A Comparison Study of Modern Heuristics for Solving the Partitioning Problem in Distributed Virtual Environment Systems

P. Morillo[1], M. Fernández[1] and J. M. Orduña[2]

[1] Instituto de Robótica. Universidad de Valencia. SPAIN.
[2] Departamento de Informática. Universidad de Valencia. SPAIN
Pedro.Morillo@uv.es, Juan.Orduna@uv.es

Abstract. Fast Internet connections and the widespread use of high performance graphic cards are making Distributed Virtual Environment (DVE) systems very common nowadays. However, there are several key issues in these systems that should still be improved in order to design a scalable and cost-effective system. One of these key issues is the partitioning problem. This problem consists of efficiently assigning clients (3-D avatars) to the servers in the system.
In this paper, we present a comparison study of different modern heuristics for solving the partitioning problem in DVE systems, as an alternative to the ad-hoc heuristic proposed in the literature. Performance evaluation results show that some of the heuristic methods can greatly improve the performance of the partitioning method, particularly for large DVE systems. In this way, efficiency and scalability of DVE systems can be significantly improved.

1 Introduction

The widespread use of both fast Internet connections and also high performance graphic cards have made possible the current growth of Distributed Virtual Environment (DVE) systems. These systems allow multiple users, working on different computers that are interconnected through different networks (and even through Internet) to interact in a shared virtual world. This is achieved by rendering images of the environment as if they were perceived by the user. Each user is represented in the shared virtual environment by an entity called *avatar*, whose state is controlled by the user input. Since DVE systems support visual interactions between multiple avatars, every change in each avatar must be propagated to the rest of the avatars in the shared virtual environment.

DVE systems are currently used in different applications [17], such as collaborative design [16], civil and military distributed training [14], e-learning [15] or multi-player games [11]. Due to their different requirements, communication rate of avatars may significantly differ among these different applications. One of the key issues in the design of a scalable DVE system is the *partitioning problem* [12]. It consists of efficiently assigning the workload (avatars) among different servers in the system. The partitioning problem may seriously affect the overall performance of the DVE system, since it determines not only the workload each server is assigned to, but also the inter-server communication requirements (and therefore the network traffic). Some methods for solving the partitioning problems have been already proposed [12, 13, 17]. These methods

provide efficient solutions even for large scale DVE systems. However, there are still some features in the proposed methods that can be improved. For example, different heuristic search methods can be used for finding the best assignment of clients to servers, instead of using ad-hoc heuristics.

In this paper, we present a comparison study of several heuristics for solving the partitioning problem in DVE systems. We have implemented five different heuristics, ranging over most of the current taxonomy of heuristics: Genetic Algorithms (GA) [8], two different implementations of Simulated Annealing [10], Ant Colony Systems (ACS) [4], and Greedy Randomized Adaptive Search (GRASP) [3]. Performance evaluation results show that the execution cost of the partitioning algorithm (in terms of execution times) can be dramatically reduced, while providing similar or even better solutions than the current best solutions, provided by the ad-hoc heuristic proposed in [13]. We plan to use these results to improve the efficiency and scalability of our DVE system, a collaborative driving simulator [7].

The rest of the paper is organized as follows: Section 2 describes the partitioning problem and the existing proposals for solving it. Section 3 describes the proposed implementations of the heuristics considered for this study. Next, Section 4 presents the performance evaluation of the proposed heuristics. Finally, Section 5 presents some concluding remarks.

2 The Partitioning Problem in DVE Systems

Architectures based on networked servers are becoming a de-facto standard for DVE systems [17, 12]. In these architectures, the control of the simulation relies on several interconnected servers. Multi-platform client computers must be attached to one of these servers. When a client modifies an avatar, it also sends an updating message to its server, that in turn must propagate this message to other servers and clients. Servers must render different 3D models, perform positional updates of avatars and transfer control information among different clients. Thus, each new avatar represents an increasing in both the computational requirements of the application and also in the amount of network traffic. When the number of connected clients increases, the number of updating messages must be limited in order to avoid avoid a message outburst. In this sense, concepts like areas of influence (AOI) [17], locales [1] or auras [9] have been proposed for limiting the number of neighboring avatars that a given avatar must communicate with.

Depending on their origin and destination avatars, messages in a DVE system can be intra-server or inter-server messages. In order to design a scalable DVE systems, the number of intra-server messages must be maximized. Effectively, when clients send intra-server messages they only concern a single server. Therefore, they are minimizing the computing, storage and communication requirements for maintaining a consistent state of the avatars in a DVE system. Lui and Chan have shown the key role of finding a good assignment of clients to servers in order to ensure both a good frame rate and a minimum network traffic in DVE systems [12, 13]. They propose a quality function, denoted as C_p, for evaluating each assignment of clients to servers. This quality function takes into account two parameters. One of them consists of the computing workload generated by clients in the DVE system, denoted as C_p^W. In order to minimize this pa-

rameter, the computing workload should be proportionally shared among all the servers in the DVE system, according to the computing resources of each server. The other parameter of the quality function consists of the overall inter-server communication cost, denoted as C_p^L. In order to minimize this parameter, avatars sharing the same AOI should be assigned to the same server. Quality function C_p is defined as

$$C_p = W_1 \, C_p^W + W_2 \, C_p^L \tag{1}$$

where $W_1 + W_2 = 1$. W_1 and W_2 are two coefficients that weight the relative importance of the computational and communication workload, respectively. These coefficients should be tuned according to the specific features of each DVE system. Using this quality function (and assuming $W_1 = W_2 = 0.5$) Lui and Chan propose a partitioning algorithm that re-assigns clients to servers [13]. The partitioning algorithm should be periodically executed for adapting the partition to the current state of the DVE system as it evolves (avatars can join or leave the DVE system at any time, and they can also move everywhere within the simulated virtual world). Lui and Chan also have proposed a testing platform for the performance evaluation of DVE systems, as well as a parallelization of the partitioning algorithm [13].

Some other approaches for solving the partitioning problem have been also proposed. One of them groups avatars following regular distributions [2]. In order to ensure good performance, this algorithm generate a number of regular distributions equal to the number of servers in the DVE system. However, this proposal does not obtain good performance when avatars are located following a non-uniform distribution. Another different approach rejects dynamic concepts associated to avatars like AOI, aura or locale [18]. Although this approach provides a fast way of solving the partitioning problem, the performance of the static partitioning is quite low when avatars show a clustered distribution. In this case, the servers controlling the areas of the clusters are overloaded, increasing the overall cost of the quality function.

The partitioning method proposed by Lui and Chan currently provides the best results for DVE systems. However, it uses an ad-hoc heuristic. We propose a comparative study of several heuristics, ranging over most of the current taxonomy of heuristics, in order to determine which one provides the best performance when applied to the partitioning problem. In this study we will follow the same approach of Lui-Chan: using the same quality function, we will obtain an initial partition (assignment) of avatars to servers, and then we will test the implementation of each heuristic to provide a near optimal assignment.

3 Heuristic descriptions

In this section, we present five implementations of different heuristics for solving the partitioning problem in DVE systems. Following the approach presented by Lui and Chan [13] (and using the same quality function C_p), the idea is dynamically applying a heuristic search method that provides a good assignment of clients to servers as the state of the DVE system changes. In this section, we describe the implementation of each heuristic search method and the tuning of its parameters for solving the partitioning problem.

All of the implemented heuristics start from an initial partition (assignment) of avatars. We tested several clustering algorithms for obtaining this initial partition. Although they are not shown here due to space limitations, we obtained the best results for a *density-based algorithm* (DBA) [5]. This algorithm divides the virtual 3D scene in square sections. Each section is labeled with the number of avatars that it contains (na), and all the sections are sorted (using Quick-sort algorithm) by their na value. The first S sections in the sorted list are then selected and assigned to a given server, where S is the number of servers in the DVE system. That is, all the avatars in a selected region are assigned to a single server. The next step consists of computing the mass-center (mc) of the avatars assigned to each server. Using a round-robin scheme, the algorithm then chooses the closest free avatar to the mc of each server, assigning that avatar to that server, until all avatars are assigned. Since the assignment of avatars follows a round-robin scheme, this algorithm provides a good balancing of the computing workload (the number of avatars assigned to each server does not differ in more than one). On other hand, avatars that are grouped in a small region and close to the mass-center of a server will be assigned to that server by the density-based algorithm. However, since these avatars are located so closely, they will probably will share the same AOI. Therefore, the density-based algorithm also provides an initial partition with low inter-server communication requirements for those avatars.

However, the assignment of avatars equidistant or located far away from the mass-centers is critical for obtaining a partition with minimum inter-server communication requirements (minimum values of the quality function C_p), particularly for large virtual worlds with only a few servers. Density-based algorithm inherently provides good assignments for clustered avatars, but it does not properly focus on the assignment of these critical avatars. Each of the following heuristic methods should be used at this point to search a near optimal assignment that properly re-assigns these avatars.

Simulated Annealing (SA) This heuristic search method is based on a thermodynamic theory establishing that the minimum energy state in a system can be found if the temperature decreasing is performed slowly enough. Simulated Annealing (SA) is a heuristic search method that always accepts better solutions than the current solutions, and also accepts worse solutions according to a probability system based on the system temperature.

SA starts with a high system temperature (a high probability of accepting a worsening movement), and in each iteration system temperature is decreased. Thus, SA can leave local minima by accepting worsening movements at intermediate stages. The search method ends when system temperature is so low that worsening movements are practically impossible. Since the method cannot leave local minima, it can not find better solutions, neither (the algorithm ends when certain amount of iterations N are performed without finding better solutions).

Each iteration consists of randomly select two different critical avatars assigned to different servers. Then, the servers that two critical avatars are assigned to are exchanged. If the resulting value of the quality function C_p is higher than the previous one plus a threshold T, that change is rejected. Otherwise, it is accepted (the search method must decrease the value of the quality function C_p associated with each assignment).

The threshold T used in each iteration i of the search depends on the rate of temperature decreasing R, and it is defined as

$$T = R - \left(\frac{R \times i}{N} \right) \quad (2)$$

where N determines the finishing condition of the search. When N iterations are performed without decreasing the quality function C_p, then the search finishes.

As literature shows [10], the two key issues for properly tuning this heuristic search method are the number of iterations N and the temperature decreasing rate R. Although they are not shown here due to space limitations, we obtained the best results for SA method with N=3000 iterations. Performing more iterations increased the required execution times and it did not provide better values of C_p. Regarding to the rate of temperature decreasing, it did not have an effect on the required execution time of the algorithm. However, we obtained the best C_p values with R=1.25.

Random Search (RS) We have also implemented a simpler heuristic based on SA. We have denoted it as random search (RS), and it consists of the SA method when eliminating the system temperature. That is, in each iteration the threshold T is not considered, and if the resulting C_p is higher than the current minimum value, then that change is simply rejected. In this case, the only parameter to be tuned is the number of iterations N that determines the finishing condition. As in the SA case, the best results were obtained for N = 3000 iterations.

Ant Colony System (ACS) This heuristic search method is derived from the behavior of ant colonies when searching for food [4]. Each ant adds an hormone called *pheromone* to the path she follows, and the ants behind her will select their path depending on the pheromone each path contains (positive feedback). On other hand, pheromone evaporates at a given rate. Therefore, the associated pheromone for every path decreases if that path is not used during certain period of time (negative feedback). Evaporation rate determine the ability of the system for escaping from local minima.

The first step in the ACS method is to select the subset of border avatars from the set of all the avatars in the system. A given avatar is selected as a border avatar if it is assigned to a certain server S in the initial partition and any of the avatars in its AOI is assigned to a server different from S. For each of the border avatars, a list of candidate servers is constructed, and a certain level of pheromone is associated to all the elements in the list. This list contains all of the different servers that the avatars in the same AOI are assigned to (including the server that the avatar is currently assigned).

ACS method consists of a population of *ants*. Each ant consists of performing a search through the solutions space, providing a given assignment of the B border avatars to servers. The number of ants N is a key parameter of the ACS method that should be tuned for a good performance of the algorithm. Each iteration of the ACS method consists of computing N different ants (assignments of the B border avatars). When each ant is completed, if the resulting assignment of the B border avatars produces a lower value of the quality function C_p, then this assignment is considered as a partial solution, and a certain amount of pheromone is added to the servers that the border

avatars are assigned to in this assignment (each ant adds pheromone to the search path she follows). Otherwise, the ant (assignment) is discarded. When each iteration finishes (the N ants have been computed), the pheromone level is then equally decreased in all the candidate servers of all of the border avatars, according to the evaporation rate (the pheromone evaporates at a given rate). ACS method finishes when all the iterations have been performed.

In the process described above, each ant must assign each border avatar to one of the candidate servers for that avatar. Thus, a *selection value* is computed for each of the candidate servers. The selection value S_v is defined as

$$S_v = \alpha \times pheromone \ + \ \beta \times C_p \tag{3}$$

where $pheromone$ is the current pheromone level associated to that server, C_p is the resulting value of the quality function when the border avatar is assigned to that server instead of the current server, and α and β are weighting coefficients that must be also tuned. The server with the highest selection value will be chosen by that ant for that border avatar.

On other hand, when a partial solution is found then the pheromone level must be increased in those servers where the border avatars are assigned to in that solution. The pheromone level is increased using the following formula:

$$pheromone \ = \ pheromone \ + \ Q \times \frac{1}{C_p} \tag{4}$$

We have performed empirical studies in order to obtain the best values for α, β and Q coefficients. Although the results are not shown here due to space limitations, we have obtained the best behavior of the ACS method for $\alpha = 1.0$, $\beta = 7.0$ and $Q = 1000$. Additionally, we have tuned the values for the number of ants N, the pheromone evaporation rate and the number of iterations that ACS method must perform to obtain a near optimal partition. We obtained the best results for N=25 iterations, an evaporation rate of 1% and a population of 100 ants.

Genetic Algorithms (GA) This heuristic consists of a search method based on the concept of evolution by natural selection [8]. GA starts of an initial population (the initial partition) and then it evolves a certain number of *generations* (iterations), providing an *evolved population* (final solution).

The proposed implementation for solving the partitioning problem starts with a population composed of a set of elements called *genomes* or *chromosomes*. The number of chromosomes is the number of partial solutions that each iteration must provide. Each chromosome is defined by a descriptor vector containing a given assignment of avatars to servers.

Starting from the initial population, each generation (iteration) is found by exchanging some elements of the population, in such a way that in each of the N chromosomes two border avatars assigned to different servers are randomly chosen and exchanged. Thus, an iteration performed on a population of N chromosomes will produce a new population of $2N$. From these $2N$ chromosomes, the N elements with the lower value of C_p will be chosen.

GA is also capable of escaping from local minima due to the *mutation* process. In each iteration, a mutation consists of randomly changing the server assigned to one of the elements (chromosomes) of the population.

The main parameters to be tuned in GA search method are the population size P, the number of iterations N and the mutation rate M. Although they are not shown here due to space limitations, we obtained the best results for P=15 individuals, N= 300 iterations and M= 1%.

Greedy Randomized Adaptive Search (GRASP) This search method is a constructive technique designed as a multi-start heuristic for combinatorial problems [6]. In this case, the initial partition does not provide any assignment for the border avatars, and the GRASP method is used to make these assignments.

Each iteration consists of two steps: construction and local search. The construction phase builds a feasible solution choosing one border avatar by iteration, and the local search also provides a server allocation for the AOI of that border avatar in the same iteration, following the next procedure: First, the resulting cost C_p of adding each non-assigned border avatar to the current (initial) partition is computed. Since each border avatar can be assigned to different servers, the cost for assigning each border avatar to each server is computed, forming the *list of candidates (LC)* (each element in this list has the form (*non-assigned border avatar, server, resulting cost*). This list is sorted (using Quick-sort algorithm) by the resulting cost C_p in descendent order, and then is reduced to its top quartile. One element of this reduced list of candidates (RLC) is then randomly chosen (construction phase). Next, an extensive search is performed in the AOI of that selected avatar. That is, all the possible assignments of the avatars in the AOI of the selected avatars are computed, and the assignment with the lowest C_p is kept.

The quality of the solution provided by GRASP search method depends on the quality of the elements in the RLC, and the range of solutions depends on the length of the RLC. Thus, the main parameter to be tuned in this case is the number of *non-assigned avatars* N that the initial partition must leave. Although they are not shown here due to space limitations, we obtained the best performance for N=5 in the case of small virtual worlds and N=20 in the case of large virtual worlds (a detailed description of both a small and also a large virtual world is shown in the next section).

4 Performance Evaluation

In this section, we present the performance evaluation of the heuristics described in the previous section when they are used for solving the partitioning problem in DVE systems. Following the evaluation methodology shown in [13], we have empirically tested these heuristics in two examples of a DVE system: a small world, composed by 13 avatars and 3 servers, and a large world, composed by 2500 avatars and 8 servers. We have considered two parameters: the value of the quality function C_p for the partition provided by the search method and also the computational cost, in terms of execution time, required by the search method in order to provide that partition. For comparison purposes, we have also implemented the *linear optimization technique* (LOT) [13]. This

method currently provides the best results for the partitioning problem in DVE systems. In the case of small worlds we have also performed an exhaustive search through the solution space, obtaining the best partition as possible. The hardware platform used for the evaluation has been a 1.7 GHz Pentium IV with 256 Mbytes of RAM.

Since the performance of the heuristic search methods may heavily depend on the location of the avatars, we have considered three different distributions of avatars: uniform, skewed, and clustered distribution. Figure 1 shows the 2D spatial location of avatars in each of these distributions.

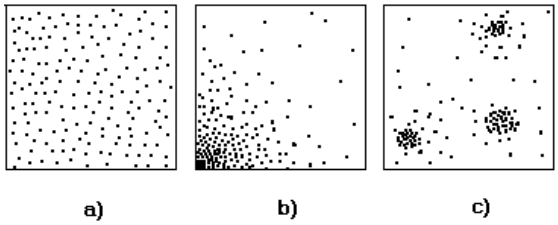

Fig. 1. Different avatar distributions in a DVE: (a) uniform, (b) skewed, and c) clustered

Table 1 shows the C_p values corresponding to the final partitions provided by each heuristic search method for a small virtual world, as well as the execution times required for each heuristic search method to obtain that final partition. It can be seen that all of the heuristics provide better (lower) C_p values than the LOT search method for a uniform distribution of avatars. For the skewed and clustered distributions, most of the heuristics also provides better C_p values than the LOT search method, and some of them (GA and SA methods) even provide the minimum value. However, the execution times required by most of the heuristics are longer than the ones required by the LOT method. Only GRASP method provides worse C_p values than the LOT method, but it requires much shorter execution times. Although these results does not clearly show which heuristic provides the best performance, they validate any of the proposed heuristics as an alternative to the LOT search method.

	Uniform distrib.		**Skewed distrib.**		**Clustered distrib.**	
	Time (s.)	C_p	*Time (s.)*	C_p	*Time (s.)*	C_p
Exhaustive search	3.411	6.54	3.843	7.04	4.783	7.91
Linear opt. method	0.0009	6.56	0.001	8.41	0.0011	8.89
Simul. Annealing	0.004	6.82	0.005	7.46	0.005	7.91
Random Search	0.002	7.37	0.005	8.06	0.006	8.35
Ant Colony System	0.0007	6.59	0.003	7.61	0.0024	8.76
Genetic Algorithms	0.002	6.54	0.003	7.04	0.005	7.91
GRASP	0.0002	7.42	0.0002	8.63	0.0003	11.88

Table 1. Results for a small DVE system

However, in order to design a scalable DVE system the partitioning method must provide good performance when the number of avatars in the system increases. That is, it must provide a good performance specially for large virtual worlds. Table 2 shows the required execution times and the C_p values obtained by each heuristic search for a large virtual world. In this case, all of the heuristics provides similar values of C_p than the LOT heuristic for the uniform distribution, while requiring much shorter execution times. When non-uniform distributions of avatars are considered, then all of the heuristics provide much better C_p values than the LOT method and they also require much shorter execution times than the LOT method. In particular, ACS method provides the best C_p values for the non-uniform distributions, requiring also the shortest execution time in the case of a clustered distribution.

	Uniform distrib.		**Skewed distrib.**		**Clustered distrib.**	
	Time (s.)	C_p	*Time (s.)*	C_p	*Time (s.)*	C_p
Linear opt. method	30.939	1637.04	32.176	3460.52	43.314	5903.80
Simul. Annealing	6.35	1707.62	13.789	2628.46	29.62	4697.61
Random Search	8.90	1687.55	13.826	2685.62	28.792	4676.22
Ant Colony System	5.484	1674.08	14.05	2286.16	23.213	3736.69
Genetic Algorithms	6.598	1832.21	14.593	2825.64	29.198	4905.93
GRASP	6.622	1879.76	13.535	2883.84	26.704	5306.24

Table 2. Results for a large DVE system

These results show that the performance of the partitioning algorithm can be significantly improved by simply using any of the proposed heuristics instead of the LOT method, thus increasing the scalability of DVE systems. In particular, ACS method provides the best performance as a partitioning algorithm.

5 Conclusions

In this paper, we have proposed a comparison study of modern heuristics for solving the partitioning problem in DVE systems. This problem is the key issue that allows to design scalable and efficient DVE systems. We have evaluated the implementation of different heuristics, ranging over most of the current taxonomy of modern heuristics. We have tested the proposed heuristics when applied for both small and large DVE systems, with different distributions of the existing avatars in the system. We have compared these results with the ones provided by the Linear Optimization Technique (LOT), the partitioning method that currently provides the best solutions for DVE systems.

For small virtual worlds, we can conclude that in general terms any of the implemented heuristics provides similar values of the quality function C_p, but the execution times required by the implemented heuristics are longer than the time required by the LOT search method. Although SA and GA methods provide the minimum value of the quality function, only GRASP method provides execution times shorter than the ones

required by the LOT method for all the avatar distributions. These results validates any of the proposed heuristics as an alternative to the LOT search method when considering small DVE systems. However, for large virtual worlds any of the proposed heuristics provides better C_p values and requires shorter execution times than the LOT method for non-uniform distributions of avatars. In particular, ACS method provides the best results. Since a scalable DVE system must be able to manage large amounts of avatars, we can conclude that these results validates ACS search method as the best heuristic method for solving the partitioning problem in DVE systems.

References

1. D.B.Anderson, J.W.Barrus, J.H.Howard, "Building multi-user interactive multimedia environments at MERL", in *IEEE Multimedia*, 2(4), pp.77-82, Winter 1995.
2. P. Barham, T.Paul, "Exploiting Reality with Multicast Groups", in *IEEE Computer Graphics & Applications*, pp.38-45, September 1995.
3. H. Delmaire, J.A. Díaz, E. Fernández, and M. Ortega, "Comparing new heuristics for the pure integer capacitated plant location problem", Technical Report DR97/10, Department of Statistics and Operations Research, Universitat Politecnica de Catalunya, Spain, 1997.
4. M. Dorigo and L. Gambardella, "Ant colony system: A Cooperative Learning Approach to the Traveling Salesman Problem", in *IEEE Trans. Evolut. Comput.* ,1997.
5. R.Duda, P.Hart, D.Stork, "Pattern Classification", *Ed.Wiley Intescience, 2000*, pp. 567-580.
6. Thomas A. Feo, Mauricio G.C, "Greedy Randomized Adaptive Search Procedures", *Resende Journal of Global Optimization*, 1995.
7. M. Fernández, I. Coma, G. Martín and S. Bayarri, "An Architecture for Optimal Management of the Traffic Simulation Complexity in a Driving Simulator", *Lecture Notes in Control and Information Sciences*, Springer-Verlag, Vol. 243, 1999. ISBN 1-85233-123-2.
8. Randy L. Haupt, Sue Ellen Haupt, "Practical Genetic Algorithms", *Ed. Willey*, 1997.
9. J.C.Hu, I.Pyarali, D.C.Schmidt, "Measuring the Impact of Event Dispatching and Concurrency Models on Web Server Performance Over High-Speed Networks", *Proc. of the 2nd. IEEE Global Internet Conference*, November.1997.
10. P.V. Laarhoven and E. Aarts, "Simulated annealing: Theory and applications", *Reidel Pub.*, Dordrecht, Holland, 1987.
11. Michael Lewis and Jeffrey Jacboson, "Game Engines in Scientific Research", in *Communications of the ACM*, Vol 45. No.1, January 2002.
12. John C.S. Lui, M.F.Chan, Oldfield K.Y, "Dynamic Partitioning for a Distributed Virtual Environment", *Department of Computer Science*, Chinese University of Hong Kong, 1998.
13. Jonh C.S. Lui, M.F. Chan, "An Efficient Partitioning Algorithm for Distributed Virtual Environment Systems", *IEEE Trans. Parallel and Distributed Systems*, Vol. 13, March 2002
14. D.C.Miller, J.A. Thorpe, "SIMNET: The advent of simulator networking", in *Proceedings of the IEEE*, 83(8), pp. 1114-1123. August, 1995.
15. Tohei Nitta, Kazuhiro Fujita, Sachio Cono, "An Application Of Distributed Virtual Environment To Foreign Language", in *IEEE Education Society*, October 2000.
16. J.M.Salles Dias, Ricardo Galli, A. C. Almeida et al. "mWorld: A Multiuser 3D Virtual Environment", in *IEEE Computer Graphics*, Vol. 17, No. 2, March-April 1997.
17. S.Singhal, and M.Zyda, "Networked Virtual Environments", *ACM Press, New York, 1999*.
18. P.T.Tam, "Communication Cost Optimization and Analysis in Distributed Virtual Environment", *M. Phil second term paper, Technical report RM1026-TR98-0412*. Department of Computer Science & Engineering.The Chinese University of Hong Kong. 1998.