underlying hardware/OpenGL version supported such a method. Occlusion testing was performed on actual geometry; thus geometry that was totally obscured, but within the viewing frustum, would be removed from the rendering phase. As culling information was garnered from actual visual on-screen rendering, occluded geometry was rendered periodically to see if its occlusion status had changed. A timer was introduced to help balance visual acuity with optimal occlusion pruning.

Aggressive scene graph pruning by use of rendered geometry for culling determination, while beneficial for performance, exhibited minor visual artifacts. Software-based frustum culling using a conic projection, for instance, can in result false positives wherein geometry close, but outside of the rendered visual frustum can be drawn. Alternatively, using the GPU for culling gives delayed determination, leading to shapes “popping” in to the scene when there is rapid movement of either the viewer or the geometry. Additionally, with GPU culling, one receives a count of rasterized fragments for each shape. Shapes with minimal but non-zero fragment counts can be discarded, but subsequent redraw attempts will show the geometry, leading to “blinking pixels” if the raster count pass/fail threshold is chosen too aggressively.

**Shader selection for FillProperties:** The X3D specification outlines certain fill properties for geometric shapes. Such properties were drawn with a GLSL Shader, so that the property requested would be mapped onto the selected geometry.

**Collision and texture threads:** During the rendering phase, the scene graph was static – all node information was read-only. FreeWRL allowed collision with the actual faces of each shape in the rendered world; thus giving the viewer a collision vector that was based on the actual geometry that was hit. Work was underway to multi-thread this phase, such that rendering of visuals and actual collisions with the geometry were done in differing threads, leading to increased frame rates on multicore machines.

Texture loading, and scaling/shaping for specific platform requirements, was performed in a separate thread. This allowed rendering to continue while textures were loaded. Actual threaded binding of texture data to OpenGL texture objects was problematic across all platforms and was therefore only utilized on specific platforms.

**Shader selection for OpenGL ES 2.0:** OpenGL ES 2.0, and OpenGL 4.0 dispense with fixed material properties. As information pertaining to the state of material properties was available during the Shape rendering phase, actual shader selection for a specific set of properties was possible. This would have increased basic throughput, as simplified shaders could have been chosen. It would have also increased visual acuity, as differing shading algorithms could have been chosen according to specific material properties. Work was underway to use computationally intensive Phong shading on shapes without textures, and Gouraud shading on textured shapes. Examples of FreeWRL running these two shading algorithms are available.

**NormalInterpolators ColorInterpolators and Geometry Shaders:** It was expected that as GPU use increased NormalInterpolator and ColorInterpolator interpolation would be executed on the GPU. This would have led to less data exchange between the GPU and the CPU. All information required for interpolation (ROUTEs, etc) was available at runtime; thus decisions about where to run these interpolators could have been made as needed.

X3D contains some basic shapes; Box, Cone, Sphere and Cylinder. FreeWRL kept track of the linear distance between the viewer and the Shape. In this way, rendering acuity could be tailored to suit distance, as with a Level of Detail (LOD) node. Unfortunately, Geometry Shaders were not widely supported at the time of NML’s investigation in 2009, so Geometry Shader work was placed on hold.

---

6 http://www.youtube.com/watch?v=igz08SVhUhw
Polyrep Structure: All meshes, with the exception of the primitives Cone, Cylinder, Box and Sphere, were converted into a common structure: the Polyrep. This allowed for improved code sharing, as all collision and rendering algorithms are common. It also allowed easy prototyping of newer shapes, such as the Geometry2D nodes.

X3D has some nodes that correlate closely to OpenGL primitives; namely the Triangle shapes (e.g., TriangleFanSet). It may at first appear that the complexity of handling these nodes increases when using a common structure because information is generated that need not be used. But, in retrospect, converting these simple nodes into Polyrep structures not only allowed for increased code reusability (and thus reliability) but also enabled handling these nodes on versions of OpenGL that do not directly support the specific type of triangle packing and index packing that was envisaged when these X3D nodes were constructed.

For instance, OpenGL ES 2.0 has removed support for TRIANGLE_STRIP operations and for directly supporting the X3D Index field. Removal of X3D MFInt32 in OpenGL ES has affected the ability to directly reference the X3D MFInt32 type fields in the OpenGL ES DrawElements function. Only UNSIGNED_BYTE and UNSIGNED_SHORT types are possible. To properly support the X3D TriangleStrip and all of the IndexedTriangle nodes required some form of reinterpretation.

5. NETWORKING PROTOCOLS
Following their participation in the MERCI and Mecanno projects NML decided that a virtual environment would provide a better match in terms of available toolsets for hosting remote collaboration than traditional videoconferencing tools. Meetings held in a virtual environment would forgo the video feed (often little used) in exchange for avatar location information and a shared view of virtual objects. Avatar location information could be used in conjunction with audio conferencing software in order to produce proximity-based audio.

At that point in time, research was ongoing in the realm of multicast communications. It was felt that bandwidth hungry applications such as conferencing and streaming media needed to move away from a client server model to a peer-to-peer model if the number of participants was to be scalable. Furthermore, real time applications such as voice conferencing were proving to be very sensitive to network latency. Client server architectures were in a markedly disadvantaged position in this regard as data had to travel effectively double the distance in order to be propagated between peers. Processing delays at the server compacted the problem. Peer to peer protocols were seen to be part of the solution to achieving real time conferencing.

IP multicast had been proposed as a peer-to-peer architecture that could be implemented by network routers. Support for IP multicast was optional, but the seemingly inevitable rollout of natively multicast IPv6 would surely make it popularly available in the future. Thus, NML developed the MVIP protocol. This protocol took the VIP protocol (an existing client server protocol for networked virtual environments) and reformed it to use IP multicast. Proximity based audio conferencing was also added to the protocol through the use of a version of the multicast Robust Audio Tool (RAT) modified to take into account speaker location [12].

MVIP was tested and improved on at the NML. It was decided to replace the RTP (real time protocol) with the RTP/I protocol - a version of RTP that was specifically tailored for interactive applications; i.e. applications where it is usual for all participants to be both sources and receivers of data simultaneously. Changes were made to the protocol to add avatar actions, object interactions, and computer controlled avatars. This version of the MVIP protocol was released as MVIP-II [13].

However, testing with the MVIP protocol suite, while easy enough in a lab situation was crippled by the lack of widespread uptake of IP multicast in the real world. Although tests were done between Ottawa and Honolulu and Ottawa and Europe it was necessary to use IP multicast tunnels between all locations. These were not only a large configuration problem for the user, but also meant forgoing any bandwidth savings that could have been realized through the use of non-tunnelled multicast. IPv6 proved not to be a success and IP multicast was not supported by most IPv4 networks. It appeared that IP multicast was not a feasible solution for peer to peer outside of the research environment.

In the intervening time end system multicast (ESM) protocols had been developed as an alternative to IP multicast. These protocols were not as efficient as IP multicast but required no special software at the router. It was decided to move MVIP to work over ESM. Research into available ESM protocols found that our options were limited. Most ESM protocols are developed either for latency insensitive applications (such as file sharing) or for situations in which there is only one person disseminating data to a large group of receivers (i.e. broadcasting). Neither of these would suit a shared virtual environment.

One protocol was found that met our requirements: ORTA [16], which is an implementation of the Narada protocol presented by Chu, Rao, and Zhang in [2]. ORTA had already been successfully tested with RAT. NML modified its MVIP-II protocol to use ORTA to transmit both audio and virtual environment data in a peer to peer fashion. This version of MVIP was renamed FreeXoo and was released as an open source software project on sourceforge in 2010.

6. LESSONS LEARNED
Rapid evolution of hardware 3D graphics: At the time of the formalization of the NML, in-house experience with 3D graphics was minimal. The NML acquired a Silicon Graphics Onyx computer from a project that had just been completed (Figure 4). Initial versions of FreeWRL and the MVIP-II protocol ran well on this computer, while rendering on Linux-based PCs was, in comparison, poor. Within two to three years, however, its Linux-based brethren outclassed the coffee table-sized and expensive Silicon Graphics computer. At the time of writing, one could argue that the 3D graphical capability of inexpensive mobile devices, such as Apple’s...
computers to mobile devices. This ability provides a simplified graphics scene that can render on any hardware, from supercomputers to mobile devices. It is referred to as "write once, render anywhere". In theory, and in practice, X3D, as an XML-based format, inherently supports this adaptability to hardware.

As time progresses, failure, shader compilation failure is more likely to occur as the underlying data encapsulated in those graphics may have been modified or lost. Therefore, shader languages need to be written in a way that ensures they are compatible with all hardware, and that any changes in hardware do not affect the rendered output.

CPU clock rates will not increase: Many graphical applications, especially games, are written in a style that works best on a single computing thread. A decade ago, single processor machines were the norm, and clock speeds were increasing rapidly. While one could have expected that trend to continue, the opposite actually happened: clock speeds decreased and multi-core CPUs became the norm. Having multiple cores in a CPU emphasizes peak computational throughput over software simplicity. Developers must write software to take advantage of the CPU by utilizing multi-threading and data-mutex locking. Distributing tasks amongst multiple cores was implemented in FreeWRL. Work in progress further subdivide the rendering process amongst cores and to further load the GPU in order to fully utilize the inherent capabilities of the hardware system.

Adaptability to hardware: X3D is a "write once, render anywhere" graphics format. In theory, and in practice, an X3D graphics scene can render on any hardware, from supercomputers to mobile devices. This ability provides a simplified process whereby authors need only to release one version of their 3D Data and it will be rendered on any device from any manufacturer.

A current trend in state of the art computing is to expose underlying hardware details and effectively force graphics authors to tie their 3D data to specific hardware and underlying low-level graphical APIs. WebGL and Collada emphasize shader-based rendering. WebGL in particular will effectively tie the output created by 3D graphics authors to a specific snapshot of hardware capability, which can only lead to effective loss of data (because of the loss of specific hardware) as time goes on.

Abstract the hardware: Over time graphics hardware and APIs change. The X3D language definition is sufficiently abstract that X3D nodes can be mapped onto all hardware-centric APIs, independent of vendor and hardware, with two noted exceptions.

The X3D Triangle node support issues have been noted above. The X3D Shader Component is one visible area where X3D’s mantra of “write once, render anywhere” is broken. Differing hardware platforms and lower layer drivers support differing shader languages. Within each version of each specific shader language, features are added and deprecated. While the Shader Component is written such that fixed-functionality Material properties will come into play in cases of shader compilation failure, shader compilation failure is more likely to occur as time progresses, leading to a degraded rendering of the X3D model.

Serious graphics: The computing press emphasizes gaming when discussing 3D graphics. This is a narrow view of the potential for 3D rendering. It has been observed during the course of a decade of VRML/X3D graphics support that there is a large contingent of people that use VRML/X3D for what we term "Serious Graphics", such as rendering molecular structures, mathematical models, and medical imaging. Such uses require not only cross-platform support, but also long-term ability to access the data encapsulated in those graphics.

Serious graphics written easily: During dialogue with FreeWRL users, it became apparent that X3D is an ideal intermediate language for those who want graphical output of their underlying data. This is of crucial importance; X3D is simple enough that it can be easily understood and created, and X3D input is text based. Users need only write a text stream and feed that stream into an X3D viewer. They do not need to understand the complex nature of OpenGL and other 3D standards, there is no compilation of graphics code, and no platform dependency. While it may be true that such generated graphics probably were not of a quality that one would expect from Hollywood, the graphics were more than adequate to enable visualization and interpretation by those concerned.

3D is the interface: As the world progresses towards mobile computing, 3D graphics has become a single unifying factor in terms of user interaction with networks. The Wireless Technology Roadmap from Industry Canada states that game developers are targeting mobile handsets: “Overall, the game developers see wireless handsets becoming the central point in users’ daily lives for communications and the need for light

iPhone and iPad, easily outperform the Silicon Graphics Onyx computer (circa 2000).
diversion with casual games. The early adoption period is over, their sights are now clearly set on the mass market.” [3]. With 3D graphics arriving on smaller screens it is likely that much of our future interaction with the electronic devices that we use in our everyday lives will be mediated by some sort of 3D graphical display.

7. CONCLUDING REMARKS
X3D is an ideal 3D graphics rendering protocol when used within its application sphere. It is not the only graphics protocol currently in use, but has strengths that will continue to garner support from a wide variety of users.

Unfortunately, as of early 2010 due to resource shortages and other organizational priorities at CRC, the NML program has been cancelled. FreeWRL and FreeXoo are available to the open source community. It is hoped that others will pick up the challenge and continue the development of these exciting technologies. The NML team wishes to thank the community of open source colleagues around the world who have contributed to the project over the last decade.

8. REFERENCES