Data management in Distributed Shared Virtual Worlds*

Tanguy Nedelec

Tanguy.Nedelec@imag.fr
LSR/IMAG,
University of Grenoble,
BP 72, 38402 Saint-Martin d'Hères, France.

Abstract

Research efforts in Virtual Worlds domain have been mainly devoted to user interfaces, communication supports and consistency issues. To the best of our knowledge, no works clearly focused on the global problem of managing virtual world's data. The promise of future large persistent virtual worlds, evolving during a long period of time and with large number of participants makes the question of data management even more crucial. This paper introduces the global problem of managing data in virtual worlds and reports our research work in using data management services for virtual worlds platforms.

1 Introduction

Distributed shared virtual worlds are applications trying to provide to several users, geographically dispersed, an immersion in a world with the illusion that it is locally managed [18]. The user feels things as though he/she interacts in real-time with the entities populating the virtual world. Usually, a particular entity, called avatar represents the user itself in the virtual world. Trough this avatar, the user is able to perceive the world (subjective views may be used) to interact with and also to manipulate other entities of the world.

Research efforts in the Virtual Worlds application domain have been mainly devoted to user interfaces (input devices, graphic rendering), communication supports (multicast groups, filtering ...) and consistency issues. All these works have lead to mature technologies and their application in several contexts: video conferences,

*Part of the work has been done in the framework of the PING project N° IST-1999-11488

distributed military simulations [10], virtual shops, virtual museums, games [5, 19], etc.

The development of Virtual World (VW) applications is currently facilitated by the use of particular platforms [1, 21, 22] proposing appropriate network, system or tools support (for example specific network protocols in Open Community platform). Therefore, a developer would concentrate on coding virtual world logics.

However, data management functions have not been well addressed in current Virtual World platforms [12]. Solutions proposed by these platforms for persistency management are not really satisfactory (see section 2.3). The promise of future large persistent virtual worlds, evolving during a long period of time and with a large number of participants makes the question of data management even more crucial. To the best of our knowledge, no works have clearly focused on the global problem of managing virtual world's data. Therefore, VW developers either choose to use a standard Data Base Management System (DBMS) or to develop an ad hoc solution based on flat files with some kind of serialisation.

DBMS are powerful systems offering a plethora of functionalities. Some of them are not required for VW applications and even if unused they have impact on the performance of the VW. For example, most DBMS offer a transactionnal support that is clearly not needed in VW applications. Even if not explicitly used in VW applications, its execution model is a transactionnal one and internal data structures and tools for managing transactions are used.

On the other hand an ad hoc solution based on flat files and serialisation would certainly[1] lack of many useful features for data management. This solution only provides a support for permanence. No main memory management, no support for optimised access path with indexes and no fault tolerance are available.

The problem we are trying to tackle is how to provide a full data management support to targeted VW applications without using a DBMS. This support has to be adaptable in order to take advantage of the VW application characteristics and to provide ad hoc solutions. It is also supposed to scale the targeted highly interactive persistent massively multi-users VW.

To reach this goal, we propose to experiment a different alternative to data management in VW. Our proposal adopts the service approach developed in the NODS (Network Open Database Services) project [2]. This project aims at defining an open, adaptable architecture that can be extended and customised on a per-application basis such as in [4]. DBMS functionalities and related tasks are unbundled into services (e.g. a query service, a persistence service) that can be used by applications when needed. A particular attention is paid on the adaptability of services.

My PhD work concentrates on the integration of a persistence service and a

[1] only certainly because nothing prevent from a brave developer that will develop all these feature alone for its system.

replication service in VW platforms. Experiments are currently in progress in the context of the PING IST project [11] that aims at developing an open and scalable platform supporting large-scale real-time interactive networked applications over the Internet.

The rest of this paper is organised as follows: Section 2 introduces data management in VW platforms and gives an overview of existing proposals. Section 3 introduces our approach for using data management services in VW platforms. Finally, Section 4 gives the current status of our work and future work.

2 Data management in Virtual Worlds

This section first characterises data management in virtual worlds and then details persistence and replication aspects.

2.1 Virtual Worlds

A distributed shared virtual world is a set of entities perceived and manipulated by several participants geographically dispersed. Before further discussion on how to manage data, it is necessary to determine the way data evolve and are accessed.

Behaviour of virtual world entities

The way an entity evolves is related to its *behaviour*. What is a *behaviour* in virtual world. If we consider an entity as a collection of attributes, then its *behaviour* is the ever-changing state of all those attributes over time. For example, an entity may have an attribute called "location"; as the entity moves around, its location changes. Behaviour of an entity could also be described as anything it does in response to external/internal stimulations or changes in its environment.

For data management purpose, it is useful to divide entities into those whose behaviour is deterministic and those whose behaviour is non-deterministic. Entities with non-deterministic behaviour are by nature unpredictable. Certainly human beings but also any entity that has a glimmering of simulated intelligence are unpredictable. An entity's behaviour qualified as deterministic means that the state of the entity is a function of time only. The deterministic intrinsic nature of an entity may be compromised by interactions with other entities. Any entity that responds to direct or indirect interactions with a non-deterministic entity is itself non-deterministic (since its state is unpredictable, being the results of unpredictable actions).

Entities behaviour can be further classified. Entities with deterministic behaviour can be "static" or "dynamic". For further details about the taxonomy of entity behaviour see [20]. The state of an entity considered as static never changes;

the behaviour is of course deterministic. Unlike static entities, dynamic entities (also referred as World physics) change over time. However, changes are easily predictable, and are a function only of time and possibly a set of pre-defined behaviour parameter (think of hands of a clock, an animated waterfall, etc).

Further comments on the taxonomy is that with regard to physics laws and the way human beings understand the world, there is only one category represented in the "real world": entities with undeterministic behaviour. Nevertheless, simulating a virtual world designed with exactly the same laws as the "real world"[2] is not possible even considering all processing ressources distributed all over the world. We need deterministic entities to simulates a world.

Accessing data

The second characteristic that have to be pointed out is the way data is accessed in virtual world applications. The way data is accessed is strongly related to the relationship between entities. In a virtual world, an entity is not aware of all other entities of the virtual world. An entity is aware of only a subset of the whole virtual world population.

Perception and interaction capabilities are associated with entities. For example, your vision field defines what you can see. Perception capabilities can be determined for each entity. Similarly, you are not able to touch everything you see. An interaction area can be then also defined for each entity. Perception and interaction capabilities of entities can be altered by other entities. For example, an opaque wall affects your vision field.

Both, behaviours characteristics of entities and their perception and interaction capabilities, provide lots of useful information that may be exploited for data management support. Next section introduces data management characteristics in virtual worlds and shows how some characteristics of virtual world entities can be taken into account.

2.2 Data management requirements

This section focuses on replication and persistence, two main characteristics of data management. It also introduces an important feature of virtual worlds: interest management.

Replication property

Distributed nature of the virtual worlds we are interested in, induces the question of where data is located. Sharing a vision of a set of entities requires several processes

[2] Such a virtual world is conceivable as far as human knowledge approach some law potentially ruling the universe.

to own at least a copy of the graphical representation of these entities with additional data like their position and rendering information. Replication is inherent to the shared and distributed aspects of VW. Apart from perception needs, replication techniques may be used for: fault tolerance, availability, and performance purposes.

Having several copies of the same data raises coherency issues. Modifications upon copies and updates propagation have to comply some rules in order to provide some level of coherency between copies.

Among the characteristics of VW applications, the types of behaviours of entities can be exploited to use different update propagation policies. Active replication, with propagation of the methods to be executed is only possible if the behaviour of the entity is deterministic. Passive replication, with propagation of the states updates can be applied with all types of entities.

Most of existing virtual worlds systems rely on a master-slave replication model [17] but do not precisely exhibit the respective roles of the master and the slaves. Analysing the literature, it is also really hard to know what kind of information circulates between replicas. For these reasons, we will concentrate on a vague notion: integrity of entities and who is responsible for that. In virtual worlds, for each entity there is a process responsible for its integrity. Such process is always able to provide an up-to-date replica of an entity. The geographic distribution of participants in a virtual world does not imply the distribution of the responsibility of entities. It is then possible to have a centralised management of entities or a distributed one based on some criteria (for example partitioning of the world). A second characteristic is whether the distribution of responsibilities may change during the execution of the VW application. A distribution of entities management is qualified as dynamic or static regarding whether it may change or not during the VW execution.

Dead-reckoning techniques are original techniques used in the context of virtual worlds application to relax coherency between copies. These techniques are based on the prediction of the evolution of an entity on a process owning slave replicas while a process owning a master replica performs the "real" evolution of the entity. The master replica also executes the prediction algorithm in order to detect when the prediction is too "far" from the "real" evolution. When the distance is greater than a given amount, the master replica propagates updates to its slaves.

Persistence property

Lifetime of data entities in today programming environments can span over the execution time of certain blocks of code. For many reasons, stemming mostly from implementation and historical roots, lifetimes are classified into two categories. Entities outliving a single application execution are referred to as persistent entities. Otherwise, they are referred to as transient ones.

In the context of shared VW, the persistence property can be attached to either some world entities or to the virtual world itself. The persistence of a VW is the

ability for this world to survive and to evolve even after every player has left. Then, the persistence of a virtual world is the consequence of the persistence of a large number (if not all) entities of the VW. Persistence is easily associated, if not mistaken, with permanence. In the context of VW, perhaps more than in other contexts, creation of copies on permanence support is a subset of tasks related to the support of the persistence property.

To ensure persistence some information have to be copied from one memory space to another. The source copy is first located in a memory space associated to the process who created the entity. The destination where a copy will be created may be either: a space in main memory associated to another process (may be on another site) or a space on permanent support (independent from any process life-cycle). Logging may also be used to store information related to updates of the source copy and therefore contributes to provide persistence.

Persistence as the ability to make the world continue to evolve may be provided by some specific processes responsible of the evolution of some entities of the world. In this case, the type of behaviour of entities is really important as it may enable to support evolution in different manners. An entity with a deterministic behaviour can be stored to permanent support and its new state would be calculated only when the entity has to be reactivated.

Interest management

Interest management can be used to reduce the amount of data manipulated by a process. Using perception and interaction capabilities of an entity, interest management allows to determine the set of entities that are relevant to this entity. Interest management schemes [15] enable to calculate the appropriate subset of entities using interest expressions. An interest expression may also refer to several attributes of the virtual world and/or entities. For example, geographic data allows to partition the world in fixed areas. Interest expression may refer to any attribute of an entity including its type (e.g.: “give me all the dogs within a 100 meters radius”).

2.3 Existing proposals

This section discusses some representatives approaches to data management in VW platforms. We pay particular attention to the way some level of persistency is ensured and how the responsibility of entities is distributed. In most of the systems overviewed, a clear distinction can be established between, on one side, static and deterministic entities and, on the other side, dynamic and undeterministic entities.

Local storage of source data

The defintion of data manipulated is mixed with the code of processes to manipulate data. In such case, the executable code contains the descrition of the world, called source data, and ensure a primitive level of persistence of the world. In NPSNET IV [14] and the Community Place infrastructure [13], this approach ensures persistence of static entities. Other kind of entities, for example avatars, are not persistent but are replicated and managed in a fixed distributed manner.

Centralised Management of data

In many VW systems, especially commercial games, management of entities is centralised and sometimes dynamic for load balancing purpose. This leads to usual advantages and drawbacks of centralised approaches: easier management of consistency issues but potential bottlenecks. In the V-Worlds platform [21], Everquest [5] and Asheron's call [19] games, two levels of persistency are provided. Local storage of world source data is used for static and deterministic entities. Storage on centralised servers is used for dynamic and undeterministic entities like user avatars and monsters. A weak level of evolution of the world description is provided by updating from time to time local storages located at each player home. Blaxxun Interactive platform 5.1 [1] also falls in this category but with use of a DBMS (Oracle or Microsoft SQL Server).

Persistent servers

Open community [22] and NetZ [16] systems use dynamic distributed management of entities. They are able to migrate mastership of entities between sites in order to support persistence. The destination of the migration is a persistent server which keeps entities alive as long as itself is alive. This technique allows to provide persistence for dynamic and non deterministic entities. Persistent servers do not support the evolution of entities.

Distributed logging approach

Massive 3 [7] and Dive 3 [6] virtual worlds systems use some neutral processes to keep track of activity in the virtual world region they are responsible for. Both systems implement a dynamic distributed management of entities. Logs can then be used to restart after a crash (Dive 3, but recovery is then quite long) or to redo some actions (Massive 3). Once again this "advanced" support for persistence is only used for dynamic and deterministic entities.

Hybrid solutions

NetEffect [3] and HP Living Space [8] systems implement hybrid solutions for distribution of entities responsibility. In these systems, a centralised main server is responsible for the distribution of entities over a set of peer servers each of them responsible of a region of the world. NetEffect uses a fixed distribution with no persistence support. In HP Living Space, when a peer server with some master copies disconnects, the responsibility of these copies (the mastership) are transfered to other peer servers. This transfer is decided by the peer servers and provides persistence to dynamic and non deterministic entities.

3 Data management services in a platform for Virtual Worlds

We claim that existing systems lack several properties to support future long lived persistent VW. As stated in Section 2.1, VW are defined by their content i.e., the entities populating the world and the relationships between them. Existing systems exploit these informations, in a hardwired way. Current systems offer several predefined policies but do not allow the addition of new ones. Considering the type of entities, the application programmer may want to have different update policies or even plug a new update policy. Another example of lack of flexibility in existing VW supports is the way persistence can be adapted to the world: adequate cache management, types of indexes, types of storage (flat files, relational databases, ...).

Following the separation of concerns methodology [9], we propose to define adaptable and flexible data management services: caching service, replication service, logging service and persistence service. These services can be instantiated to define managers as shown in figure 1.These managers provide functions to object / event management. Services are designed so as to allow to implement different policies to do a specific task. Gray boxes / interfaces are the services we are working on. White interfaces are required but are not developed by us.

We currently left coherency and consistency out of the scope of our work. Other services are designed to be independent from these issues. For example, the replication service deals with how synchronisation is ensured between copies of a single object. In the current prototype (for the PING platform), the replication service can be used with different consistency policies dealing with when synchronisation will be effective. These policies are implemented in the consistency control service, responsible for ordering of events according to some causal and real-time constraints. The networking service provide both reliable and unreliable unicast and multicast communication channels between nodes.

We provide an open persistence service, allowing to investigate how to “tune” the persistence machinery to the particular requirements of VWs applications. Per-

sistence service is composed of several underlying components. A storage manager provides means to manage data on stable storage. Index managers provide specific index structures. A log manager provides atomic write operations, as well as recovery capabilities. A cache manager provides in-memory management of persistent data.

Our replication service allows to create and destroy replicas, and furthermore, manages the replica synchronisation according to replica accesses. The replication service is based on the logical object abstraction that groups together all the replicas of the same application object. The application perceives only the logical object, allowing so to manage the consistency of the replicas in a transparent way.

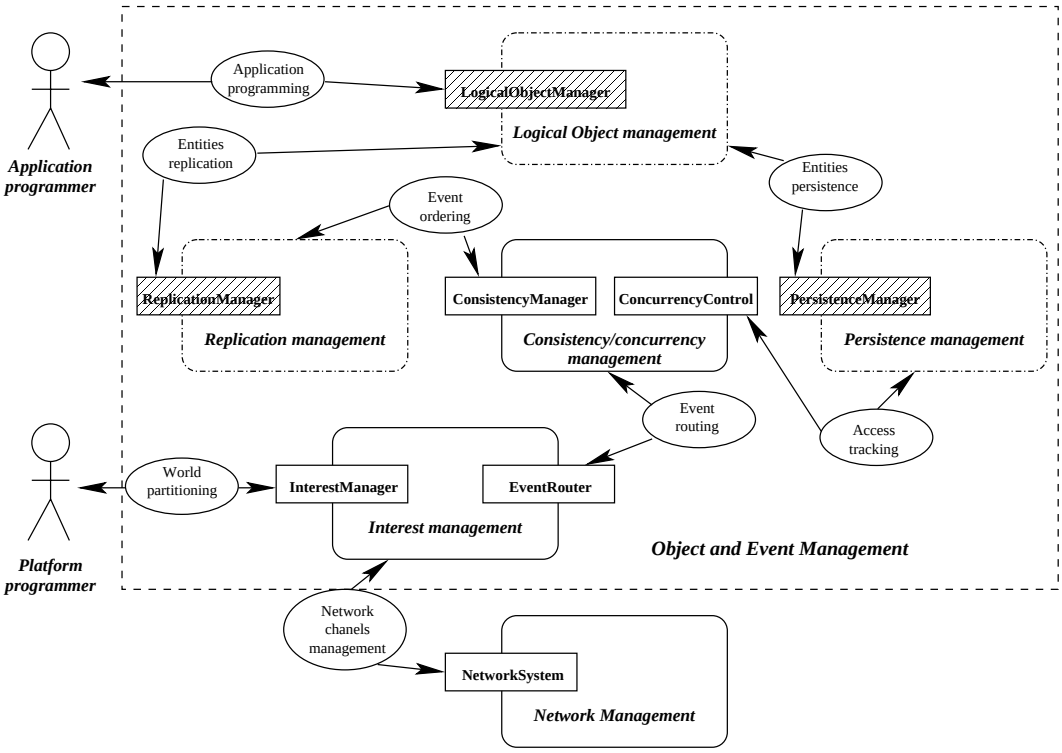

Figure 1: Object manager components

We point out that the object management architecture given here is not limited to a functional abstract view. Each service is implemented to preserve some level of isolation from other components of the platform. Interactions between components of the platform is made through well defined generic interfaces. For example, the caching service exhibits four methods. In addition to this interface, two other interfaces provide methods to tune and configure a cache manager.

One of our main objectives in the design of the object manager is to exploit when possible, many applicative hints. The basic idea is to try to take advantage of the application semantics to improve object management. In virtual world applications,

applicative hints are mainly provided by interest management. In the context of an object manager, interest management is defined as a way to distinguish data relevant for a given participant in the virtual worlds from the whole amount of data of the virtual world. Our object manager relies on high level interest management scheme. Interest management is defined as the ability to determine which entities are relevant for a given entity. The object manager does not make any assumption on the interest management policy used and it does not rely on any particular one, it only interacts via the high level paradigm of "which entities is a given entity interest in". This paradigm is used to:

- “drive” cache management, i.e. add, fix and unfix the relevant objects in the cache,
- “drive” propagation of persistence property. This allow to make persist all entities relevant to entities that have been explicitly made persistent,
- “drive” replication of objects and then to determine where it is necessary to create which replica.

4 Current status and future work

An integrated implementation of the managers we presented is already running. The replication service instance uses a master/slave protocol. The persistence service currently manages local persistent instances. Entities can be made persistent if the process owning their mastership is bound to a local persistence service instance. Two instances of the caching service provide respectively main memory caching for entities and memory pages. Last manager is an instance of the logging service. Checkpointing / recover operations are not yet implemented. The whole platform is coded in Java enabling to design virtual worlds applications in Java. The application programmer describes the VW entities in files which are used to generate adequate Java code in order to make these entities sharable among the VW. The application programmer still need to code entity behaviour and the application logics. He may also tune and configure persistence and replication service if default tuning and configuration do not match to its needs.

Future work includes the addition of distribution within the support. We intend to explore the construction of such a support using the existing replication service and local persistence service. This may become an intermediate service, with its own interface, including distribution of the service among servers and load balancing algorithms. Particular attention will be paid to both pure and hybrid peer to peer approaches which may be the more appropriate to scale a large number of participant sites. The integration of the persistence service with interest management needs to be further explored. Also the ways to access data have to be refined to

provide the application programmer with methods closed to the most used access patterns.

Acknowledgement

Special mention to all people involved in the definition of the platform: Luciano Garcia-Banuelos, Stephane Drapeau and Phuong-Quynh DUONG for their respective contribution via their Ph.D project currently in progress: the persistence service, the replication service and logging service (towards a fault tolerance service); Claudia Roncancio and Christine Collet for their hard work in supporting my Ph.D project.

References

[1] Blaxxun interactive, Elsenheimerst. 61-63, 80687 Munich, Germany. *Virtual Worlds Platform 5: product specification. version 5.1*, 2001.

[2] Christine Collet. The nods project: Networked open database services. In *Proceedings of the International Symposium on Objects and Databases*, pages 153–169, Sophia Antipolis, France, 2000. Springer Verlag.

[3] T. K. Das, G. Singhal, A. Mitchell, P. S. Kumar, and K. McGhee. Neteffect: A network architecture for large scale multi-user virtual worlds. In ACM Press, editor, *Proceedings of the ACM Symposium on Virtual Reality Software and Technology (VRST97)*, pages 157–164, New York, September 1997.

[4] K. R. Ditrich and A. Geppert. *Component Database Systems*. Morgan Kaufmann Publishers, 2001.

[5] Sony Online Entertainment. Everquest, 1999. http://www.everquest.com.

[6] E. Frecon and M. Stenius. Dive: A scalable network architecture for distributed virtual environments. *Distributed Systems Engineering Journal (DSEJ)*, 5:91–100, 1998.

[7] C. Greenhalgh. Massive-3/hivek: Introduction. Technical report, Communications Research Group, June 1999.

[8] R. Hawkes and M. Wray. Livingspace: A living worlds implementation using an event-based architecture. Technical report, HP Laboratories Bristol, Bristol, October 1998.

[9] Walter Hürsch and Cristina Videira Lopes. Separation of concerns. Technical Report NU-CCS-95-03, College of Computer Science, Northeastern University, Boston, Massachusetts, February 1995.

[10] IEEE Computer Society. *HLA Interface Specification version 1.3*, February 1998. IEEE 1516.1.

[11] IMAG-LSR and France Telecom R&D. Object and event management: First specification. Technical report, PING Consortium, 2001.

[12] IMAG-LSR, France Telecom R&D, ENST, and Universisty of Reading. Persistency, replication and real time consistency. Technical report, PING Consortium, 2000.

[13] R. Lea, Y. Honda, K. Matsuda, and S. Matsuda. Community place: Architecture and performance. In *Proceedings of the VRML'97 conference*, Monterey, February 1997.

[14] M. R. Macedonia, M. J. Zyda, D. R. Pratt, P. T. Barham, and S. Zeswitz. Npsnet: A network software architecture for large scale virtual environments. *Presence: Teleoperators and Virtual Environments*, 3(4):265–287, Fall 1994.

[15] Katherine L. Morse. Interest management in large scale distributed simulations. Technical report, Departement of Information and Computer Science, University of California, Irvine, 1996.

[16] Quazal. *NetZ 2.0: Technical Overview*, November 2001.

[17] C. Roncancio, T. Nedelec, E. Perez-Cortes, and A. Gerodolle. Issues in the design of large-scale shared networked worlds. In *Proceedings of sixth International workshop on groupware (CRIWG 2000)*, pages 158–161, Madeira, Portugal, October 2000. IEEE Computer Society Press.

[18] S. K. Singhal and M. Zyda. *Networked Virtual Environments Design and Implementation*. ACM Press, addison-wesley edition, July 1999.

[19] Turbine Entertainment Software. The turbine engine 2.0, 1999. http://www.turbine-games.com/tech.htm.

[20] Lancaster University and ARMINES. Ping reactive programming framework specification. Technical report, PING Consortium, 2001.

[21] M. Vellon, S. Drucker, K. Marple, and D. Mitchell. The architecture of a distributed virtual worlds system. In *Proceedings of the 4th USENIX Conference on Object-Oriented Technologies and Systems (COOTS)*, Santa Fe, New Mexico, April 1998.

[22] R. C. Waters and D. B. Anderson. The java open community version 0.9 application program interface. Technical report, Mitsubishi Electric Research Lab, February 1997.