Interest Management for Collaborative Environments Through Dividing Their Shared State

Michal Masa, Jiri Zara

Czech Technical University in Prague, Karlovo nam. 13
121 35 Praha 2, Czech Republic
michal.masa@ciant.cz, zara@fel.cvut.cz

Abstract. Not all participants in a collaborative virtual environment (CVE) need to be informed of every other participant's activities. The technique used for filtering irrelevant messages is known as interest management, which has to minimize network traffic and to reduce the burden of clients. However, considering the CVE shared state maintenance, interest management is nothing else than a disruption of the perfect case where every CVE participant maintains the identical copy of the state. In this paper we present an interest management technique that organizes the shared state into domains and sub-domains and enables clients to express their interest in particular sub-domains only. This approach specifies an interest management in a general way and it can be used for a wide range of CVE applications. Key ideas are being implemented as part of General Variables (GV) library and verified in our testbed CVE system for social interaction called e-Agora.

Keywords. 3D graphics, collaborative virtual environments, interest management

1 Introduction

Collaborative virtual environments (CVE) in general are aimed at interaction among users connected by network and spread physically. In a typical CVE, users do not need to know about every other user's activities. Filtering irrelevant messages is usually referred to as interest management. Its main goal is to minimize network traffic and to reduce the burden on clients (notion client stands for combination of software and hardware in this text).

The shared state of CVE describes the current state shared among CVE participants [12]. It can be divided into two parts: fixed and dynamic. The fixed part remains unaffected for the whole life of the CVE. It usually describes the geometry of landscapes, buildings, rooms and other virtual objects. In contrary, the dynamic part varies during the system runtime. It can describe positions of users in VE, state of light switches etc. When a client connects to the system, it has to collect both, the fixed and the dynamic part. By composing them together it can produce the actual VE representation. From the point of communication view, the dynamic state is delivered to the client in two forms: initial state and state updates. The former is delivered only once after the client has connected to the system and it contains the actual state at the time of connection. The letter is being delivered to the client until it disconnects from the system and it represents particular changes of the dynamic state.

The optimal situation would be if every CVE participant maintained the identical shared state. From this view, the interest management is a disruption of this ideal. When interest management is employed, participants maintain and synchronize only those portions of the shared state that are of particular interest for them.

In this paper we present an interest management technique that organizes the shared state into domains and sub-domains. The technique is content independent since it filters the data extrinsically [10]. For specifying clients' needs we partially adopt the general aura-nimbus interest management model [5, 12] and extend it by including security issues as well.

In section 2, we divide CVEs into two main directions and characterize them from the view of shared state and interest management. In section 3, we briefly review GV concept and e-Agora CVE. Section 4 contains the description of our approach to interest management and we clarify the notions of domain and sub-domain. In section 5 we demonstrate the usability of our method in e-Agora CVE. Section 6 highlights the consequences of the approach. The experimental implementation is discussed in section 7 and section 8 concludes the work.

2 Previous Work

The research in the field of CVE can be divided into two main directions. The first direction represents large-scale distributed simulations (LSDS) where thousands of entities move and interact in a large and open area [10]. Battlefield simulations are the typical case. The entities should be aware of other entities in their proximity so the visibility is the most important factor for interest management here. The technique used to approximate visibility computations in these simulations is to break up the world spatially into a number of regions of various shapes – grid cells. Every client that wants to receive information subscribes into regions, which intersect with its area of interest. The client can also subscribe directly to an entity of interest to receive high fidelity information about the entity [1, 11].

In LSDS, the environment is usually considered static. The shared state consists of entities' state mostly and no centralized repository is used to store the shared state. The entities are forced to transmit their state periodically (heart-beat) so the late joins can obtain the actual state.

The second research direction is aimed at CVE systems for social interaction and multi-user cooperation [4, 9]. Although interest management techniques used in LSDS apply here as well, the interaction among users is based on a much more subtle basis. In social and cooperative environments, the scene is usually composed of enclosed logical spaces (rooms in a building). The partitioning of the scene into regions for interest management purposes is more intuitive and regions typically correspond to logical spaces. Outdoor spaces can be partitioned in a similar manner as in LSDSs. However, interest management cannot consider only visibility information or entity type information (as usual in LSDS). The scope of information exchange among users is much broader and should not be restricted only to geographic regions. Users can form logical workgroups and interaction within a workgroup can be shared only by workgroup members – either for efficiency or security reasons. For example, while workgroup members who are located in separate regions do not need positional information about other members they cannot see, they should still be able to communicate with them. Alternatively, users who are located in the same region at the same time, but belonging to different workgroups, might be limited in the way they can interact. This technique is called functional filtering [3].

In contrast to LSDSs, social and cooperative systems enable users to modify the virtual environment in a number of ways. The users can introduce new objects in the scene, remove them or change their properties. To be more specific, in our testbed application e-Agora the users could play desk games, install exhibitions or paint graffiti on walls. The state is usually stored in some form of centralized repository (can be virtual in non client-server architectures), which is used to update late joins.

However in many systems, the partitioning of the shared state for interest management purposes is strongly explicit and suited for a particular purpose. For example MASSIVE-3 divides the virtual environment into Locales containing several Aspects [6]. While Locales are used for spatial subdivision, Aspects define functional and organizational scope. In contrast to hard coded approaches we tried to define an interest management abstraction to provide a common base for a wide range of collaborative applications.

3 Overview of General Variables (GV) and e-Agora

The GV concept [8] was designed to formalize storing and distributing updates in a typical net-VE system. If a user performs an interaction in the world the client sets a particular GV to some value (byte stream). This value is then sent to a server, which updates its GVs database and forwards the value to other connected clients. They parse the value and perform the original action locally. If a new client (late join) connects to the system, the server sends it the content of the GVs database so that the client can update its state promptly.

e-Agora [2] is our testbed net-VE system aimed at social interaction and culture content dissemination. Visitors connected via the Internet can see each other by the help of avatars and communicate by chat and gestures. The virtual environment is a model of an existing culture centre in the city of Prague. The system has been built on the top of the GV concept and VRML technology.

4 Partitioning the Dynamic Shared State: Domains and Sub-domains

To allow users to express their interest in only some part of the shared state only, we propose to partition the shared state into domains and sub-domains (Figure 1a). The domains represent categories of areas of interest (logical groups, regions, navigation, chat or game playing). The sub-domains represent concrete areas (particular group, room, chat theme or specific game). Any state variable can belong to any number of domains – its domain set is specified upon its creation and it defines the scope of the variable.

When the state variable update is being sent (a user performs some action in the VE), the actual sub-domain values (sub-domain set) have to be associated with the update, one sub-domain value for every domain in the variable domain set. To construct the sub-domain set, we utilize aura (Figure 1b). The aura represents an individual set of areas of interest, which are impacted by the client. The aura is specified as a set of sub-domains. These sub-domains, which domains match with the variable domains are put in the variable update sub-domain set (Figure 1c).

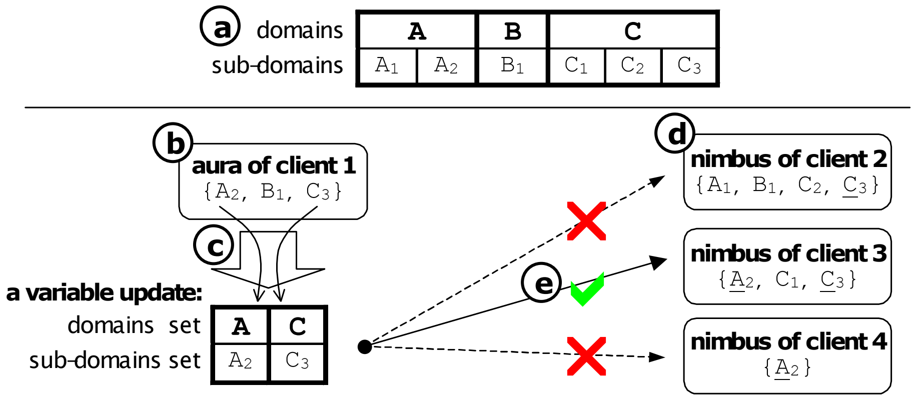

Fig. 1. Process of creating, filtering and receiving the variable update: a) sample domains and sub-domains, b) source client's aura, c) filling the variable sub-domain set according to aura, d) nimbi of potential receiving clients, e) update propagation to a final recipient client 2. The other clients' nimbi do not contain all sub-domains from the update's sub-domain set. Common sub-domains are underlined in clients' nimbi

To express the client's interest in a set of areas, we utilize nimbus (Figure 1d). Nimbus is a counterpart of the aura and is also specified as a set of sub-domains. To determine whether a client is interested in a particular update, we check if every sub-domain in the variable update sub-domain set is contained in the nimbus. If so, the update is propagated to the client (Figure 1e). In other words, we perform restricted intersection of the aura and the nimbus for domains contained in the variable domain set. If the intersection contains a sub-domain for every domain, the update propagation occurs.

5 Application of Domains to e-Agora

While we have specified how to organize the shared state and how to express the interest, the semantics of domains is arbitrary and application specific. To explain our approach, we will illustrate how it can be applied to a future version of e-Agora CVE, which is currently being developed.

Table 1. Domains and sub-domains identified in e-Agora CVE

Domain		Sub-domain	
Chat	C	Different chat themes	C_1, C_2, C_3, …
Navigation	N	General navigation	N_1
Scene Editing	E	General editing	E_1
Desk games	D	Desk games within a world	D_1, D_2, D_3, ...
Spaces	S	Spaces within a world	S_1, S_2, S_3, …
Worlds	W	Worlds making up the VE	W_1, W_2, W_3, …
Logical groups	G	Logical groups within the VE	G_1, G_2, G_3, …

Table 2. Types of state variable domain sets used in e-Agora

Domain set of the variable	Semantics of the variable
{ C, G }	Chat within a group.
{ N, G, W, S }	Navigation in VE. The scope is limited to combination of the group, the world and the space.
{ E, G, W, S }	Independent editing of the scene. The scope is limited to combination of the group, the world and the space.
{ E, W, S }	Joint editing of the scene. The scope is limited to combination of the world and the space. However, it is not limited to the group, so changes are visible to every user.
{ P, G, W }	Playing desk games within the group and the world.

First, we have to define domains by finding possible categories of area of interest. In e-Agora, users can chat, navigate in VE, play desk games or edit objects in the environment (install exhibitions, move chairs and tables, etc.). The VE is made up of several worlds (culture centres) connected together. Every world is composed of spaces (rooms). Users are divided into logical groups that operate independently. According to this classification, we have formed domains and sub-domains as listed in Table 1. Now we can partition the shared state by assigning state variables to one or more domains – construct the variables' domain sets. Table 2 contains five types of domain set used in e-Agora along with their semantics description.

The combination of domains restricts the scope of the variable. Since the chat variables in Table 2 do not contain S or W domain, the communication among users is limited only to participants of the same group and it is not limited in space. Navigation variables describing positions and orientations of the users are a different case. A user has to express interest in a particular combination of the group, world and space to receive updates. The e-Agora also enables two types of editing in the VE. First we have independent editing for every group, where each modification occurs only in the group it has originated from. It can be used for group related projects. Secondly, cross-group editing occurs in every group. An example is installation of an exhibition that should be visible to all users. Playing desk games is also not related to particular space, so users can iconize the desk game, move to another space and continue playing the game.

When a user enters the VE, her client expresses the interest with the help of the nimbus. If she is interested only in chat, her nimbus will contain only particular chat theme or themes and the group to which she belongs: e.g. { C1, G2 }. If she wants to see the other users too, her client extends the nimbus and specifies a particular world and space along with the navigation interest itself: { C1, G1, N1, W2, S3 }. If she wants to see the users in adjacent spaces too, her client extends the nimbus again by adding these spaces: {C1, G1, N1, W2, S3, S2, S1}. And finally, if she wants to join or watch some desk game, her client extends the nimbus by the particular desk game: { C1, G1, N1, W2, S3, S2, S1, D2 }. The aura is being composed in the same manner. However, the aura usually contains only one sub-domain for each domain; for example the user is usually presented in one space only.

6 Consequences of the Proposed Approach

6.1 Generalized Late Joins

As we have illustrated, users may be interested only in particular updates of the shared state. If two users play chess in a pub, most of other users in the pub do not need to be informed about their turns. However, the same findings apply to late joins - only a fraction of the global shared state is of particular importance for them. In our concept, we generalize the notion of late join to include already connected clients that has changed their nimbus. Once the client changes its nimbus (or sets it up for the first time), it becomes a late join for specific part of the shared state. Like if another user in the pub from example above wants to watch the chess game, the client changes the nimbus accordingly and receives the current state of the game. Subsequent game updates can follow until the client reverts the nimbus. The client can provide a timestamp of the last update it has received to obtain only recent changes of the state.

6.2 Security Management

Interest management techniques have to minimize network traffic and reduce burden of clients. However, messages can be filtered for security reasons too. This is applicable in cases when users should be restricted to perform some activities or to observe activities of other users. These security requirements can be seamlessly integrated with our interest management concept. Every nimbus and aura change performed by a client can be a subject to a security checking. If the client tries to add a sub-domain to its nimbus, the security checking compares the client's access rights with the rights required to receive updates from the sub-domain. The nimbus modification has no effect if the comparison fails. Similarly, if the clients try to add a sub-domain to its aura, client's access rights are compared with the rights required to send updates to the sub-domain. Again, if the comparison fails, the aura modification has no effect.

7 Implementation

To verify the proposed approach, we extended our existing GV library by aura/nimbus manipulation methods. The GV library is implemented as a JavaBean [7] to ensure platform independence and reusability.

To eliminate any dependencies on particular networking schemes (client/server, peer-to-peer, multicast), we integrated an abstract class Channel to the concept. Implementations of this abstraction are responsible for dissemination of the updates to all nodes, which nimbi intersect with the source node's aura. The appropriate Channel for the update propagation is selected based on flags associated with the update. Several channels were implemented, each providing different QoS: TCPChannel for reliable ordered connection, UDPChannel for unreliable unordered transmission, MChannel for unreliable multicast and LRMPChannel for reliable multicast. Channels are registered by addChannel method of the class Concept, which inserts them into an ordered list by supplied priority level. Whenever an update is being sent, sorted list of registered channels is traversed and the first channel that accepts the particular flags settings of the update is delegated to broadcast the update. Channels are also responsible for receiving the update and notifying the Concept class that in turn invokes variableUpdated callback method.

7.1 Future Work

We have already implemented a straightforward client-server solution for aura/nimbus manipulation and for storing and sending updates to late joins. Currently we are working on pure peer-to-peer architecture, which involves appropriate mapping of domains and sub-domains to a given set of available multicast addresses.

8 Conclusion

In this paper, we have identified the problem of interest management in the context of the maintaining complex CVE shared state. We argue that interest management technique actually limit or restrict the CVE participants to maintain and receive only specific parts of the global CVE shared state. We have proposed a general interest management method, which is based on partitioning the shared state into domains and sub-domains. The aura-nimbus model has been adopted for expressing clients' impacts and interests in the VE. We demonstrated an application of our approach on e-Agora, a CVE system aimed at social interaction.

9 Acknowledgments

This work has been partly supported by the Ministry of Education, Youth and Sports of the Czech Republic under research program No. Y04/98: 212300014 (Research in the area of information technologies and communications).

References

1. Abrams, H., Watsen, K., and Zyda, M. Three tiered interest management for large-scale virtual environments. In Proceedings of Virtual Reality Systems and Technology, 1998.
2. Adamec, J., Cizek J., Masa M., Sidoni P., Smetana P., and Zara J. Virtual House of European Culture: e-AGORA. In Proceedings of the 1st International Conference on Virtual Storytelling, 2001.
3. Capps, M., and Teller, S. Communications Visibility in Shared Virtual Worlds. In Proceedings of the Sixth IEEE Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises, 1997.
4. Dive Web site: http://www.sics.se/dive/
5. Greenhalgh, C., and Benford, S. Boundaries, awareness, and interaction in collaborative virtual environments. In Proceedings of the Sixth IEEE Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises, 1997.
6. Greenhalgh, C., and Snowdon, D. Inside MASSIVE-3: Flexible Support for Data Consistency and World Structuring. In Proceedings of the Third International Conference on Collaborative Virtual Environments, 2000.
7. JavaBeans specification: http://java.sun.com/products/javabeans/docs/spec.html
8. Masa, M., and Zara, J. The General Variables Concept: A Simple Step from Single- to Multi-user Environment. In Proceedings of 2001 IEEE International Conference on Information Visualisation, 2001.
9. MASSIVE-3 Web site: http://www.crg.cs.nott.ac.uk/research/systems/MASSIVE-3/
10. Morse, K. L. Interest management in large-scale distributed simulation. 1996.
11. Morse, K.L., Bic, L., and Tsai, K. Multicast Grouping for Dynamic Data Distribution management. In Proceedings of the 31st Society for Computer Simulation Conference, 1999.
12. Singhal, S., and Zyda, M. Networked Virtual Environments. Design and Implementation. ACM Press, 1999.