DESIGN AND IMPLEMENTATION OF A DISTRIBUTED VIRTUAL ENVIRONMENT DEVELOPMENT PLATFORM

Wang Yongjun, Wang Yijie, Zhao Long, Li Sikun
National University of Defense Technology, P.R. China

ABSTRACT

A novel design method of the distributed virtual environment development platform is studied in this paper. First, a language model based on complex behaviors for virtual environment is proposed to improve the ability of behavior modeling. Then a computing model of virtual environment based on the complex behavior is proposed to formalize the simulation computing unit of virtual environment extended to build up an distributed architecture of virtual environment application. The performance requirements of virtual environment are analyzed, and the concept of scalability is redefined. Some strategies of dynamic load distribution are discussed to maintain the performance of distributed virtual environment. Finally, *YHVRP*, a prototype system of distributed virtual environment development system, is introduced.

1. Motivation

Currently there is little research work that focuses on systematically building up a distributed virtual environment development platform. Obviously this relates to modeling of the virtual environment, system architecture, etc, but there exist many weaknesses in all these research areas:

- *Weak support of complex behavior modeling.* Little attention is paid to behavior modeling in many current virtual environment development systems. For example, some famous distributed virtual environment development systems, such as MR [1], DIVE [2], AVIARY [3] and SPLINE [4], make very weak support to complex behavior modeling and their modeling languages are used only to describe simple behaviors. Thus it's very important to design a modeling language for more wider application areas of virtual environment.

- *Lack of facilities to keep special performance of virtual environment.* Highly dynamic is the main characteristic of virtual environment computing, and how to keep the special performance real time interactivity and fidelity when the computing load changes is also a very important problem. But in most of the current virtual environment application development systems, only fundamental functions are provided to improve the system performance, and most of the optimization work has to be done by application developers, so it's very inefficient when developing complex applications.

In this paper a novel design method of a distributed virtual environment development platform is studied especially for the solution of the above two problems. In Section 2 a language model based on complex behaviors is proposed that formalizes the representation of virtual environment and can define the control relation between the behaviors of entities and their time constraints flexibly. In Section 3 a computing model of virtual environment based on complex behavior is proposed which is extended to build up a distributed architecture for a virtual environment application. In Section 4 the performance of the virtual environment is analyzed and the concept of scalability is redefined. Some strategies for dynamic load distribution are discussed in connection with maintaining the performance of distributed virtual environment. Finally, in Section 5 a prototype system for a distributed virtual environment development named *YHVRP* is introduced.

2. Language Model Based on Complex Behaviors

It may be very difficult to build up a general behavior model because different entities in different applications have different behavior models. Thus discussing basic and general model languages to represent different behaviors may be more practical.

2.1 APPLICATION REQIREMENT

The entity in a virtual environment has different meanings in different applications, so a general modeling language should have the ability of a general representation, and can represent the following behaviors:

- all kinds of agent behaviors, such as intelligent behavior and physical behavior;
- interaction between entities;
- the behaviors as the concept of general simulation.

The other basic application requirements the modeling language should satisfy are as follows:

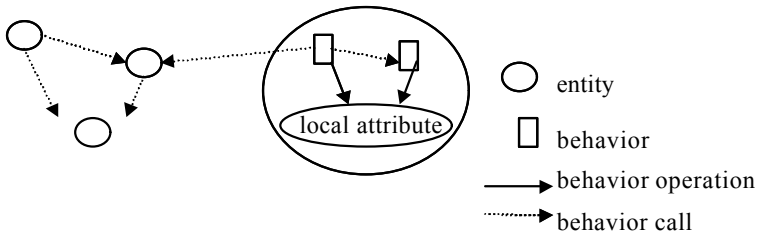
- the representation of time constraint;
- the representation of conflict resolution of behaviors.

2.2 LANGUAGE MODEL BASED ON COMPLEX BEHAVIORS

Virtual environment consists of many entities, the representation of which is given in Fig.1(a). An Entity consists of some behaviors and local attributes, the representation of which is given in Fig.1(b). The execution body of a behavior includes two parts: (1) behavior operations which are references to local attributes of entity, and (2) behavior calls which motivate other behaviors of the same entity or of other entities. The language model will be shown in the following definitions.

Def1 Entity: Entity can be represented as a 3- tuple:
Entity ::=<*EntityName, AttribSet, BehSet*>
where *AttribSet* is the set of local attributes, and *BehSet* is the set of behaviors.

Def2 Complex Behavior: Complex Behavior is represented as a 5- tuple:
ComplexBeh ::=
<*EntityName,BehName,TrigCond,BehBody,ConstraintTime*>,
where TrigCond *is the condition under which complex behavior is triggered;*BehBody *is the execution body of behavior;* ConstraintTime *is the time constraint specified by the user.*



(*a) virtual environment* *(b) entity*

Figure 1: Virtual environment and its entities

In the language model behaviors are classified into two types: **GeneralBeh,** which operates to the entity attributes, and **ConflReslBeh,** which arbitrates and resolves the conflicts between different behaviors. The conflict resolution behavior can set the state of general behaviors directly or readjust the execution structure of general behaviors. It is more advanced than the general behaviors. These two kinds of behaviors are distinguished by *TrigCond*, defined as follows:

Def3 TrigCond: TrigCond consists of two kinds of events:

TrigCond ::= *GeneralEvent* | *BehConfEvent*.

where *GeneralEvent* is the trigger condition of general behavior, which consists of system defined events (such as clock event, collision detection event, behavior call event) and user-defined events (such as some condition statements which should be satisfied to be true); *BehConfEvent* is the trigger condition of conflict resolution behavior, which is represented by a set of general behaviors. If the behaviors in the set are all active, the conflict happens.

Def4 BehBody: BehBody is represented as a 3- tuple:

BehBody::=

<InputAttrSet,OutputAttrSet, ProcessingProc>,

where *ProcessingProc* is behavior-computing procedure; *InputAttrSet* is the set of input attributes of the entity the behavior computing needs; *OutputAttrSet* is the set of output attributes of entity the behavior computing needs.

Def5 ProcessingProc:

ProcessingProc is programmed by a C-like language, in which there are some special functions as follows:

1. SetBehState(): set the state of general behavior, which includes potential, active, suspended and terminated.

2. PBehCall(): call a set of general behaviors parallelly at the same time. After all the executions of the behaviors in the set finish, the function finishes. This function provides the ability to describe more complex control relation among behaviors.

3. SetLock() and ReleaseLock(), lock the entity or the attributes of entity to prevent from behavior conflicts. According to the grammar description of BehBody, complex behaviors can be classified into two types: one is **AtomBeh**, which does not call other behaviors with PBehCall function and the other is **CombinedBeh**, which calls other behaviors with PBehCall function. The called behavior is named **SubBeh**.

According to the attributes behavior operates on, **GeneralBeh** can be classified into three types: The first is **PresentationBeh**, which operates only on the presentation attributes, which are the physical attributes related with the output of senses such as visual sense, aural sense and so on. The geometry model, speed and position are also examples of PresentationBeh. The second is **NonPresentationBeh**, which only operates to the non-presentation attributes, for example, the intelligent attribute. The third is **SynBeh**, which operates to not only on **PresentationBeh,** but also on **NonPresentationBeh**.

Def6 ConstraintTime: ConstraintTime specifies the time-related parameter of behavior execution, and is represented as a 2-tuple:

ConstraintTime ::= <BeginTime, DeadLine>,

where *BeginTime* is the time when the behavior is triggered, and *DeadLine* is the time interval during which the behavior should finish the execution.

DeadLine has two meanings: one is **ExecTime,** which is the time constraint of behavior execution; the other is **PresentTime,** which is the time constraint from the start of behavior execution to the time when the behavior-modified attributes are presented and sensed by users. The related time of behavior is represented in Figure 2:

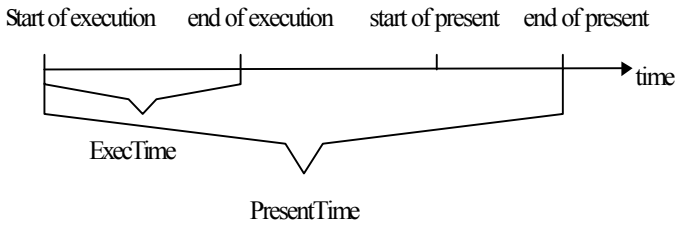


Figure 2: Related times of behavior

BeginTime can be specified by users, or determined by the time the behavior is triggered when virtual environment runs.

3. Computing Model of Virtual Environment Based on Complex Behavior

Usually the virtual environment system involves three computing units: input unit, simulation unit, and present unit. This means the computing model of virtual environment is organized around users.

The language model based on complex behaviors formalizes the description of virtual environment, and a computing model of virtual environment based on complex behaviors is proposed to formalize the representation of virtual environment computing. In that model the entity represents the coarse-grained computing unit, and the complex behavior owned by the entity represents the fine-grained computing unit.

There exist three kinds of entities in a typical virtual environment: the first is *UserEntity* whose main behavior is to transmit the interaction message between the user and the virtual environment. The second is *GeneralEntity*, which is the object simulated by different applications. The third is *RenderEntity,* whose main behavior is to render and display the virtual scene according to the viewpoint and view angle of the user.

The situation where computing in a virtual environment is processed by a single CPU is considered. The computing in a virtual environment based on complex behaviors consists of loops of the simulation steps. In every simulation step, there are three phase operations: In the first phase, UserEntity processes the input data. In the second phase, all GeneralEntitys compute their active behaviors and modify corresponding attributes. In the third phase, RenderEntity does the present computing.

The emphasis of our study was the second phase. Figure 3 shows the detail. It's noticeable that the ConflReslBeh must be executed before GeneralBeh.

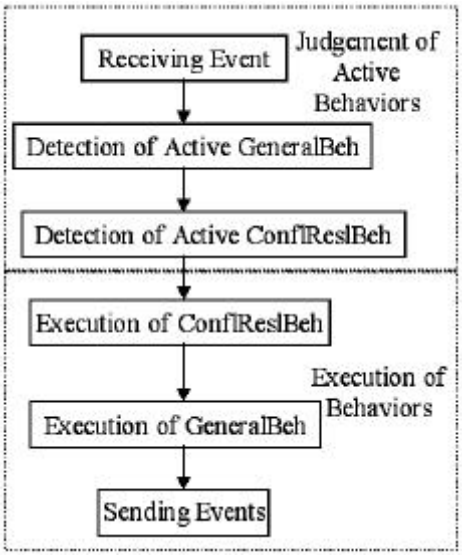


Figure 3: Computing and entity behavior

4. Scalable Distributed Virtual Environment System

4.1 DISTRIBUTION OF COMPUTATION AND DATA

Two general concepts [5], *Data* and *Computation***,** are used to describe the objects processed by a distributed virtual environment system and the processing itself. They are different for each kind of application. The efficiency of a distributed virtual environment depends on its management of the distribution of Data and Computation.

The goal of Data distribution is to try to find the best way to place the newest data on the corresponding node needed by the computation. Data distribution can be determined by application developers before running the system, and it never changes. It can also be adjusted dynamically when the system is running to achieve better performance. The dynamic adjustment occurs in two situations: 1) dynamic control by the application developer in the program; 2) the distribution of computation as the data changes dynamically.

The goal of computation distribution is to improve parallel computing in the virtual environment and achieve better performance. The distribution of computation can be done statically or dynamically. Its dynamic adjustment includes *computation migration*, which can reduce the computing load of node, and *computation copy*, which builds the copy of computation to reduce the frequency and delay of network transmission. The goal of dynamic adjustments is to reduce the response time.

In fact, the distribution of Data and Computation is related not only to the computational characteristic of different applications, but also to performance of the network of the distributed virtual environment.

In the language model, *Data* refers to the attributes of entities in the virtual environment, and *Computation* refers to the behaviors of entities. The computation and communication cost of nodes comprises that of behaviors. Thus the distribution of Data and Computation equals the distribution of behaviors.

4. 2 A SCALABLE DISTRIBUTED STRUCTURE OF VIRTUAL ENVIRONMENT

The concept of scalability of a distributed virtual environment system was first defined by Macedonia [6], who has developed many important technologies for the DIS system in the famous NPSNET_IV system [7]. Those technologies fit all kinds of distributed virtual environments. The definition of scalability is when the quantity of concurrent dynamic entities and users in the distributed virtual environment increases greatly, the software architecture will support it without modification.

In fact, this definition is neither clear nor precise, and a more general definition of scalability is proposed: When the computing load and communication load of a distributed virtual environment are increased greatly, the software architecture will keep the performance of the virtual environment without modification.

To achieve the scalability, the most advanced technology monitors the computing load and communication load, adjusting dynamically and automatically the distributed virtual environment system to maintain the performance.

An advanced scalable architecture of distributed virtual environment system is proposed as shown in Figure 4 in which there are three kinds of nodes: *UserNode, NonUserNode,* and *VE Server*.

UserNode is the process by which users join in and interact with the virtual environment, so UserNode usually is tied with some virtual reality I/O devices such as an HMD, stereoscopic glasses, data-glove, or high performance CIG (Computer Image Generator). When the complexity and quantity of behavior computing increase greatly, the processing capability of UserNode cannot satisfy the performance requirement in the virtual environment. Thus NonUserNode is used to process the computing load distribution from UserNode. Usually there are lots of UserNodes and NonUserNodes which can join the virtual environment dynamically.

The work of the VE Server is to cooperate with UserNode and NonUserNode to control the process of dynamic load distribution. Usually there is only one VE Server in a local network.

4.3 PERFORMANCE OF DISTRIBUTED VIRTUAL ENVIRONMENT

All of us know that real-time interactivity and fidelity are the performance requirements in virtual environment.

- *Real-Time Interactivity:* Real-time interactivity means that the computing in a virtual environment is fast enough to respond to the input of the user and output the changes in realtime. For example, the virtual scene will change as soon as the viewpoint of the user changes. If the lag exceeds some threshold, the feeling of immersion and the interactivity will be damaged [8].

- *Fidelity of Behaviors:* The fidelity of virtual environment consists of two parts: the fidelity of presentation (for example, the high quality image) and that of the computing simulation. Much research has been done on the first kind of fidelity, and many matured light models and rendering algorithms have been proposed to get the high-fidelity image. However, there has been little work done on the second kind.

Depending on the complex behaviors of the virtual environment, the fidelity of the computed simulation is determined by the fidelity of entity behaviors. How does one achieve fidelity of behaviors? There are four relevant aspects: (1) modeling behaviors to ensure the fidelity of the behavior models; (2) a constant frame rate is important to maintain fidelity of interaction between users and the virtual environment [9]; (3) the time fidelity of the entities' behaviors. Time fidelity means that the actual execution of a behavior should finish within the time constraint that the user or the real world experience has defined; (4) Synchronization of nodes. If the frame rate of each UseNode can be maintained constant by adjusting system load supported by the scalable architecture dynamically, the synchronization of nodes can be supported naturally.

4.4 DYNAMIC LOAD DISTRIBUTION OF COMPLEX BEHAVIOR

To maintain the performance of a distributed virtual environment, dynamic load distribution of virtual environments is needed. There has been little work on the dynamic load distribution of virtual environments. Currently typical distributed virtual environment systems put the emphasis on the computing load distribution only to increase the number of entities. In these systems the entity is the fundamental unit of distribution. For example, the function to migrate entities is provided in AVIARY and DIVE, but the migration must be controlled by application developers. In WVAES [10] a simple algorithm balances load on the basis of the partition of virtual space.

With complex behavior it is not enough to distribute the load only in terms of the partition of virtual space. A new dynamic load distribution of complex behaviors is proposed to develop the fine-grained parallelism of computing. Its scheduling evidence focuses on prediction of computation and communication costs based on the time constraint of behaviors, which are the substance of virtual environment computing.

Because in most situations it is periodic behaviors that contribute most to the computing load of virtual environment, they are distributed dynamically. Similar to the traditional dynamic realtime load distribution algorithm, the strategy of scheduling periodical behaviors consists of the following: the opportunity to schedule behaviors, the choice of migrated behaviors, and the choice of node to receive the migrated behaviors. The second is very important.

There are two heuristic strategies for the choice of migrated behaviors.

- *For migrated grain of complex behaviors.*

Complex behaviors make the grain of migrated behavior abnormal. Considering that PresentationBeh and NonPresentationBeh have different effects on the computing load and the communication load of the virtual environment, a heuristic strategy is determined as follows: For NonPresentationBeh all of its SubBehs can be scheduled; for PresentationBeh, only the AtomBeh can be scheduled. For SynBeh all of its SubBehs that are of NonPresentationBeh can be scheduled.

- *For the priority of behavior to be scheduled out.*

Two terms are defined to represent the priority of behaviors to be scheduled. The time the *Laxity* behavior costs is the time that could otherwise be used to time constrained behavior. However, there is a time limitation on it. *SchedulOutValue* of behavior depends on the prediction of computing cost, the communication cost of the behavior. and the cost of the node at the same time. The computing cost is larger than the communication cost, making the SchedulOutValue larger.

When choosing the migrated behaviors the priority of the behavior whose Laxity is the largest is the highest to be scheduled out. For the behaviors with the same Laxity, behavior whose SchedulOutValue is the largest has the highest priority of being scheduled out.

5. Prototype System

What has been discussed above is the theoretical framework which will direct the design and implementation of a real distributed virtual environment system. A prototype system of a distributed virtual environment development platform, *YHVRP*, is implemented to test the efficiency. YHVRP is based on a popular general platform such as Windows NT or a local ethernet.

5.1 YHVML

YHVRP provides application developers with the modeling language named YHVML that embodies the language model of a virtual environment based on complex behaviors and supports 3D geometry modeling at the same time. YHVML is implemented by its interpreter, which can very easily implement the computing model of virtual environment based on complex behaviors without regard to different operating systems. That is, YHVML is platform-independent. YHVRP also provides a function to reliably test the execution of the behavior of an entity programmed in YHVML, and uses that function for performance prediction.

5.2 YHVRP

The architecture of YHVRP was shown in Figure 4. The architecture of UserNode is shown in Figure 5. The Manager of the Database manages the attributes and behaviors of entities which are loaded into memory. The Scheduler predicts the computing cost and communication cost of the node for the purpose of deciding if the computing and communication cost overload thereby running the algorithm of dynamic load distribution. In the YHVRP system all communications between VE Server, UserNode, and NonUserNode are processed by the Data Distributor, which is implemented with the WinSock API.

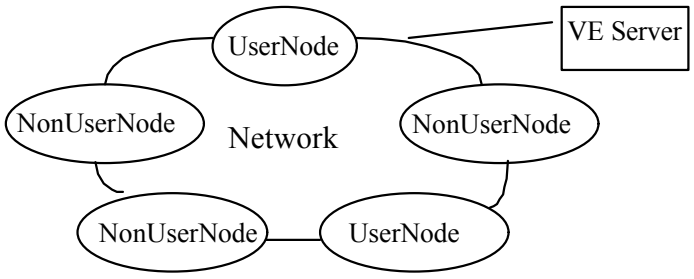


Figure 4: Scalable architecture of a distributed virtual environment

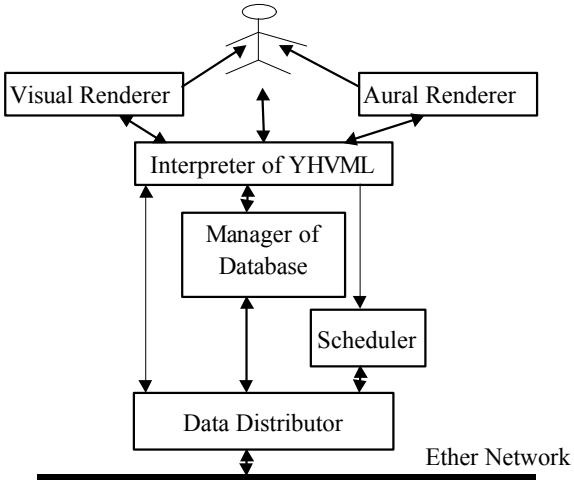


Figure 5 Structure of UserNode

In YHVRP some parameters related to general software, the hardware platform, and the application must be determined first**.** Some simple application examples involving behavior animation [11] that have been programmed with YHVML and run on YHVRP have demonstrated the efficiency and flexibility of YHVRP.

6. Conclusion

With the proposal of a language model of virtual environments based on complex behavior, a series of new concepts have been proposed, including the computing model, a performance analysis of a virtual environment, scalable architecture of a distributed virtual environment, and a strategy of dynamic load distribution of periodical complex behaviors. All of these provide a new extensible research framework for distributed virtual environment systems based on our research on higher-level behavior modeling methods and other methods of dynamic load distribution.

REFERENCES

[1] C. shaw and M. Green, "Decoupled Simulation in Virtual Reality with the MR toolkit", ACM Transactions on Information systems, Vol.11, No. 3, July 1993, pp. 287-317.

[2] O. Hagsand, "Interactive Multimedia VEs in the DIVE System", IEEE Multimedia, Vol. 3, No.1, March 1995, pp. :30-39.

[3] D.N. Snowdon, "AVIARY: Design Issues for future Large scale Virtual Environments", Presence: Teleoperators and Virtual Environments, Vol. 3, No. 4, Fall 1994, pp. 220-241.

[4] J.W. Barrus, R.C. Waters, and D.B. Anderson, "Locals: Supporting Large Multiuser Virtual Environments", IEEE Computer Graphics and Applications, Vol. 16, No. 6, April 1996, pp. 50-57.

[5] R. Hawkes, "A Software Architecture for Modeling and Distributing Virtual Environments", Ph.D. thesis, University of Edinburgh, England, 1996.

[6] M.R. Macedonia, "A Network Software Architecture for Large Scale Virtual Environments", Ph.D thesis, Naval Postgraduate School, USA, June 1995.

[7] M.R.Macedonia et al. "NPSNET:A Network Software Architecture for Large-Scale virtual Environments", Presence:Teleoperators and Virtual Environments, Vol.3, No. 4, Fall 1994, pp. 265-287.

[8] M.M.Wloka, "Lag in Multiprocessor Virtual Reality", Presence: Teleoperators and Virtual Environments, Vol. 4, No. 1, Winter 1995, pp. 50-63.

[9] R. Hawkes, "Update Rates and Fidelity in Virtual Environments", Virtual Reality, Vol.1, No.2, 1995, pp:99-108.
[10] R. Kazman, "Load Balancing, Latency Management and Separation of Concerns in a Distributed Virtual World", In Zomaya A.(Ed.), Parallel Computer - Paradigms and Applications, Van Nostrand, 1996.
[11] Y.J. Wang, "The Research of modeling language and system architecture of Distributed Virtual Environment and its Implementation", Ph.D. Thesis, National University of Defense Technology, Chansha, P.R.China, 1998.

BIOGRAPHY

Yongjun Wang is an Assistant Professor at the Department of Computer, National University of Defense Technology. He received M.S. and Ph.D. degrees from the National University of Defense Technology, both in computer architecture. He has published more than 30 papers. His current research interests include distributed VR, medical VR, simulation, and databases.

Contact information:

Dr. Yongjun Wang
Department of Computer, National University of Defense Technology
Changsha, Hunan, P.R.China, 410073
Phone: +86-0731-4506606
Email: mailto:yjwang@nudt.edu.cn