Eurographics Symposium on Virtual Environments (2006)
Roger Hubbold and Ming Lin (Editors)

Managing Missed Interactions in Distributed Virtual Environments

S. E. Parkin, P. Andras and G. Morgan

School of Computing Science, University of Newcastle upon Tyne, UK

Abstract
A scalable distributed virtual environment (DVE) may be achieved by ensuring virtual world objects communicate their actions to only those objects that fall within their influence, reducing the need to send and process unnecessary messages. A missed interaction may be defined as a failure to exchange messages to appropriately model object interaction. A number of parameters under the control of a DVE developer may influence the possibility of missed interactions occurring (e.g., object velocities, area of influence). However, due to the complexities associated with object movement and the deployment environment (e.g., non-deterministic object movement, network latency), identifying the value for such parameters to minimise missed interactions while maintaining scalability (minimal message passing) is not clear. We present in this paper a tool which simulates a DVE and provides developers with an indication of the appropriate values for parameters when balancing missed interactions against scalability.

Categories and Subject Descriptors (according to ACM CCS): H.5.1 [Information Interfaces and Presentation]: Multimedia Information Systems - Artificial, Augmented, and Virtual Realities C.2.4 [Distributed Systems]: Distributed Applications

1. Introduction

Providing distributed virtual environments (DVEs) that may scale to support many hundreds and thousands of users while satisfying real-time (responsive virtual worlds) and consistency (participants view mutually consistent events) requirements is a significant research challenge. In addition, as message exchange is the only way to propagate events to geographically dispersed users, care must be taken to prevent exhaustion of bandwidth and available processing resources.

Interest management is an approach to achieving scalability in a DVE (e.g., [ZP91] [GB95]): freeing up bandwidth and processing resources via targeted message passing (not broadcast), ensuring messages are only sent to recipients that may be interested in them. Defined areas within a virtual world are used to restrict message passing: a message and its receivers are associated to an area of a virtual world. Areas used for restricting message passing are commonly termed areas of influence and an object is said to exert influence (send messages) to all other objects that share its area of influence.

When modelling restricted message passing based on object localities in a virtual world there is an opportunity for *missed interactions*. Missed interactions occur when objects that should interact don’t due to a lack of message exchange. Missed interactions are related to the consistency-throughput tradeoff mentioned by Singhal and Zyda [SZ99]:

“*It is impossible to allow dynamic shared state to change frequently and guarantee that all hosts simultaneously access identical versions of the state*”.

Considering the consistency-throughput tradeoff, missed interactions occur because the degree of inconsistency present in a DVE is sufficient to allow an object’s traversal of an area of influence to go undetected by an interest management scheme. Assuming that networking and processing resources may not be easily altered, there are three parameters within a DVE that may be manipulated by a developer in an effort to minimise missed interactions: (i) frequency of message sends; (ii) object velocities; (iii) sizes of areas of influence. At the moment, deriving values for such parameters is left to a developer's ad-hoc estimation.

We argue that due to the way objects move and interact in a virtual world, ad-hoc estimations by developers are not the most appropriate manner in which to attain values for parameters that influence missed interactions and scalability in DVEs. We present a simulation tool that allows a developer to make a more appropriate choice for determining the values of such parameters. We demonstrate our findings via a series of experiments that highlight the usefulness of our simulation tool. Section 2 continues with background and related work, section 3 describes our simulator, section 4 describes our experiments and section 5 presents our conclusions.

2. Background and related work

In this section we describe approaches to interest management and how these approaches are associated to the missed interaction problem. A classification of the missed interaction problem is presented and the common steps a developer may take to minimise missed interactions are described.

As this is the first time a classification of missed interactions has been explicitly described we decided to limit the scope of our work. We restrict our model to the origins of interest management and only consider client/server and peer-to-peer DVE implementations.

2.1 Interest Management

Interest management may be classified into two categories: (i) region based; (ii) aura based. In the region based approach (e.g., [ZP93] the virtual world is commonly divided into well defined uniform sized regions (figure 1.i) that are static in nature (i.e., their boundaries are defined at virtual world creation time). The recipient of a message resides within the same, or neighbouring, region as the sender. In the aura based approach (e.g., [GB95]) each object is associated to an aura (usually a sphere) that defines an area of the virtual world over which an object may exert influence. Ideally, an object may potentially communicate their actions to only objects that fall within their aura.

The aura based approach to interest management provides a more accurate model of object interaction on which to base message exchange than the region based approach. Figure 1 describes a virtual world scene using region and aura based interest management. Using auras we may determine that object *e* (plane) may influence objects *c* and *d*. However, in the region based approach object *e* may only influence object *d* (as *c* is in another region). We could expand the area of influence of object *e* to additional regions and so allow *e* to influence *c*, but this would result in object e appearing to influence object *f* (messages unnecessarily sent from *e* to *f*).

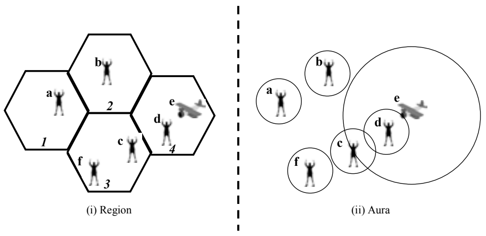

Figure 1: Areas of influence.

An alternative to exerting influence only when an object is within an aura is when auras overlap. In this model the individuals in figure 1 (*c* and *d*) may influence the plane (*e*). This particular approach is commonplace when most objects in the virtual world are similar in nature (i.e., all human type avatars). In addition, gaming environments tend to favour this scenario as influencing someone else without their ability to react detracts from game play (is not fair) [S05].

2.2 Message exchange

To facilitate message exchange a communication subsystem must exist that identifies message recipients in a manner suitable for implementing an interest management scheme. In the region based approach, message exchange may be achieved by associating each region with an identifier that may be used to send/receive messages to/from. For example, each region may be assigned an IP-multicast address. If an object, say Obj_a, traverses a region boundary, say region 1 to region 2, Obj_a will subscribe as a sender/receiver for region 2's IP-multicast address and unsubscribe from region 1's IP-multicast address. Objects need only be aware of which region they are in to enable message passing to occur. There is no need for individual objects to contact each other directly to determine message recipients.

Due to the lack of static regions in the aura approach, identifying message recipients is via the objects themselves (peer-to-peer) or via a server. In a peer-to-peer approach all objects must exchange messages periodically to realise when aura influence is exerted over objects. Such messages are commonly termed heartbeat messages (indicating the location of a sending object) and occur infrequently to avoid exhausting available bandwidth and message processing resources. Once aura influences have been determined, via a heartbeat message, then high frequency message exchange between objects may be enacted. In the peer-to-peer approach this is commonly achieved independently between two objects: if an object, say Obj_a, receives a heartbeat message from Obj_b, and determines that Obj_b is influencing Obj_a then Obj_a will make it known that it is interested in receiving Obj_b's high frequency messages.

In the server approach to aura based interest management all objects send high frequency messages to a server, which assumes responsibility for determining interaction via aura influences. Once a server determines the appropriate influences then high frequency messages may be relayed between objects via the server.

2.3 Missed Interaction

For the purposes of defining different types of missed interactions, we introduce the notion of a *session*. A session is an unbroken period of influence exerted by one object over another. Using a session, we may describe two types of missed interactions: **Complete** – throughout a session no messages were exchanged as expected; **Partial** – throughout a session a smaller number of messages were exchanged than expected.

A session may be unary or binary in nature depending on the expected flow of message exchange. A binary session occurs when both objects should exchange messages (they are influencing each other). A unary session occurs when only a single object should send messages (only one object is influenced). In the region based approach all sessions are binary (as objects that share a region influence each other). In the aura approach unary sessions are possible if an object must be within an aura to be influenced. Missed interactions may manifest themselves as a unary session when a binary session should have occurred. For example, two objects, say Obj_a and Obj_b, should be exchanging messages, but only Obj_a has realised this during the session.

The manner in which a DVE is implemented (server or peer-to-peer) has an influence on missed interactions. A server must inform objects of sessions taking place before missed interaction occur. As all messages are sent between objects and a server at a high frequency, missed interactions relate to the speed (scalability) of the server in determining interaction. In a peer-to-peer based solution missed interactions are related to heartbeat message exchange: the less frequent heartbeat messages are exchanged the more likely missed interactions will occur.

2.4 Avoiding missed interactions

Although much effort has gone into building scalable, responsive DVEs, attempts to minimise missed interactions have received little interest in the literature. In fact, no attempt has been made to describe the types of missed interactions that may occur and how these may relate to different interest management schemes given client/server and peer-to-peer implementation models.

Minimising missed interactions is achieved in an ad-hoc manner at the moment: manipulation of maximum velocity of objects, varying interval between heartbeat message exchanges and altering the area of influence size. Basically, if missed interactions are occurring too frequently as to render the DVE inoperable then a developer may (i) reduce the maximum velocities of objects or (ii) increase the areas of influence so there is a greater chance influence may be resolved before areas of influence are traversed by objects. In addition, when taking a peer-to-peer approach the interval between heartbeat message exchanges may be decreased.

Altering the parameters, described above, that govern the function of a DVE will manifest themselves as a change in Quality of Service (QoS) presented to users: decreasing the velocity of objects may reduce the responsiveness of a virtual world whereas increasing areas of influence or decreasing interval times between heartbeat message exchanges may result in unnecessary message passing. The key question is "what should parameters governing influence and message exchange frequency be set at to provide a scalable, responsive virtual world"?

2.5 Contribution of paper

We have emphasised the difficulties in determining parameters that have an influence on missed interactions, responsiveness and scalability in a DVE. Rather than an ad-hoc approach for determining such parameters, we have developed a simulation tool to provide a more appropriate indication of parameter values. Such a tool allows experimentation with different parameter values while allowing a developer to see how such parameter values have an affect on missed interactions and the volume of messages sent in a simulated DVE. This allows a developer to balance scalability and responsiveness (limited message passing) against missed interactions.

The contribution of the paper may be summarised as follows: (i) provide a clear understanding of the missed interaction problem; (ii) describe a simulation tool to aid in managing missed interactions; (iii) provide performance figures giving an indication as to how best to manage missed interactions.

3. Simulating a DVE

When developing our simulator we decided to concentrate on a single approach to interest management and implementation. We chose aura based interest management deployed in a peer-to-peer implementation as auras provide a more accurate, therefore desirable, model of influence than regions (see 2.1). In addition, this choice provides a more challenging test as missed interactions are also dependent on heartbeat message exchange.

3.1 Objects

We want to simulate an environment that is similar to a virtual world supported by a DVE application. This requires simulating object movement in a three dimensional space. We assume an object's movement is dependent on object type and there is more than one type of object present. For example, a plane will tend to follow a more deterministic flight path at high velocities as opposed to an avatar (digital representation of DVE user) that may exhibit a more non-deterministic movement at lower velocities.

When investigating the literature we found that the majority of studies concentrated on human behaviour during "crowding" in the real world. For example, studying crowds when leaving/entering stadiums in large numbers to improve safety [BBP*05] or studying crowds in urban areas to improve town layouts [HBJ*05]. The mathematical models and associated analysis that have been applied in the real world have not been applied in virtual worlds. Furthermore, there is no proof that what occurs in the real world may be replicated in a virtual world. However, we found that spread throughout the DVE literature there is an understanding that objects, particularly avatars, will crowd and disperse throughout a simulation (objects rarely stand in isolation in a virtual world – there is a drive to interact) [SZ99].

When considering which virtual world properties to simulate we decided against introducing obstacles: we decided to allow objects to roam freely without hindrance. The reason for this decision was to ensure that figures we receive from the simulation would be free from virtual world environmental constraints (such as space limitations due to obstacles). This will give a clear indication of missed interactions occurring due to underlying networking and processing constraints as opposed to reflecting the influence virtual world obstacles may be having on missed interactions. However, exploring how different virtual world layouts, and the associated dynamics of crowding, influence missed interactions is an interesting area for research, but is beyond the scope of this paper.

Introducing types for objects based on velocity and style of movement is a necessity for modelling missed interactions. As can be seen from the descriptions provided in section 2.3, object velocities may vary to such an extent that one object may pass another before influence can be detected and message passing enacted. The style of movement is also a factor in determining missed interactions: influences of objects that have a tendency to stop moving for periods of time may be identified more accurately in real-time than those objects constantly on the move.

When considering the style of movement, we define four types of object movement style for our simulation:

- **Direct** – Objects move, without stopping, along a linear path at a fixed velocity.
- **Indirect** – Objects move along a linear path at a fixed speed but may deviate for periods of time from their path.
- **Stuttering** – Objects move along a linear path at a fixed speed (when moving) but may pause for periods of time.
- **Static** – Objects do not move.

Our choice is based on the assumption that these styles of movement exhibit the basic combinations of directed object movement possible. By combining these styles there is a possibility of deriving quite complex movement scenarios. However, in the first instance concentrating on these styles was considered sufficient for our purposes.

An attempt is made to provide realistic movement of objects within the virtual world via the positioning of targets. A number of targets (*T*) are positioned within the virtual world that objects (*O*) travel towards. Each target has the ability to relocate during the execution of an experiment. Relocation of targets is determined after the elapse of some random time (between T^t_{min} and T^t_{max}) from the time the previous relocation event occurred. Furthermore, objects may change their targets in the same manner (random time between O^t_{min} and O^t_{max}). Given that the number of targets is less than the number of objects and T^t_{min}, T^t_{max}, O^t_{min} and O^t_{max} are set appropriately, objects will cluster and disperse throughout the simulation.

3.2 Messages

To model missed interactions we need to structure the simulations in such a way as the delays associated to processing and networking overheads are present. That is, the delayed receiving of heartbeat messages that lead to missed interactions must be modelled.

Introducing network latency and processing overhead complicates the simulation, as the sending of a message and the receiving of a message will not occur at the same point in global time. Therefore, the receiver, say *Obj*$_1$, of a heartbeat message sent by *Obj*$_2$ will be making a judgement on whether the current known aura of *Obj*$_1$ overlaps with the aura of *Obj*$_2$ with the position of *Obj*$_2$'s aura described at the time *Obj*$_2$ sent the message. This means that the judgement *Obj*$_1$ makes is arrived at using out of date data relating to *Obj*$_2$'s aura.

To model the delay associated to network latency and processing overhead we introduce two variables that describe two periods of time: D_{lat} describes delay associated to network latency; D_{prc} describes processing time required to resolve interest management (processing overhead). For example, given low numbers of objects but limited networking resources D_{lat} would be set high whereas D_{prc} would be set low.

As we do not wish our simulation to be influenced by real processing delays present in an execution environment (e.g., CPU speed, memory availability, preemptive operations in the operating system) we base our measure for time on algorithm iterations. That is, one iteration of our simulation counts as one unit of time.

3.3 Parameters

A number of parameters may be set on a per object basis: velocity, heartbeat interval, high frequency interval, network and processing delay, aura size, and object type. There are a number of parameters that may be set that relate to the virtual world itself: size, number of objects, and number of targets.

Targets may be specifically placed around the virtual world and objects assigned targets. As there may be many hundreds, possibly thousands, of objects in a single simulation such values may be loaded into the simulation via a file. Creation of such a file is left to the developer. However, the simulator does allow a quick setup of a simulation (and creation of appropriate file), setting up aura size, velocity, heartbeat interval and high frequency interval on a per object type basis.

3.4 Outputs

The simulator provides graphical representations (line graphs), describing complete missed interactions, true interactions (when interactions should have occurred), unary interactions, partial interactions and number of messages sent (high frequency and heartbeat). Graphs are drawn in real-time on the screen for a developer to inspect as the experiment progresses. In addition, a file is produced that contains, in text format, the data relating to the outputs listed, allowing a developer to load their data into an analysis tool (e.g., spreadsheet) for further inspection. Figures 2 and 3 are screenshot from the simulator showing true aura overlap and high frequency message exchange graphs generated during a simulation.

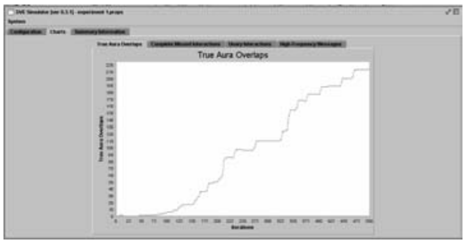

Figure 2: True aura overlaps graph.

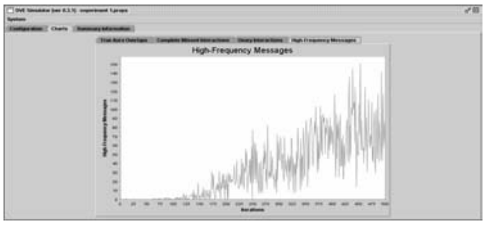

Figure 3: High frequency message exchange

4. Performance

To demonstrate the usefulness of our simulator we carried out an experiment to aid in determining the appropriate parameters for a DVE implementation. Our goal was similar to many developers of DVEs and has been mentioned a number of times throughout the paper, but is worth repeating here for clarity:

> At what time interval should I send heartbeat messages and at what size should I set auras to minimise missed interactions while ensuring I don't overburden the available networking and processing resources (achieve scalability)?

4.1 Experiments

Two series of experiments were conducted to determine the influence varying aura size and heartbeat interval have on missed interactions (complete and partial) and the number of messages sent. In each experiment all parameters remained the same except the parameter in question (i.e., aura size, heartbeat interval):

Number of objects: 50, **Virtual world size:** 5000^3, **Number of iterations:** 500, **Number of targets:** 2, **Distribution of object types:** 25% (equal distribution), V**elocity:** randomly chosen between 10 and 20 for all objects, **High frequency**

message interval: 5 (for all objects), **Network latency:** 2 (for all objects), **Processing latency:** 1 (for all objects), **Number of targets:** 2.

The parameter values used in the experiments model an environment with a low number of objects in a high performance network. Objects are moving relatively quickly in the simulation (direct moving objects capable of moving from one side of the world to another within the execution time of the experiment).

Targets are not relocated, ensuring that most objects will have a reasonable chance of reaching their targets during the execution of an experiment. This guarantees that aura overlaps will be common, and grow in commonality as the simulation progresses. Figures 2 and 3 are actual graphs derived from one of the instances of our experiments, with figure 2 showing that true aura overlaps do increase as expected due to clustering.

In the first experiment series we increased heartbeat intervals gradually from 5 through to 50 inclusive leaving aura sizes for all objects at 80. In the second series of experiments we increased the aura size (for all objects) from 5 through to 300 inclusive leaving heartbeat interval at 25 for all objects. At each experimental step we ran ten experiments and derived the mean value associated to complete missed interactions, partial interactions and number of messages sent.

4.2 Results

The results are shown in the graphs displayed in figures 4 through to 9. We first consider the graphs that relate to the experiments in which the heartbeat interval was altered (figures 4, 5 and 6).

In the graph shown in figure 4 we observe that as the heartbeat interval increases the number of messages sent decreases. This decrease in messages sent is significant in that at a heartbeat interval of 5 there are ~250,000 messages sent compared with ~50,000 sent when the heartbeat interval is 50. This indicates that heartbeat messages dominate Heartbeat Interval ;ages sent. This is to be expected as a heartbeat message is sent to all objects. This calculation approximates an estimate a developer could carry out when determining how many messages are sent:

Determine number of messages sent if all objects send messages to all other objects: Number of objects * Number of objects – 1 = 50 * 49 = 2450

Determine the number of times all objects would send messages to all other objects: Number of iterations / heartbeat rate = 500 / 5 = 100 (send rate = 5), 500 / 50 = 10 (send rate = 50)

From the above calculations determine, approximately, how many heartbeat messages are sent overall: 2450 * 100 = 245, 000, 2450 * 10 = 24, 500

The graph in figure 4 approximates the estimate when heartbeat interval is around 5. However, as the heartbeat interval increases to 50 the difference between the estimate and the actual values deviate significantly, indicating that high frequency message exchange (which is dependent on aura overlap detection) influences the figures in this range. This graph does indicate that the simulator is providing results that may be considered equivalent, in respect of messages sent, to what a developer can best estimate and is not, therefore, depicting something unexpected. The graph, therefore, is most useful when heartbeat messages become less dominant as a percentage of all messages sent.

The complete missed interactions graph (figure 5) rises from 20% to almost 40% for heartbeat intervals changing from 5 to 10, but after 10 the graph flattens out, rising only a few percentage points between heartbeat interval values 10 and 50. This indicates that if a developer wants to see a marked improvement when avoiding complete missed interactions then heartbeat intervals lower than 10 must be considered. In addition, the graph indicates that a developer may as well use a heartbeat interval of 50 rather than 15 (saving networking and processing resources).

In the graph shown in figure 6 the percentage of partial missed interactions rises as the heartbeat interval rises. This graph is interesting in that it does not flatten out in the same manner as the complete missed interactions graph in figure 5. Therefore, although the number of complete missed interactions differs little between heartbeat rates of 10 and 50, there is a rise from 4% to 16% for partial missed interactions in the same period.

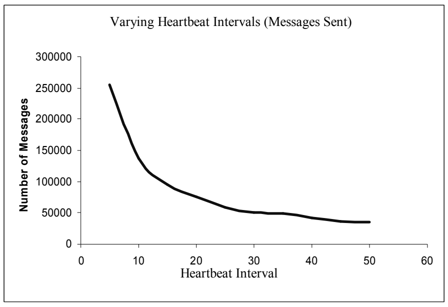

Figure 4: Number of messages sent as function of heartbeat interval.

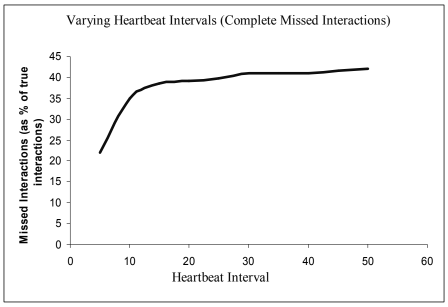

Figure 5: Number of missed interactions as function of heartbeat interval.

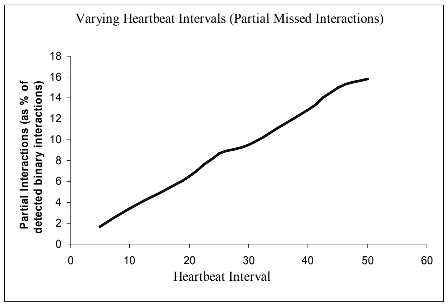

Figure 6: Number of partial missed interactions as function of heartbeat interval.

In the graphs described in figures 7 through 9 we show what affect varying aura size has on the number of messages sent, complete missed interactions and partial missed interactions. Estimating the appropriate aura size is not as straightforward as estimating how many messages will be sent. Developers could assume the worst case scenario when determining aura size: how far can two objects travel directly towards and past each other at full speed during a heartbeat interval – aura must be large enough to cover this size. This would give a large aura size (>200 in our experiment). This type of estimate is not very useful as objects may not be moving directly towards each other in non-direct ways.

The graph in figure 7 indicates that messages sent increases slightly as the experiment progresses. This indicates that heartbeat messages are predominantly responsible (given our earlier estimate relating to graph 4). As the heartbeat interval remains unchanged then the rising curve indicates how high frequency messages become a larger proportion of all messages sent as more and more auras overlap. We can see that varying aura size has little impact on the volume of messages sent.

On inspecting the graph in figure 8 we can determine that the fewest missed interactions occurred with aura size set to 37. In fact, as aura size grew larger than 37 there was a minor increase (1% - 3%) in complete missed interactions. This may be explained by a rise in aura overlaps relating to objects that are moving to different targets, but their auras overlap for a brief period as they pass each other (when world size remains fixed there is more chance of such overlaps). The aura size is much smaller than a worst case estimate. In fact, such an estimate would have resulted in more missed interactions.

Figure 9 shows the partial missed interactions. This shows a similar curve as the complete missed interactions, but the fewest partial missed interactions occurs with an aura size of 32, slightly lower than the optimum aura size for complete missed interactions.

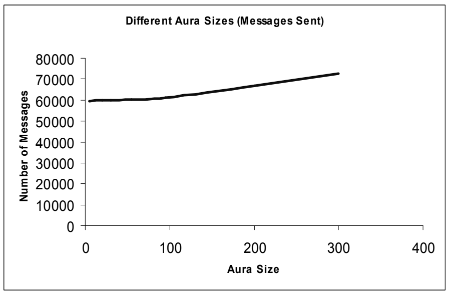

Figure 7: Number of messages sent as function of aura size.

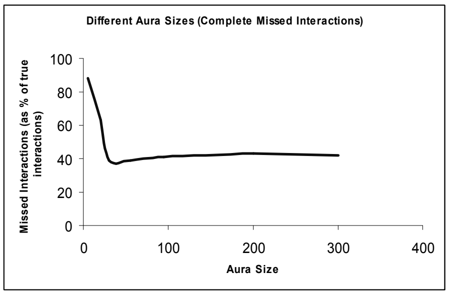

Figure 8: Number of complete missed interactions as function of aura size.

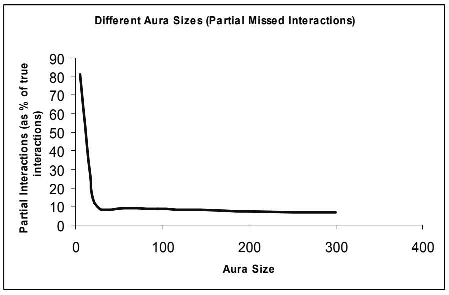

Figure 9: Number of partial missed interactions as function of aura size.

From our experiments it is clear that basic estimations of the type a developer may deduce are not sufficient for determining appropriate values that govern a DVE. There may be more intricate estimations using more formulas, but in our experiences we have not come across them (either in the literature or industry). The simulator provides information not readily available using estimation alone.

5. Conclusions

Using a simulation of a DVE to determine appropriate values for parameters that govern a virtual world is unique. Our simulator provides results, via a series of graphs, which clearly demonstrate how various parameter settings can influence scalability and missed interactions. Using our simulator a developer may tailor the characteristics of the objects that inhabit a simulated virtual world (movements and velocities), ensuring that a variety of DVE application types may be accommodated. For example, virtual worlds that may have objects of greatly varying velocities and diverse movement characteristics (e.g., planes and people in military type simulations) to objects that share similar velocities (e.g., people in collaborative tasks).

We have demonstrated the usefulness of our simulator via experimentation in this paper. We have also employed our simulator in determining the parameters for our own DVE implementation [MLK05].

We consider this an introductory paper to the area of missed interactions in DVEs. We hope the community will explicitly address the missed interaction problem in their own DVE research in the future. For example, at the time of writing this paper IBM appears to be putting significant effort into online game technology [IBM06] (commercial DVEs). Although this work is of high quality and tackles well defined problems, there is no explicit mention in their work of missed interactions and how to overcome them. This is not an anomaly on IBM's behalf, as the community have neglected to define missed interactions with respect to their own work so far.

Our next step is to incorporate our middleware monitoring software [MPM*05] with our simulator to provide appropriate changes to our DVE parameters to accommodate changes in networking and processing resources and virtual world properties during runtime. Simulator may be found at: `http://homepages.cs.ncl.ac.uk/graham.morgan.`

Acknowledgements

This work is funded by UK EPSRC Grant GR/S63199: "Trusted Coordination in Dynamic Virtual Organisations" and UK EPSRC Grant GR/S04529/01: "Middleware Services for Scalable Networked Virtual Environments".

References

[BWA97] BARRUS J. W., WATERS R. C., ANDERSON D, B.: "Locales: Supporting Large Multiuser Virtual Environments", IEEE Computer Graphics and Applications 16,6, Nov, (1997), pp. 50-57

[BBP*05] BROCKLEHURST D., BOUCHLAGHEM D., PITFIELD D., PALMER G., STILL, K.: "Crowd circulation and stadium design: low flow rate systems", Structures & Buildings, (2005), Volume: 158, Issue: 5

[GB95] GREENHALGH C., BENFORD S.: "MASSIVE: a distributed virtual reality system incorporating spatial trading", Proceedings IEEE 15th International Conference on distributed computing systems (DCS 95), Vancouver, Canada, (1995)

[HBJ*05] HELBING D., BUZNA L., JOHANSSON A., WERNER T.: "Self-Organized Pedestrian Crowd Dynamics: Experiments, Simulations, and Design Solutions", Transportation Science, Vol. 39, No. 1, (2005), pp. 1–24

[IBM06] IBM: Online Game Technology, IBM Systems Journal (2006), Vol. 45, No. 1

[MLK05] MORGAN G., LU F., AND STOREY K.: "Interest Management Middleware for Networked Games", In Proc. of the I3D 2005. ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games, Washington, DC, April 3-6, (2005) pp. 57-63

[MPM*05] MORGAN G., PARKIN S., MOLINA-JIMENEZ C., SKENE J.: Monitoring Middleware for Service Level Agreements in Heterogeneous Environments, In the proc. of the fifth IFIP conference on e-Commerce, e-Business, and e-Government (I3E 2005), October 26-28, (2005), IFIP Volume 189 pp. 79-93

[S05] SWEENEY T.: "Unreal Networking Architecture", http://unreal.epicgames.com/Network.htm, viewed December (2005)

[SZ99] SINGHAL S., ZYDA, M.: "Networked Virtual Environments, Design and Implementation", Addison Wesley, (1999)

[ZP93] ZYDA M. J., AND PRATT D. R.: "NPSNET: A 3D visual simulator for virtual world exploration and experience", In Tomorrow's Realities Gallery, Visual Proceedings of SIGGRAPH 91, (1991) p. 30