Predictive Interest Management: An Approach to Managing Message Dissemination for Distributed Virtual Environments

Graham Morgan & Fengyun Lu

School of Computing Science

Newcastle University, Newcastle upon Tyne, NE1 7RU, UK

Telephone: + 44 191 222 7983, Fax: + 44 191 222 8232

E-mail: Graham.Morgan@newcastle.ac.uk, Fengyun.Lu@newcastle.ac.uk

Abstract

Interest management aims to overcome limited network resources to provide a distributed virtual environment (DVE) that is scalable in terms of number of users and virtual world complexity (number of objects). Interest management limits interactions between objects in a virtual world by only allowing objects to communicate their actions to other objects that fall within their influence. An important aspect of any interest management scheme is the ability to identify when objects should be interacting and enable such interaction via message passing. Existing approaches to interest management are not suited to objects that may travel at greatly varying speeds and may only interact briefly. In such a scenario, the time taken by existing interest management schemes to resolve which objects influence each other may be too large to enable the desired interaction to occur.

In this paper we present an approach to interest management based on the predicted movement of objects. Our approach determines the frequency of message exchange between objects on the likelihood that such objects will influence each other in the near future. Via this mechanism we aim to ensure a scalable DVE that may satisfy message exchange requirements of briefly interacting objects irrelevant of the speed such objects may traverse a virtual world.

Keywords

Virtual collaborative environments, simulation, games, interest management, information dissemination

1. Introduction

A *distributed virtual environment* (DVE) provides a graphical representation of a virtual world that may be navigated by geographically dispersed users. Mechanisms are provided to enable users to interact with the virtual world, and each other, in real-time. A major challenge for the DVE research community is providing networked virtual environment services with qualities that ensure users share a mutually consistent view of a virtual world while enabling users to interact with each other in real-time. This is exemplified in some types of multiplayer networked games [7] [8], where the success of game play relies on modeling quite intricate user interaction in real-time (e.g., shooting an opponent). The scalability issues addressed by a DVE relate to both the number of users and their geographic location. An increase in the number of users leads to an increase in network traffic, placing greater demands on the underlying network. Furthermore, due to the heterogeneous nature of common public access networks (such as the Internet), message latency may vary on a per user basis, increasing the difficulty of satisfying the consistency and real-time requirements of users on slow connections (e.g., connected via a modem).

Currently, approaches to increasing the scalability of DVE applications have been via the provision of services that use division of the virtual space and predictive modeling techniques. Both these techniques seek to lower the volume of message passing required to ensure consistency. A class of DVE applications, known as Collaborative Virtual Environments (CVE), has pioneered research into the division of virtual spaces (commonly termed *interest management*). These applications aim to support human collaboration and communication (possibly employing such techniques as video conferencing). Ideally, CVE applications limit interactions between objects in the virtual world by only allowing objects to communicate their actions to other objects that fall within their influence. Predictive modeling, based on *dead reckoning* has been used extensively in the production of military type simulations [5] [6], with Distributed Interactive Simulation (DIS) [9] applications an example of these types of DVEs. Dead reckoning refers to the use of predictive modeling techniques to allow the construction of an assumed view of the virtual world from limited data (stored on the local computer) in an effort to satisfy real-time requirements (by removing the network latency element from state updates). DIS is suited to military simulations as movements of objects participating in a virtual world may be predicted with a reasonable amount of accuracy (e.g., flight of an aircraft). Furthermore, such objects may be traveling at high speeds comparative to objects participating in a CVE making CVE techniques difficult to apply to DIS type virtual worlds.

In a modern day DVE there may be scalability benefits drawn from both CVE and DIS research. For example, a virtual world may contain objects of greatly varying types (e.g., foot soldiers and fighter aircraft) that exhibit different movement properties that do not lend themselves to existing approaches to virtual space division. This paper considers the problem of designing a DVE that may provide an interest management scheme suitable for applications that exhibit DIS and CVE type qualities. We base our scheme on the notion of predictability of object movement in such a way that when the possibility of interaction increases between objects (distance between objects decreases) the frequency of message exchange between such objects increases. We continue with a description of interest management in section 2, describe our approach to interest management in section 3 followed by conclusions and future work in section 4.

2. Interest Management

We assume a DVE that represents a geographic space containing objects that may navigate such a space. The DVE is deployed across geographically separated nodes connected by an underlying network. Each node may host a number of objects, their local objects, with nodes responsible for informing each other of the actions (e.g., movement) of local objects via the exchange of messages across the network.

Region and aura are two basic approaches used in virtual space division. We continue with descriptions of these two approaches and identify the problems each approach presents when aiming to satisfy real-time and consistency requirements of a DVE when objects of greatly varying speeds coexist within a virtual world.

2.1. Region

The Scalable Platform for Large Interactive Network Environments (SPLINE) [2] and DIstributed Virtual Environment (DIVE) [1] make use of virtual space division to promote scalability. The virtual world is divided into regions (locales in SPLINE and sub-hierarchies in DIVE). The recipient of a message is limited to only interested participants (i.e., reside within the same, or neighbouring, region as the sender). Nodes hosting objects participating in a virtual world identify which region their objects belong and send messages to a well known reference (possibly a server, group of servers, or a group address [13] representing a region) that supports message dissemination for that particular region.

An important consideration in a region based approach is the size of the regions. A region must be of sufficient size as to ensure objects have the ability to purposely disseminate messages in one region before entering another region. When an object traverses a region boundary the DVE is required to update region membership (identify a region an object belongs to). If there is a possibility that an object can traverse a region in less time than it takes to realize regional membership changes then a node hosting such an object may be unable to disseminate messages effectively. When considering an object that represents a fighter aircraft we can see that the size of a region may be appropriately measured in kilometers. However, this size of region may not be suitable for all types of objects. For example, if region size is decided when considering the top speed of a fighter aircraft then the presence of soldiers traveling on foot may give rise to unnecessary message exchange between nodes that host foot soldier objects that are separated by a distance too great for such objects to influence each other. Furthermore, if region size is more suited to foot soldier objects then a fighter aircraft may traverse region boundaries with such frequency that region membership may not be resolved in a timely fashion. Therefore,

when objects coexist within the same virtual world and can traverse the virtual world at greatly varying speeds, relying on a region based approach alone may not be appropriate.

2.2. Aura

Another approach to the division of the virtual space is illustrated by the work carried out by the Model, Architecture and System for Spatial Interaction in Virtual Environments projects [3] [4] (MASSIVE 1 & 2). MASSIVE uses an object's view of the virtual world to aid in identifying the degree of interaction between objects. Ideally, objects would communicate their actions to only objects that fall within their influence as defined by their aura. An aura may be defined as an area of a virtual world with the owner of the aura potentially exerting influence over all other objects located in this area of the virtual world. The introduction of focus and nimbus allows the degree of awareness between objects to be identified. A focus specifies an object's area of interest and a nimbus identifies an object's level of prominence (as perceived from a viewing object).

In the aura approach to interest management there is no need to regionalize a virtual world. However, there is a requirement for all nodes to exchange positional update information relating to the objects they host in order to identify when aura collision occurs. The frequency of these message exchanges must be sufficient to ensure that aura collision may be determined in a timely fashion to allow nodes to purposely disseminate messages as and when aura collision occurs. There is the possibility that aura collision may occur but objects are unaware of this as such a collision may not last for a sufficient amount of time to enable the DVE to ready the group membership details before objects move away from each other (aura collision no longer exists). Consider again the example of a fighter aircraft object and a foot soldier object. If the fighter aircraft flies over the position of the foot soldier and initiates an attack on the soldier's position then the DVE must detect when aura collision occurs and enable message exchange between the appropriate objects. The aura of the fighter aircraft object may only collide with the aura of the foot soldier object for such a small period of time that it may not be possible to resolve the appropriate object group membership in a timely fashion. A solution to this would be to extend the fighter aircraft's aura to enable such interaction. However, extending the aura may result in the fighter aircraft potentially influencing many more objects than is necessary and may result in scalability problems as the node hosting the fighter aircraft would be required to participate in redundant message exchange with many nodes.

Solutions to interest management scalability that combine aura and region based approaches have been proposed [14]. However, there is still an issue as to what region sizes are appropriate and the ability to determine aura collision in a timely fashion is still a problem.

3. Predictive Interest Management

In this section we describe an interest management scheme based on predicted movement of objects in a virtual world. For simplicity we consider only an object's aura and therefore an object's potential influence. We refer only to an object's aura throughout the remainder of the paper. However, our scheme does not exclude the support of the focus, nimbus and awareness model for further enhancing more detailed interest management schemes.

3.1. Identifying Scope of Interest

The aura of an object describes an area of the virtual world enclosed by a sphere (figure 1). The radius of an aura is specified on a per object basis and is fixed at object creation time, with each object having a single aura and the position vector of an object identifying the centre of its aura. Objects have the ability to influence each other when their auras collide. Objects exert their influence over each other via the exchange of messages.

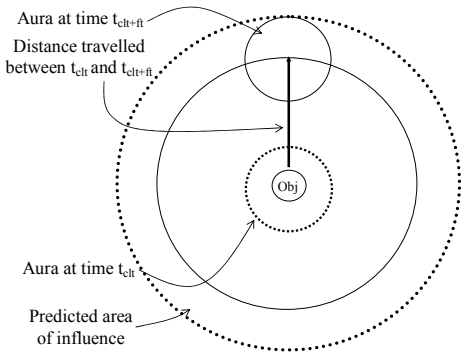

Figure 1: Defining predicted Area of Influence (PAI)

A predicted area of influence (*PAI*) identifies the extent of an object's aura over a period of time given the distance an object may travel in a straight line in any direction (figure 1). The period of time used to identify a *PAI* is bounded by current local node time, say t_{clt}, and some future time (t_{clt+ft}, where *ft* is a positive number and is defined system wide). By this method the distance an object, say obj_1, travels in a straight line identifies the radius of a sphere that encloses all the areas of a virtual space reachable by obj_1 between t_{clt} and t_{clt+ft} with the position vector of obj_1 at time t_{clt} defining the centre of this sphere. Extending this radius by the radius of obj_1's aura defines a sphere that describes the *PAI* for obj_1. When determining a *PAI* we assume an object is traveling at its highest speed (defined on a per object basis) in a straight line at time t_{clt} and continues at this speed and direction until t_{clt+ft}. This presents a *PAI* that is guaranteed to contain all possible auras of an object between t_{clt} and t_{clt+ft} irrelevant of an object's velocity. Assuming the highest speed remains constant for an object throughout its lifetime allows a *PAI* to be calculated and fixed at object creation time.

When the *PAI*s of two objects collide, but not their auras, there is a possibility that such objects may influence each other and subsequently exchange messages at some point in the near future. A period of time (*window of collision*) may be defined within which the auras of such objects may collide. The upper bound of a window of collision is infinity (as both objects may never move). However, assuming two objects are traveling towards each other in a straight line at their respective top speed (figure 2) provides an *optimistic upper bound value* (*OUBV*) as the value of *ft*. Due to the possibility that the auras associated to objects may be different in scope, the value of *OUBV* is specific between a pair of objects that share a collision window.

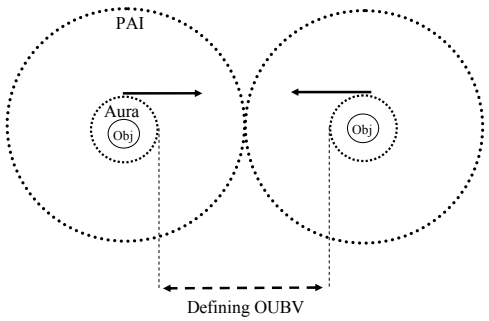

Figure 2 : Defining OUBV

Using collision detection techniques based on the intersection of spheres we may identify if a collision window exists between two objects. This technique is computationally cheap compared to collision detection between polygons as we only need to determine if the distance that separates objects (*sd*) is less than twice the radius of the *PAI*s associated to the objects. Once a collision window has been established between two objects we may determine if their auras collide (the distance that separates two objects is less than the sum of the radii of each object's aura). If a collision window exists between two objects then *sd* may be used to determine an approximate upper bound value (*AUBV*) indicating the time taken for the auras of these objects to collide assuming they are traveling towards each other in a straight line at their respective full speeds. This calculation

for *AUBV* may be derived by first calculating the distance between the edges of two object's auras and deducing the percentage value of *sd* when compared to this value.

We can use values gained for *AUBV* to provide a basis for predicting the appropriate frequency for message exchange between two objects within a collision window. We now continue with a description of a scheme that may be used to initiate the exchange of messages to enable interest management.

3.2. Message Exchange

Each node is responsible for sending a positional update message (*PUM*), identifying the position vector of the objects that it hosts, at regular intervals to all other nodes. Only those objects that have changed position since the last *PUM* was sent will be included in the next *PUM*. Therefore, a *PUM* contains position information relating to a subset of all objects hosted by a node. *PUM* messages are sent frequently between nodes and form the basic mechanism for message exchange between nodes that host objects that influence each other. A *PUM* also carries the unique identifier of the node that sent it.

In order to allow nodes to calculate if *PAI*'s overlap we must propagate additional information. Therefore, a node sends an *admin PUM* (*APUM*) that contains aura radius, *PAI* radius and vector position information for all the objects it hosts (an *APUM* also carries the unique identifier of the node that sent it). Nodes must exchange an *APUM* before they can exchange *PUM*s. *APUM*s are sent much less frequently than *PUM*s and form the basic mechanism for identifying when nodes should exchange *PUM*s and which nodes should receive them.

We assume the existence of a publish/subscribe (messaging) service capable of providing reliable FIFO event channels for message dissemination between nodes. We consider the messaging service reliable because we assume a message published by a node (publisher) is eventually receivable by all interested nodes (subscribers). Messaging services may use the notion of event channels and filters [17] for disseminating messages. An event channel may be registered with the messaging service allowing publishers to place messages on event channels. Subscribers register to one or more event channels and receive messages placed on these event channels. The use of filtering allows publishers and subscribers to further refine the types of messages they are interested in.

We register the following event channels in the messaging service:

- **Admin** – Used to disseminate *APUM*s to all nodes. All nodes subscribe and publish to this event channel.
- **Local** – Created on a per node basis to provide a mechanism for passing *APUM*s and *PUM*s between nodes without the need to publish such messages to all nodes.

A node maintains a list relating to the subscription of nodes to its local event channel. Let us define this list as *L* and a node identifier defined as *i* with a subscript integer used to differentiate between nodes such that $L = \{i_1, i_2, ..., i_n\}$. A local event channel contains a filter such that a number of nodes registered on the local event channel are prevented from receiving *PUM*s but not from receiving *APUM*s. This filter is used to differentiate those subscribed nodes that are known to host objects that are not within a local node's influence but may come within the local node's influence in the near future (i.e., *PAI*s of another node's objects overlap with locally hosted objects but their auras do not overlap). We identify the nodes registered under this filter using the list identifier and a subscript *pai* such that $L_{pai} = \{i_1, i_2, ..., i_n\}$. A logical clock *LC* is defined on a per node basis and is incremented immediately prior to the publishing of a message by a node and used to timestamp published messages. A published message is identified by the type of message and a timestamp written as a subscript (e.g., $APUM_{ts}$). The local *LC* is also used to identify when a node is added to *L* and is described by the timestamp presented as a superscript (e.g., i_1^5 represents a node i_1 added to *L* at time 5). When a node is no longer restricted by the filter L_{pai} we update the timestamp associated with the node to the current value of the local *LC*. $pub_i(m)$ denotes the event of publishing message *m* by a node *i*, $sub_i(m)$ denotes the event of subscribing to a message *m* by a node *i* and $rec_i(m)$ denotes the event of node *i* receiving a message *m*. When the local host is publishing, subscribing or receiving we drop the subscript for clarity. The sender of a message is identified by its node identifier and appended to the message type as superscript (e.g., $APUM_3^{ia}$ identifies a message with timestamp 3 issued by node i_a).We may now clarify our descriptions relating to the validity of event channel subscription and what messages subscribers should receive:

- **If $\mathbf{i_b \in L_{pai}}$ then $\mathbf{i_b \in L}$**, a *node that is registered under the filter L_{pai} is registered under the local event channel L.*

- **If $i_b^x \in L$ then $\nexists\, i_c^y \in L$ such that b = c,** a *node may only be registered once in the local event channel L irrelevant of differences in timestamp identifiers.*
- **If $i_b^x \in L$ and $i_b^x \notin L_{pai}$ then sub(PUM_y) given that $y \geqslant x$**, *a node that is a member of the local event channel L but not inhibited by filter L_{pai} is subscribed to all PUMs that are time stamped greater than the node's timestamp as recorded in L.*
- **If $i_b^x \in L_{pai}$ then sub($APUM_y$) given that $y \geqslant x$**, *a node that is a member of the local event channel L that is associated to filter L_{pai} is subscribed to all APUMs timestamped greater than the node's timestamp as recorded in L_{pai}.*

A node maintains three local timers responsible for regulating the frequency it publishes *APUM*s on the admin channel (*ta*), *APUM*s on the local channel (*tal*) and *PUM*s on the local channel (*tp*) respectively. The time interval used to define the frequency a node publishes *APUM*s on the admin channel may be appropriately measured as a percentage of *ft* (20% of *ft* would be reasonable) and is described as ta^t. The time interval used to define the frequency a node publishes *APUM*s on the local channel is dependent on the *AUBV* times identified by the local node (this is explained in detail later) and is defined as tal^t. The time interval used to define the frequency a node publishes *PUM*s on its local channel is defined on a per node basis (left to developers discretion) and is defined as tp^t. As mentioned previously, the publishing of *PUM*s is more frequent than the publishing of *APUM*s as *PUM*s are used as the message exchange mechanism between nodes hosting objects that influence each other.

On receiving an *APUM*, say from i_b, the receiving node, say i_a, determines how to handle the *APUM* based on the existing message exchange scenario that exist between i_a and i_b. There are three scenarios that may exist: (1) i_b is registered in L but not associated to filter L_{pai}; (2) i_b is not registered in L; (3) i_b is registered in L and is associated to filter L_{pai}. Irrelevant of scenario, i_a receiving an *APUM* from i_b results in i_a calculating if there are any collision windows that exist between those objects described in the *APUM* and objects hosted by i_a. If collision windows exist but auras do not overlap then the lowest *AUBV* value is identified ($AUBV_{low}$). We now describe the events that take place when i_a receives an *APUM* from i_b in each of these scenarios.

(1) If registered in L but not associated to filter L_{pai} -
   a. If no collision windows exist then i_b is unsubscribed from L.
   b. If one or more collision windows exist and auras overlap then the *APUM* is examined to determine if new objects have been introduced into the virtual world by i_b since the last *APUM* sent by i_b.
   c. If collision windows exist, auras don't overlap and $AUBV_{low}$ is greater than ta^t then i_b is unsubscribed from L. If $AUBV_{low}$ is less than ta^t then i_b is associated to L_{pai} and tal^t is reevaluated to take into account $AUBV_{low}$.

(2) If not registered in L -
   a. If no collision windows exist then the *APUM* is ignored.
   b. If collision windows exist and auras do overlap then i_b is subscribed to L (but not associated to L_{pai}). Once subscribed, i_a publishes an *APUM* on L ensuring that i_b gains the latest information regarding object properties at i_a before receiving any *PUM*s from i_a.
   c. If collision windows exists, auras don't overlap and $AUBV_{low}$ is less than ta^t then i_b is subscribed to L, associated to L_{pai} and tal^t is reevaluated to take into account $AUBV_{low}$. If $AUBV_{low}$ is greater than ta^t then the *APUM* is ignored.

(3) If registered in L and also associated to L_{pai} -
   a. If no collision window exists then i_b is unsubscribed from L. tal^t is reevaluated to take into account that an earlier *APUM* that i_b sent may have resulted in its current value. That is tal^t is only valid if derived from member nodes of L_{pai}.
   b. If collision windows exist and auras overlap then i_b is disassociated from the filter L_{pai} and the *APUM* is examined to determine if new objects have been introduced into the virtual world by i_b since the last *APUM* sent by i_b. As in 3.a, this will require the reevaluation of tal^t.
   c. If collision windows exist, auras don't overlap and $AUBV_{low}$ is greater than ta^t then i_b is unsubscribed from L. As in 3.a, this will require the reevaluation of tal^t. If $AUBV_{low}$ is less than ta^t then tal^t is reevaluated to take into account $AUBV_{low}$ and i_b remains associated to L_{pai}.

From our descriptions we can see that the receiving of an *APUM* may result in the initialization of *PUM* exchange between nodes, the suspension of *PUM* exchange between nodes or an increase or decrease in the frequency of *APUM* exchange between nodes (identified by the reevaluation of tal^t). We may now clarify our

descriptions relating to the way a node manages local subscriptions and evaluates tal^t based on information received in an *APUM*:

- **If $rec_{ia}(APUM_x^{ib}) \wedge ((AUBV_{low}$ of $APUM_x^{ib}) \preccurlyeq i_a^{aura})$ then $i_b \in L^{ia} \wedge i_b \notin L_{pai}^{ia}$**, *a node that receives an APUM from i_b that identifies overlapping auras requires i_b to be subscribed to L but not associated to L_{pai}.*
- **If $rec_{ia}(APUM_x^{ib}) \wedge (i_a^{aura} \preccurlyeq (AUBV_{low}$ of $APUM_x^{ib}) \preccurlyeq i_a^{tat})$ then $i_b \in L^{ia} \wedge i_b \in L_{pai}^{ia}$**, *a node that receives an APUM from i_b that identifies no overlapping auras but overlapping PAIs less than ta^t requires i_b to be subscribed to L and associated to L_{pai}.*
- **If $rec_{ia}(APUM_x^{ib}) \wedge (AUBV_{low}$ of $APUM_x^{ib}) \succcurlyeq i_a^{tat})$ then $i_b \notin L^{ia}$**, *a node that receives an APUM from i_b that identifies no overlapping PAIs less than ta^t requires i_b not to be a member of L.*
- **If $rec_{ia}(APUM_x^{ib}) \wedge ((AUBV_{low}$ of $APUM_x^{ib}) \preccurlyeq i_a^{talt})$ such that $i_b^y \in L_{pai}^{ia} \wedge y \preccurlyeq x$ then $i_a^{talt} = AUBV_{low}$,** *a node that receives an APUM from a member of L_{pai} that indicates an $AUBV_{low}$ lower than the current value of tal^t then tal^t must be set to $AUBV_{low}$.*

3.3. Discussion

Sending *APUM*s infrequently on the admin channel ensures that all nodes participating in a virtual world may eventually realize the objects hosted on other nodes. The frequency of sending *APUM*s (ta^t) on the admin channel needs to take into account approximated worst case message latency times between nodes and the need to ensure nodes are prepared for possible *PUM* message exchange in the near future. When distances between objects hosted on different nodes decreases the frequency of message exchanges between such nodes increases. Similarly, when the distance between objects hosted on different nodes increases then the frequency of message exchange decreases. This is a result of updating tal^t based on received *APUM* messages. Via this technique we may ensure that prior to objects influencing each other, our approach to interest management may prepare in advance for appropriate message exchange between nodes.

Message exchange increases between nodes that host objects that move closer to each other in the virtual world. Such objects may move apart before their auras overlap. This may be viewed as unnecessary message exchange between nodes as such objects are not influencing each other. However, we accept this overhead to ensure that objects are prepared for message exchange if their auras eventually overlap.

A difficult design decision is determining the frequency a node sends an *APUM*. We base our decision on $AUBV_{low}$. However, there may be nodes with a high *AUBV* compared to other nodes registered in L_{pai}. Unfortunately, these nodes will be sent *APUM*s at a frequency much higher than required. For example, node i_1, node i_2 and node i_3 host objects that are participating in a virtual world. Both i_2 and i_3 are registered in i_1's L_{pai}. Assume the *AUBV* of i_2 is 1 and the *AUBV* of i_3 is 5. In our scheme i_3 will be sent *APUM*s by i_1 once per second when in fact a frequency of once every 5 seconds would be sufficient. A different design choice would be to send i_2 and i_3 messages independently of each other by i_1 to match the different frequency requirements. This results in increasing the volume of messages sent and the processing required by i_1 managing the sending of messages. This problem is not restricted to DVE development but is common to publish/subscribe systems in general (related to server side filters). The way publish/subscribe services attempt to solve such problems is via the management of message dissemination not at the sender but at the server (scalable publish/subscribe services are commonly deployed as an arrangement of servers). There are several works that are investigating scalable message dissemination for publish/subscribe services [10] [11] [12]. Message dissemination that takes into account varying *AUBV* values may be more appropriately achieved at the server side of a publish/subscribe system rather than the client side (i.e., not in i_1, i_2 and i_3 in our example).

4. Conclusions and Furture Work

Our approach is primarily aimed at virtual worlds that share the same properties as DIS and CVE type applications. Objects within the virtual world may vary greatly in the properties they exhibit (such as speed) and the virtual world may represent a large geographic area. However, there is still a requirement to support intricate interactions between objects in a timely fashion. We have proposed a message exchange policy based on interest management and predictive modeling of object movements. Our approach aims to vary message exchange between nodes based on the likelihood that objects will influence each other in the near future. A node may determine the frequency of sending messages based on the top attainable speed and the proximity of objects. Our

approach requires no regionalization of the virtual world and is suitable for virtual worlds that contain objects of greatly varying speeds. As such, we aim to avoid problems associated with missed interactions involving aura and regionalization by enabling the DVE to ensure message exchange occurs at an appropriate frequency before, during and after the collision of auras. Message exchange is enabled via a publish/subscribe service. We argue that existing research into publish/subscribe services provides appropriate avenues for ensuring message dissemination for DVEs and that DVEs should make use of such services in the future.

Future work will concentrate on identifying our approach to interest management coupled with publish/subscribe services that may provide appropriate message dissemination. This approach is related to the aggregation techniques found in [15] [16] and is already highlighted as an appropriate approach to message dissemination [18] [19]. An implementation of our approach using Java 3D and the CORBA Notification service is in development.

Acknowledgements

This work is funded by the UK EPSRC under grant GR/S04529/01: “Middleware Services for Scalable Networked Virtual Environments”.

References

1. C. Carlssom, O. Hagsand, “DIVE – A platform for multi-user VE”, Computer & Graphics 17(6), p 663-669, 1993
2. J. W. Barrus, R. C. Waters, D, B. Anderson, “Locales: Supporting Large Multiuser Virtual Environments”, IEEE Computer Graphics and Applications 16,6, Nov, 1997, p 50-57
3. C. Greenhalgh, S. Benford, “MASSIVE: a distributed virtual reality system incorporating spatial trading”, Proceedings IEEE 15th International Conference on distributed computing systems (DCS 95), Vancouver, Canader, June 1995.
4. C. Greenhalgh, S. Benford, “A Multicast Network Architecture for Large Scale Collaborative Virtual Environments”, Proceedings of European Conference on Multimedia Applications, Services and Techniques 1997 (ECMAST 97), Milan, Italy, May 1997, p 113 – 128
5. D. Miller, J. A. Thorpe. “SIMNET: The advent of simulator networking”, In Proceedings of the IEEE 83(8), p 1114-1123, August 1995.
6. Standard for Information Technology, “Protocols for Distributed Interactive Simulation (DIS) Applications”, version 2, Institute for simulation and Training report IST-CR-93-15, University of Central Florida, Orlando Florida, May 28, 1993.
7. T. Sweeney, “Unreal Networking Architecture”, http://unreal.epicgames.com/Network.htm, as viewed August 2001.
8. Quake Shareware, http://www.gamers.org/dEngine/quake/info/techinfo.091, as viewed October 2001.
9. D. C. Miller, “The DOD High Level Architecture and the Next Generation of DIS”, Proceedings of 14th Workshop on Standrads for the Interoperability of Distributed Simulations, Orlando, Florida, 1996.
10. A. Carzaniga, D.S. Rosenblum and A.L.Wolf. Design and Evaluation of a Wide-Area Event Notification Service. In ACM Transactions on Computer Systems, 19(3):332-383, Aug 2001
11. A. Rowstrom, A. M. Kermarrec, M. Castro and P. Druschel, “SCRIBE: The Design of a Large-Scale Event Notification Infrastructure”, Proceedings of 3rd International Workshop on Networked Group Communications (NGC2001), UK, 2001
12. S. Duarte, J. Legatheaux Martins, H. J. Domingos and N. Preguicua, “DEEDS – a Distributed and Extensible Event Dissemination Service”, In Proceedings of the 4th European Research Seminar on Advances in Distributed Systems, 2001
13. S. Deering, “Host Extentions for IP Multicasting”, IETF RFC 1112, Aug 1989
14. H. A. Abrams, “Extensible Interest Management for Scalable Persistent Distributed Virtual Environments”, MERL TR-95-16a, 1996
15. S. Benford, C. Greenhalgh, “Introducing Third Party Objects into the Spatial Model of Interaction”, 6th European Conference on Computer Supported Coopertive Work, 1997
16. S. K. Singhal, D. R. Cheriton, “Using Projection Aggregations to Support Scalability in Distributed Simultaiont”, 16th I International Conference on Distributed Computing Systems, 1996
17. OMG, “Notification Service Specification”, OMG TC Document telecom/99/07/01, 2000
18. P. García et al. “MOVE: component groupware foundations for collaborative virtual environments”, Proceedings of the 4th international conference on Collaborative virtual environments, 2002
19. D. Lee, M. Lim, S. Han, “ATLAS: a scalable network framework for distributed virtual environments”, Proceedings of the 4th international conference on Collaborative virtual environments, 2002