A transitional framework for multi-user virtual environments

N.A. Goonasekera
Department of Computing
Informatics Institute of Technology, Sri Lanka

N. Devinuwara
Department of Computing
Informatics Institute of Technology, Sri Lanka

ABSTRACT

The creation of multi-user virtual worlds has been an area of much research and interest, yet the promise of having a ubiquitous network of such virtual worlds has still not become a reality. The reasons for this include the lack of flexibility in some solutions which are unable to cope with the highly domain-specific needs of Networked Virtual Environments (NVEs), the lack of commercial viability and user-acceptance of some proposed architectures and a proliferation of proprietary solutions which largely ignore existing infrastructure when delivering content to the end-user.

This paper details a software framework which attempts to address the above issues by devising a transitional approach to presenting multi-user virtual worlds over the Internet, through utilization of the large base of web and application servers already deployed. The resulting proof-of-concept framework provides a base set of services for creating NVEs, including code migration through weak mobility, support for extending communications protocols at run-time and support for voice communications between participants. The framework employs a hybrid client-server architecture written entirely in Java. The client component uses Java3D and the Java Media Framework for providing graphics and multimedia services, and is embeddable as an applet within a standard web browser. The server component is deployable atop an industry standard application server, and utilizes its services for providing a loosely-coupled component based development model. Emphasis has also been placed on avoiding the typically restrictive nature of browser based programming through runtime extension of client capabilities. Analysis of the prototype framework indicates that the suggested architecture is suitable for delivering virtual reality applications in a wide range of domains.

1.0 INTRODUCTION

Traditionally, the greatest impediment to the creation of networked virtual environments (NVE) has been considered to be bandwidth limitations. However, with the rapid spread of Internet connections such as ADSL, the average home user now has access to broadband Internet. Hence, the barriers to the creation of a ubiquitous network of virtual worlds hyper linked to each other are rapidly eroding. In addition, with the notable success of online multi-player games which have displayed favourable scalability characteristics, the feasibility of creating such virtual worlds has also been displayed.

The initial answer from the web community in dealing with this requirement was the creation of the VRML standard. After enjoying a brief period of popularity, it gradually dwindled away from use, due to its monolithic structure and limited abilities in creating interactive worlds. Later, attempts were made to retrofit the VRML standard with support for multi-user interaction, through use of the External Authoring Interface (EAI). Most notably, Blaxxun Interactive [1] and Active Worlds Inc. [2] have created virtual communities which can be delivered via a web browser with a VRML plugin. However, these attempts have proven to be inflexible in nature, complex to program and unable to cope with the highly domain specific requirements of networked virtual environments. Meanwhile, non-commercial research carried out on the creation of NVEs have succeeded in solving many issues, yet tend to be highly proprietary in nature and largely ignore the existing infrastructure when delivering content to the end-user.

This paper describes the creation of a software framework which addresses the above issues by devising a transitional approach to presenting multi-user virtual worlds over the internet. It attempts to identify the services essential to the creation of multi-user virtual worlds, and to implement them by leveraging and extending existing technologies such as the VRML standard and Java3D. Finally, it proposes that objects in a 3D world have no upper limit on their possible behaviour, allowing the creation of arbitrarily complex objects and hence, arbitrarily complex 3D worlds. In what follows, the design rationale of the framework and its architecture are explained brieﬂy. A detailed discussion can be found in [3]. The discussion starts by looking at some
key issues in networked virtual environments. Section 3 and 4 describe the design and implementation of the framework respectively. Finally, some potential areas of research and final conclusions are presented in sections 5 and 6.

2.0 ISSUES IN NETWORKED VIRTUAL ENVIRONMENTS

2.1 Commercial aspects
Research on virtual environments originally started as military funded projects, mainly because of the possibilities in using VR technology for large scale distributed battlefield simulations. After the adoption of VE concepts in games, and later online games, the commercial aspects of VE systems became apparent for entertainment purposes. However, other areas such as collaborative applications, online-shopping and educational applications have not been exploited as widely, despite some significant attempts being made by [1] and [2]. Furthermore, many additional commercial applications do exist, as outlined by [4]. It should also be noted that whilst online gaming environments such as “Battle.net” [5] and “The Sims” [6] have proven their commercial viability, these systems are highly proprietary in nature and are optimized for a narrow domain, making them difficult to extend for general purpose VR applications.

Commercial acceptance is vital for the widespread use of VR, as this would fuel further research and funding into its development. The framework has attempted to take this factor into account by attempting to utilize the large base of existing web servers as potential launching pads for virtual world servers. Issues which are important for commercial acceptance, such as maintainability and control, have also been considered in the design process.

2.2 User acceptance
User acceptance is another key issue in the adoption of VR technology. However, user acceptance has been hampered due to technical constraints such as firewalls and bandwidth limitations which, despite the significant advances mentioned earlier, are nevertheless the dominant issues in the widespread adoption of VR technology. In addition, the fact that most users are already familiar with the use of browser technology also needs to be considered when considering a means of end-user delivery. This prompted a design decision that the client potion of the framework should be embeddable within the context of a standard web browser.

2.3 Graphics representation
In discussing issues related to graphics in virtual environments, it should be noted that client graphical hardware power has continued to increase at an exponential rate, with dedicated graphics processing units available, which are capable of rendering high resolution 3D graphics at well over 30fps in real-time. Hence, it is clear that the issues in rendering 3D graphics in modern day clients are not a matter of great concern - most of them have been successfully addressed. With respect to virtual environments, the communications overhead in downloading the required 3D world data far overshadows the issues in presenting them. Therefore, already well established techniques such as scene graph based APIs were utilized in the implementation of the framework, and no novel solutions were deemed necessary.

2.4 Communications
One of the major goals of this framework is to address the issues in providing a multi-user experience. The multi-user experience is directly related to the communications infrastructure, and therefore communications issues continue to be the major barrier in the creation of virtual worlds. When dealing with this, one of the first issues that needs to be considered is what communications architecture should be utilized. Communications architectures are commonly organized around the following paradigms.

a. Client-server
Client-server architectures are the most common form, and have been used in many existing virtual environments, commercial online servers as well as classical MUDs [7]. Indeed their popularity can be understood, in particular due to ease of administration and management. Further, due to the large existing base of web servers, which could easily be converted to serve virtual world content, the adoption of a client-server architecture had the potential to increase the commercial viability of the framework. However, in most cases, the server quickly becomes the bottleneck to performance [7].

b. Peer-to-peer
P2P architectures have also been used in several VR systems [8]. However, [7] argue that peer-to-peer NVEs are difficult to scale due to lack of hierarchical structure and are therefore more suited for LAN environments. In contrast, [9] presents a different view, and shows that it is a false dichotomy to argue between the two, as a full-spectrum of functionality exists between client-server and peer-to-peer. This hybrid approach is discussed next.

c. Hybrid
Hybrid approaches have been used in systems like NPSNET [10] and have favourable scalability but at the cost of additional complexity [7]. The development of VRTP also attempts to support such hybrid communications capabilities, and several arguments have been
In analyzing the above information, the potential for using existing web servers, along with administrative benefits and greater commercial acceptance, argues strongly in favour of a client-server approach. However, the hybrid approach suggested in research efforts on VRTP cannot be ignored, especially for streaming media content [9]. The suitable strategy therefore, was to adopt a primarily client-server approach, with some support for P2P communications. In the absence of P2P possibilities for some clients, for example due to firewalls and similar impediments, the server would be responsible for ensuring that those clients receive the required information (through tunneling mechanisms).

A similar situation exists for distribution methods. The unicast approach remains the primary means of communicating over the Internet. Both UDP and TCP based unicast approaches have been used extensively in the design of virtual environments such as BrickNet and RING [4]. The primary advantage in unicast approaches is that it is the most widely available method, as opposed to multicast and broadcast. However, [9] point out that it is imperative that multicasting is supported to provide any form of scalability to NVEs. It is also shown that current unicast and broadcast mechanisms fail miserably in scaling to more than a few hundred users. The problem becomes compound when scaling to WANs, where latencies rise higher and reliability levels drop lower. However, the multicast back bone (MBone), which is the means of multicasting over the Internet, is not widely spread and therefore is especially not useful for applications such as MMORPG [12]. Currently, little over 20 countries support MBone [11]. This leaves out a large segment of users, which is undesirable, especially since the success and adoption of the technology is directly proportional to its accessibility. Further, use of MBone must be carefully controlled as it is a limited and scarce resource.

This indicates that the most suitable strategy is inherently dependent on the problem at hand. Virtual worlds requiring heavy use of streaming media content or LSVEs may require multicast capabilities to achieve a reasonable degree of scalability. Simpler worlds or worlds specializing in specific environments may find a unicast approach sufficient. The logical conclusion is that any general purpose VE framework must support both forms of communications to maintain favourable scalability characteristics, and this approach was therefore adopted within the framework.

The next section goes onto describe some of the main design decisions of the framework.

### 3.0 DESIGN ISSUES

#### 3.1 Protocols

When researching into communications protocols, research carried out on the creation of a virtual reality transfer protocol [9] proved to be of great importance. This research has identified four main types of communications that are necessary in the creation of virtual reality applications, which encompass the wide variety of communications needs in virtual environments.

Hence, the traditional approach of a single, monolithic protocol handling all communications needs is no longer viable. This issue has already been realized and systems like Bamboo [13] and NPSNET-V [10] use a pluggable micro-protocol architecture. This architecture allows each object within a virtual environment to communicate with each other using custom tailored protocols, allowing optimizations based on domain specifics. The cross-format schema protocol (XFSP) devised for NPSNET-V [14] allows the dynamic update of the networking protocol in a running virtual environment through redistribution of an XML schema definition.

In addition, optimized and efficient binary protocols for light-weight interactions are almost mandatory to avoid flooding the network with large frequently transmitted packets of data. [15], has devised the Dynamic Behaviour Protocol (DBP) which augments NPSNET-V with hot-swappable binary protocols based on XML schemas.

Considering the above facts, it was considered prudent to adopt as flexible a strategy as possible in dealing with communications protocols. Hence, the framework too supports a pluggable protocol architecture, which can be extended at run-time to support additional protocols.

#### 3.2 Interest management

Area of Interest Management (AOIM) schemes have been used extensively in NVEs, as they can help reduce network traffic drastically. For example, in the SIMNET system [4], an object broadcasts events to all other objects. The receiving object must decide whether or not to process the event. This affected the scalability of the SIMNET system. The NPSNET system, a descendent of the original SIMNET system uses AOIM techniques, which gives it much higher scalability.

Systems such as BrickNet employ a client server architecture [16]. It uses UDP for communication between client and server. Its AOIM technique is for clients to register interest in desired objects when connecting. The server will only notify changes in subscribed objects, thereby saving bandwidth. Another similar client-server system utilizing AOIM
techniques is RING [4] which specializes in densely occluded environments. Therefore, while RING would perform well in a museum walkthrough, its performance would be less than adequate for large open spaces, such as a flight simulator [16]. The interesting observation here is that different AOIM techniques are valid for differing environments, making it difficult to develop an AOIM scheme that is common to any application – the AOIM scheme is inherently dependent on the particular application at hand. In dealing with this situation, it was decided that the framework should adopt a modular AOIM scheme that could be replaced according to each application’s needs.

3.3 Dead reckoning

Dead reckoning algorithms are a means of reducing communications processing, by minimizing updates at the cost of extra computation by the host system. Although computation costs are important, in general Internet scenarios, network latencies far overshadow computational latencies. Hence, this is an acceptable tradeoff in increasing communications efficiency.

For many virtual worlds developed using the framework, the movement of an object can be extrapolated using a simple dead reckoning algorithm, providing great performance gains. For example, a user moving through a virtual building has a fairly predictable movement. Hence, a simple dead reckoning model was employed in order to predict such movements. This is discussed further in the implementation section.

3.4 World behaviour

Different distributed VE systems use different approaches in dealing with the execution of code. Some require that the virtual environments be statically linked and deployed whilst others such as NPSNET opt for dynamic downloading and execution of code [10].

Systems like Avacado [4] take on a hybrid approach, with the basic core application needing to be statically linked with additional support provided for the execution of script based languages.

When analyzing the above data, it is clear that the first approach of static linking is unsuitable for a general Internet scenario. Reasons include portability, as internet users are a diverse audience and hence have a diverse range of operating systems. Also, security concerns are another reason for disallowing execution of arbitrary code that cannot be verified on a client computer.

Hence, it is evident that one of the latter two approaches needs to be considered for the purpose. Once again, a hybrid approach is difficult to implement for the same reasons mentioned above, and the choice is reduced to interpreted languages as a form of downloading virtual world behaviour. Indeed this is verifiable by the currently popular paradigm of script based behaviour execution in standard web browsers. Almost all browsers support the ECMA-Script standard, and till recently, code download in the form of Java applets as well. However, the current browser scripting languages are weak in their expressiveness and restrictive in their capabilities, severely limiting the complexity and flexibility of applications that can be developed using them. Clearly, this limitation needed to be avoided in the framework. Therefore, a more general purpose and powerful means was necessary in order to deal with the diverse requirements of virtual environments. Only one clear choice emerged here – Java, due to its cross-platform performance and expressiveness and it was therefore adopted as the means of executing world behaviour.

3.5 Streaming media

Streaming media remains an area that has been largely ignored in the creation of VR applications. Focus instead, has been given to 3D graphics and communications issues. However, well established solutions for streaming media exist. In particular, the RTP protocol (RFC 1889) has been specifically engineered for delivering streaming audio and video over the Internet in a variety of encoded formats. It is transport protocol independent, adding further appeal to its use. In addition, the Java Media Framework provides extensive support for RTP based streaming media transmission. Hence, this was utilized in providing streaming media capabilities to the framework.

4.0 IMPLEMENTATION

4.1 Overall architecture

Figure 4.1 depicts the overall architecture of the framework. It shows two clients connecting to a virtual world server. The clients utilize Java3D and the Java Media Framework as mentioned previously. The server component runs on top of an industry standard application server. Also shown are some typical communications that take place between the clients and server. Initially, model data and executable classes maybe downloaded by the client using the HTTP protocol, with additional compression using GZIP to further reduce bandwidth consumption. Reliable, connection oriented communications take place using an XML based protocol, which is validated against an XML schema definition. Further, the positions of entities within a world are transmitted using the DIS protocol, with RTP being used for transmitting streaming media.
4.2 Server design
The server component was designed to run on top of a standard application server. The reasoning process behind this was to avoid reinventing services provided by such application servers. These application servers provide well developed services such as database access, load balancing, transaction management, naming etc. Further, application servers are fast becoming the de-facto means of deploying server applications. It is also in keeping with the transitional approach of the framework, as many organizations are already using this technology and developers will need little effort to switch and can build on existing code bases to provide for virtual world functionality. Existing web sites can be gradually converted into virtual world applications without rendering existing code obsolete.

4.3 Client design
The development of the client can be considered the highlight of the entire framework. This is because the design and extensibility of the client generally dictates the evolution of the entire platform. A suitable analogy is the existing browser based paradigm of web surfing. Although browser based web technology has achieved notable success, it is also important to note the fact that no significant improvements (other than some minor graphical improvements) have occurred in modern day browsers. The delivery of multimedia content and truly interactive media via the browser has still not become a reality. This is mainly due to the restrictive and in-extensible design adopted by browsers. The programming ability of a browser is typically limited to a few weak scripting languages that can do little other than manipulate a few graphical objects. This makes evident the need for the technological capabilities of the client to evolve rapidly in order to cope with environmental changes.

This becomes an exceptionally delicate issue for virtual reality applications. At one extreme, the graphical capabilities of client hardware are increasing at a phenomenal yet unpredictable rate. The network bandwidth of clients is also increasing, as more people clamour for the benefits of broadband Internet access. Coping with such change requires that the client also be rapidly extensible, unlike the “frozen-in-time” approaches adopted by current browsers. Hence, the design of the client has concentrated on its extensibility, both in terms of technology and through utilization of several design patterns [17]. Next, some key features of the client are discussed.

a. Embedded execution
The client program supports execution both as an applet and an application. This allows the client environment to be run within the context of an existing browser or in standalone mode, as required. This gives several important advantages in terms of user acceptability. It should be noted however, that the client will require installation of the JRE (Java Runtime Environment), Java3D and JMF (Java Media Framework) in order to execute properly. However, these downloads are freely available and can be installed without any difficulty.

b. Java applet extensions
As another step towards building on existing technology, the framework uses extensions to existing applet technology to support virtual environments. There are a few reasons for this
- Applets run in a sandboxed environment and have tightly controlled security to ensure client safety.
- Many developers are familiar with programming applets, and they will find it to be an easier transition for creating virtual worlds.
Applets can do anything that a standard application can do, except that they simply run in a more secure environment and must follow a standardized life-cycle. It should be noted that this design gives the client browser the additional advantage of being able to execute any standard applet within its context. Writing a virtual world applet is a simple process of extending the Applet3D class. The Applet3D class is in turn an extension of the java.applet.Applet class and provides a few additional functionalities conducive to VR development.

c. Pluggable protocols
The pluggable protocol manager allows the definition of additional protocol handlers for the client framework. The reasons for adopting this approach are explained in greater detail in the design issues section. This too has been built as an extension to the standard Java protocol handling mechanism and allows for the runtime registration and deregistration of protocol handlers.

d. Object registry
The object registry provides a convenient way to extend the file types and formats recognized by the client application. It defines an XML based mapping between object types and their corresponding handlers. The initial mappings are loaded from an XML file name vr-handler-registry.xml. A sample is shown in figure 4.3.

<handler>
  <handler-name>SceneHandler</handler-name>
  <handler-desc>Handles Java 3D scenes</handler-desc>
  <handler-class>lk.ac.ics.vr.handler.SceneHandler</handler-class>
</handler>
<class-mapping>
  <class-name>com.sun.j3d.loaders.Scene</class-name>
  <class-name>SceneHandler</class-name>
</class-mapping>

Figure 4.3 - Extract from object registry

This feature allows the content types handled by the client to be extended dynamically.

e. Dead Reckoning
Figure 4.4 shows the design of the entity update model of a local entity. The local entity is registered with the avatar manager, which is responsible for observing entity state updates. It then consults the dead reckoning service to determine whether the entities state update needs to be transmitted. If so, it informs the AOIM manager. The AOIM manager will determine which entities need to receive the update and use the entity state dispatcher to inform only interested parties, thus saving bandwidth.

5.0 FURTHER WORK

Several enhancements such as the ability to permanently install signed code libraries on the client, more dead reckoning models, support for a world physics engine etc. would be useful additions to the framework. In addition, some interesting possibilities for further research also present themselves. These are described below.

5.1 A network LOD manager
A Level of Detail (LOD) manager that would be coupled with the AOIM manager to allow a user configurable policy for downloading information would be an interesting research possibility. For example, some specific application users might tolerate a reduction in graphics quality provided that the sound quality is high. Similarly, other users might want improved graphics at the expense of sound quality. Bandwidth limitations may also dictate a different policy. An LOD manager that takes these constraints into account when requesting specific resources is of great interest. A network efficient LOD scheme has been implemented in HIVEK [4]. However, this is a fairly simple implementation which only concerns itself with managing model LOD. Hence, the development of a more comprehensive system is open to formal research.

5.2 Global object library
A readymade library of objects could drastically cut down the bandwidth requirements for more complex virtual worlds. These client-side libraries could contain typical objects such as avatars, trees, chairs and other such common objects. Further, they could also be enhanced to generate on the fly objects, such as terrain, more complex variations of forests etc. The library would also grow as new worlds are browsed and thereby further reduce the number of required downloads (Similar to a global cache). These libraries need not be limited to objects alone, and could be extended for high-resolution textures as well. There would need to be the development of a directory
service which would enable an application to query the library for available objects based on desired characteristics.

6.0 CONCLUSION

The aim of the project was to provide a framework for creating arbitrarily complex multi-user virtual worlds. The outcome of the project was the creation of a prototype development framework along with a methodology for tackling the problem of transitioning to a virtual reality experience within the confines of existing infrastructure. Some key findings are the need to adopt a hybrid client-server approach, importance of client extensibility over server extensibility, the need to use existing browsers as a means of delivery and the possibilities in utilizing already available application servers and web servers as launching pads for networked virtual environments.

Some future areas of research are also suggested. These include a client side object library to overcome the limitations imposed by lack of bandwidth and a network efficient LOD scheme for increasing performance by defining a user configurable scheme for controlling the fidelity of audio and 3D model information received.

Many bottlenecks have been encountered and breakthroughs made in this field yet the transition to web based virtual reality has not yet occurred. Despite the significant research efforts made in this area, VR over the web has become stuck in a rut and there is still a long way to go before VR systems become as commonplace as web sites. It is hoped that the ideas presented herein will provide some traction for moving forward.

7.0 REFERENCES