SIACG 2002 - 1st Ibero-American Symposium on Computer Graphics
1-5 July 2002, Guimarães - Portugal

Virtual Reality in Cooperative Teleoperation

Aura Nancy Rodriguez Jean-Pierre Jessel Patrice Torguet
IRIT - Institut de Recherche en Informatique de Toulouse
Toulouse, France
{rodri,jessel,torguet}@irit.fr

Abstract

Virtual reality enables to build systems that enhance communication by facilitating information display. In teleoperation applications, virtual reality improves task performance thanks to intuitive manipulation and exploration of working environment. In more, an interactive 3D simulation allows to give "virtual assistance" to the operator by adding guides or actors, present or not in the real world, which aid the operator in task learning or execution. Our research in telerobotics systems involves the integration of (real or virtual) autonomous entities as assistants for teleoperation missions. This paper describes a platform for teleoperation systems prototyping that is able to manage teleoperated and autonomous entities. This platform, called ASSET, includes control of interaction and acting devices, 3D rendering and easy integration of user components. Our system is analysed in this paper and a concrete example of its applicability in cooperative teleoperation is shown.

Keywords

teleoperation, virtual reality, autonomous agents, testbed, robotics, prototyping

1. INTRODUCTION

Virtual reality enhances communication between a system and its users by making intuitive examination and manipulation possible. The use of virtual environments in a teleoperation context can improve task performance, thanks to the efficient display and visualization of complex temporal and spatial relationships [Stanney98]. In fact, a 3D world in the controller site of a teleoperation system provides a safe environment to the operator for exploring naturally the machines and objects of the working environment. Cooperative teleoperation or telesystems tries to improve the teleoperation process by supplying aid to the operator. These systems have long been recognized as a very useful technology for space exploration, and they can be integrated to a variety of others applications such as manipulation in nuclear processing plants, rescue, fire fighting, intervention operations in hazardous environments, security, chirurgy and rehabilitation [Murphy96]. In human-robot cooperation, it is important to develop an interface system that affords intuitive interactive communication. In teleoperation, the quality of the human-machine connection impacts performance, so an effective operator interface is critical for conveying information and feedback to the operator [Fong00].

By combining capabilities of both Virtual Reality and Artificial Intelligence, a system can be developed in order to overcome some of the current limitations of telesystems. Following this idea, we are developing a system, based on virtual environments, for cooperative teleoperation research. This system called ROVE (Robotics Virtual Environment) consists of a teleoperation system architecture that understands the events of the dynamic environment and a set of modules of behavioural simulation that control the autonomous or semi-autonomous robots. This paper describes ASSET (Architecture for Systems of Simulation and Training in Teleoperation), our system platform for telerobotics systems. ASSET is a Java-based system that includes a virtual environment where teleoperated, autonomous and semiautonomous entities can evolve. This document presents an overview of our system, and an example of application development, i.e. the ROVE system implementation, is described.

2. RELATED WORK

Research in virtual environments (VE) has included research in VE construction toolkits, VE software architectures and VE training projects such as NPSNET [Macedonia95], which is a multi-user, distributed virtual environment used to recreate complex military missions. The Distributed Interactive Virtual Environment (DIVE) is a virtual reality system allowing many users to explore a 3D space and interact with each other [Hagsan96]. VIPER, a system developed in our team [Torguet00] is also a generic, multi-user distributed virtual reality platform that is able to run on heterogeneous physical architectures. Minimal Reality Toolkit (MRToolkit) is a set of software tools for the production of virtual reality systems and other forms of three dimensional user interfaces; it includes device drivers, support programs and a language

for describing geometry and behavior [Shaw93]. Bamboo is a component framework enabling the development of virtual environments that loads dynamically language-specific plugins into and out of memory. Typically, plugins are archived and subsequently downloaded at runtime via HTTP from one or more developer-specified web servers over the Internet [Watsen98]. Avango (a.k.a. Avocado) replicates a Performer scene graph across all sites over the network. In addition to this functionality, it supports a scripting language for rapid application development [Tramberend99]. VR Juggler is an open source virtual platform for virtual reality application development, which provides dynamic reconfiguration, input abstraction and performance monitoring. VR Juggler provides researchers with a framework to simplify application development by hiding low-level details of virtual reality [Just98]. Blue-c is a collaborative virtual environment, which combines immersive projection with real-time video acquisition and advanced multimedia communication. Eventually, multiple blue-c portals, connected via high-speed networks, will allow remotely located users to meet, communicate and collaborate in a shared virtual space [Staad00].

Some systems flexibility allows their use as higher-level tools, as in the work described by Balcisoy [Balcisoy00]. This work presents a framework for testing object design in an augmented reality context. The definition of modelling object geometry is extended with modelling object behaviour to allow users to experiment with a large set of possibilities without having extensive knowledge on the underlying simulation system. This approach shows that users can decrease the time spent on prototype evaluation and have a realistic testing environment, like the system described by Jung et al. [Jung99], a VRML/Java-based virtual environment - populated with heterogeneous articulated agents - which serves as testbed for exploring principles for the design of autonomous agents. Diverse kinds of articulated agents can be integrated into the virtual environment by supplying VRML models of their visual appearance and agent control clients that follow a predefined communication protocol. In the interest of web-based telemanipulation, Ghiasi et al. have proposed a general framework of Java-based components [Ghiasi99]. This framework attempts to reduce the level of skill required to successfully develop a teleoperation device by providing mechanisms to interact with, and providing tools that allow to build different GUIs and to make extensions or modifications to existing functionalities.

Virtual reality allows operators to perceive and control complex systems in a natural way. The Virtual Environment Vehicle Interface (VEVI), a system developed by the NASA Ames Research Center's Intelligent Mechanism Group, exploits this ability. VEVI uses real-time, interactive, 3D graphics and position/orientation sensors to control different vehicles in a variety of diverse environments [Piguet95]. VEVI was designed and implemented to be modular, distributed and easily operated. The VEVI system was followed by VIZ, the latest version of the Autonomy and Robotics Area's developments of Virtual Environment for Vehicle Interfaces [Nguyen01]. VIZ is designed to be a generic modular and flexible VR-based interface that allows easy reconfiguration and rapid prototyping. VEVI and VIZ have shown that virtual reality interfaces can improve the operator task knowledge and provide valuable tools to understand and analyse the remote environment. The 3D simulation enables telerobot control even when communication rates are slow. Aditya et al, have used Java 3D in their telerobotic system for creating and manipulating a Mentor robot [Aditya00]. Robot simulation is chosen to substitute a real image from a camera for reducing communication latency and system traffic. This system is a sample of the efforts made to applying teleoperation systems to low bandwidth networks.

In this work we aims at providing a tool for helping development of telerobotics systems, following philosophy of experimental platforms for virtual reality systems. We have also adopted distributed simulation techniques for minimizing the network bandwidth use and object-oriented development to offer a modular, flexible and easy to use system. Furthermore, because one of major limitation of revised systems is that they are available on just one or on very few platforms, we have chosen Java and Java3d to develop our system, so that it can run on any platform without further changes.

Our system ASSET is a set of reusable Java components that offers to programmers the services related to teleoperation missions: communications, simulation, device management, etc. It provides a testbed for experimenting with behaviors, simulation models, new devices and the different possibilities of human-robot collaboration. ASSET is designed to facilitate the research and development of teleoperation applications by allowing dynamic integration of modules and by providing 3D simulation in order to ease experimentation.

3. SYSTEM OVERVIEW

Our system ASSET is a flexible prototyping system supporting the design and evaluation of new teleoperation systems. This system provides a modular design to easily make extensions and modifications, it allows reusing already built components and facilitates the integration phase because the functional interface of each module remains constant.

3.1 Architecture

The structure of our system is composed of a general communication server and two clients modules (one in the user side and one in the real system side).

As shown in the figure 1, the ASSET's architecture consists of:

- User Interface Manager. The User Interface Manager (UIManager) is responsible for the communication

between the system and the user; it handles the local simulation and the interaction devices. Communication between this module and the others system modules is managed by the Communications component.

- Real System Manager. This module controls the real system. It is very similar to the UIManager, but it controls the acting devices (robots). Real System Manager (RSManager) executes the commands and manages coherence between the real state and the simulated state in order to update the user's simulations.
- Administrator. The administrator coordinates the interactions between the participating entities, users and robots. It is a communication server that transmits the commands to effectors and the information from the real system to the users. It has a Coordination component to solve conflicts raised by orders of different users.

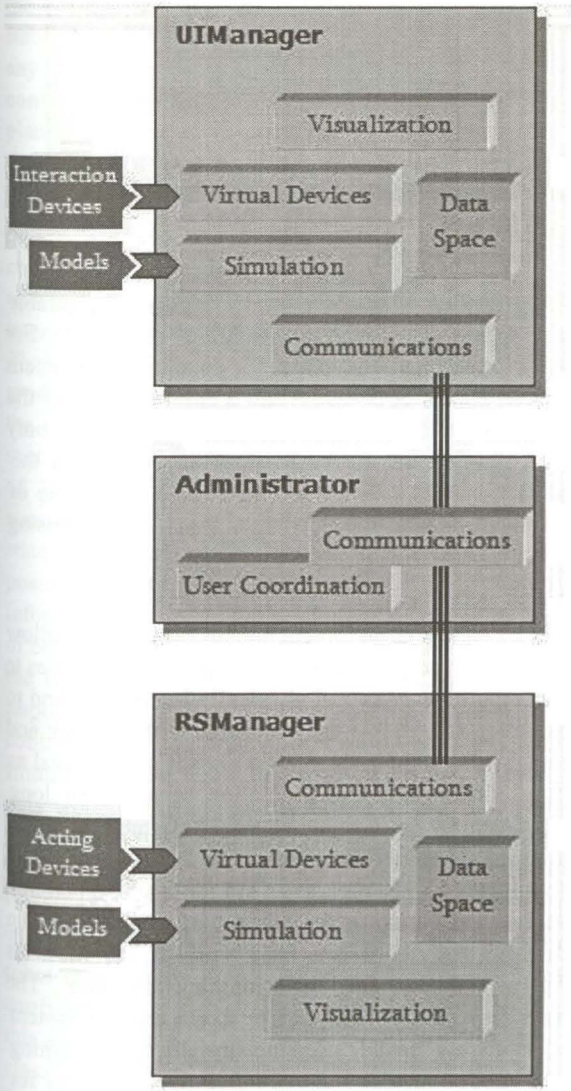

Figure 1: Architecture of the ASSET system

One key feature in ASSET is the abstract management of the basic components of a teleoperation application (i.e. devices management, communications, state...). In this way, applications can use their own classes or libraries to link into the general platform provided by the system. The user's classes make data processing while ASSET's core classes make data transporting. This allows to have domain application independence and to easily make modifications because core classes rest unchanged.

In ASSET we have defined the mechanisms that allow the interaction between the different modules and the different components of each module. To set the communication between the various components of a module, we have defined a data space that maintains device information, commands, and network messages. The data space notifies occurrences of a written event (when a component adds a message) allowing devices, simulation objects or communication units to recover and to process it. This capability allows having domain and devices independence.

3.2 Simulation

Simulation facilitates experimentation because it allows to evaluate a system in inaccessible environments or to face rare events. Moreover, by replicating simulation and models in every host participant, we have rapid feedback and filtration of invalid commands. In ASSET, there is a simulation module in each UIManager as well as in the RSManager. The simulator in the UIManager makes possible to give feedback to the user without delay while the simulator in the RSManager avoids the transmission of data at the end of each simulation interval.

After a simulation update, the RSManager checks if the difference between the real system state and the simulation state exceeds an error threshold (defined by the user at initialisation), and in this case, it sends the accurate values in order to update UIManager simulation objects. This technique aims at reducing network traffic by limiting the messages sent by the real system. To calculate the difference between real and simulated state, ASSET uses the conditions defined by the user for the specific application. The user defines the set of variables that constitute the system state and for each variable, the maximum error value. If there are one or more variables that have reached their maximum error value, simulation objects update is triggered. The state and error threshold are defined in a configuration file, so it is possible to easily change them to calibrate the system. Variables type and difference calculation algorithm can be also modified because the classes developed by the user to manage variables are instanced dynamically.

The simulation component in ASSET managers offers the services of 3D visualization, collision detection and Java3D/VRML2.0 models loading. ASSET allows defining behaviour for each simulation object, so it is possible to have entities with different degrees of autonomy sharing the same environment. Definition of object's geometry

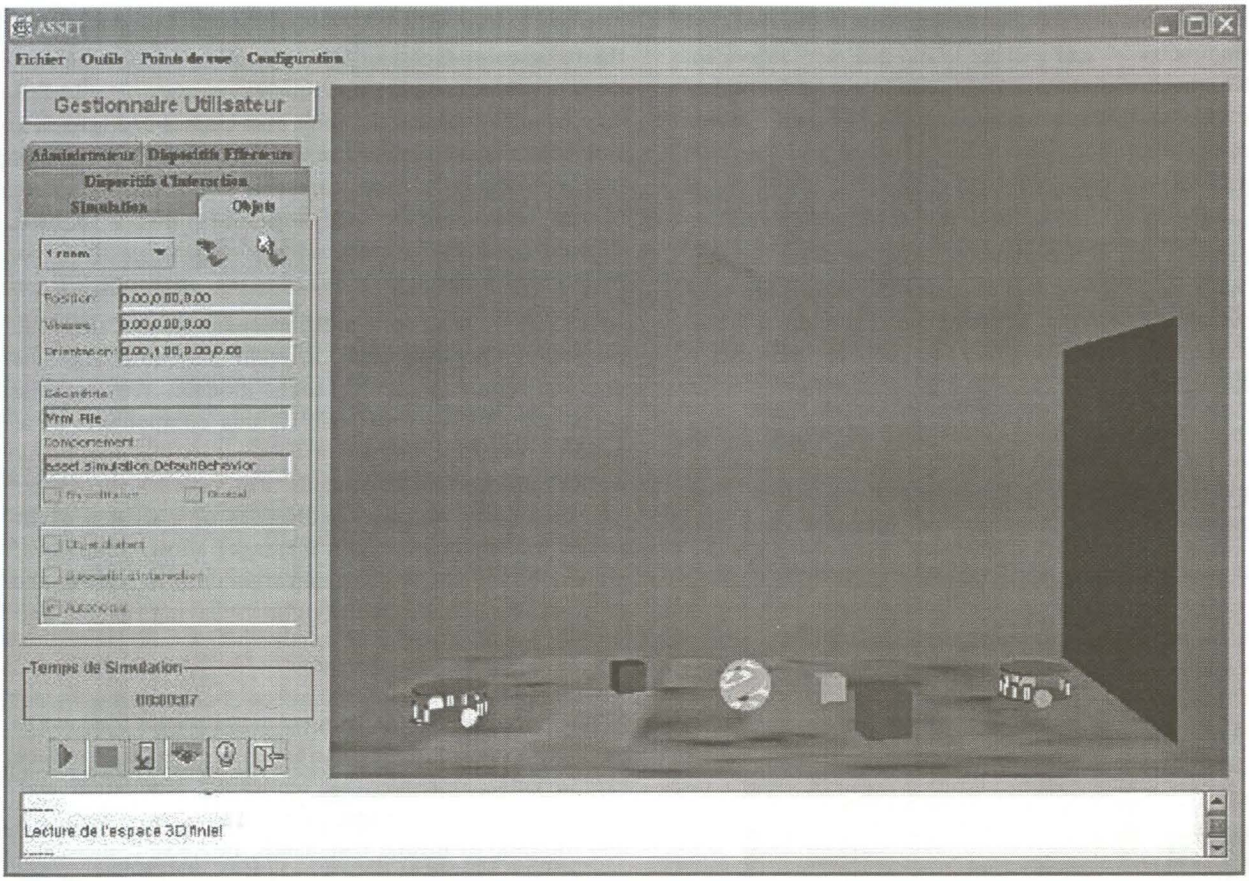

Figure 2: ASSET system interface

and behaviour in ASSET is made via a configuration file load at initialisation. The behaviours must implement a predefined interface to be successfully linked to the simulation objects.

ASSET can be used in simulation mode –i.e. without communicating with the real world- to allow training tasks and experimentation.

3.3 User Interface

The virtual environment is the communication medium between the robot and the user. The user interacts with the virtual robot by the means of an interaction device, defined by the user at initialisation. The user can also interact with the entire environment by navigating in the 3D world and by asking the state of different objects. We have defined virtual devices to offer a set of common services that can be implemented with various physical devices. The virtual device is a mediator between the real device and the system ensuring the independence between the application and the specific devices –only the virtual device can communicate with the actual device- .The user commands are sent to the simulation for validity checking and processing. Furthermore, if ASSET is used in teleoperation mode, commands are sent to the real entities.

With the architecture and mechanisms defined in the ASSET system, we have achieved important requirements like extensibility, adaptability and a highly configurable system that can integrate available resources. For example, for testing behaviours, the developer only needs to provide a geometric model and a control class for the entity. In order to test a new device, the developer only needs to implement the basic services defined in the virtual device. This feature allows reducing the time of development and let the developer focus on optimising his work.

4. CASE STUDY

Assistance robots complement human faculties and allow systems to take advantage of the computer's capacities to achieve repetitive tasks and physically hard work, and to use as better as possible the expert dexterity to look and to react at the right time. The assistant can be real as robots used for shared manipulation of heavy or long objects [Arai00] or in teleoperation task such as space exploration, dangerous works or unreachable environments (for instance, as an aid for disabled people). It can be useful also in the context of rehabilitation, in which technology is used to train a user in certain tasks and give practice and strengthening exercise. The assistant may also be virtual, like in the case of system Steve, a virtual human used to support team training [Rickel99]. The virtual humans serve as instructors for individual students and they can substitute for missing

team members, allowing students to practice tasks when some or all human instructors are unavailable.

This section presents a case study implemented using ASSET. The case study is representative of a range of applications in teleoperation using assistance robots (cooperative teleoperation). Using this case study we describe how to setup ASSET to manage interaction between teleoperated and autonomous entities. To present our idea we worked on ROVE, a system for facilitating cooperative teleoperation research. The goal for the human users and the autonomous robots is to achieve a common task in the virtual environment. The ROVE system consists of a reactive system (ASSET) that understands the events of the dynamic environment, and a behavioural simulation system called Adaptive Autonomous Agents (A^3), which controls the autonomous robots whose mission is to help the user.

The A3 system is a hybrid controller composed of 3 layers: a planner layer at the top, a reactive layer at the bottom, and a intermediate layer to combining/reconciling the others two. The intermediate layer consists of a finite state machine coupled with a buffer of messages. Then any "important" changes discovered by the low-level controller are passed back to the planner in a way that the planner can use to re-plan. The low-level controller (and thus the robot) is stopped if it must wait for the high level planner to tell it where to go [Heguy01].

Figure 2 shows the ROVE system with a user who cooperates with an autonomous robot. The user addresses commands to Khepera robot 1 by means of an interaction device and the user interface (UIManager). These commands are first sent to the local simulator to produce user feedback. If the result of that action triggers the simulation to another valid state, the commands are sent to the control module of the real system by the way of the Administrator. The RSManager updates the simulation allowing the behavioural unit A^3 to know the state of each simulation object, and to react as well as possible to the new environment. The A^3 unit filters information to obtain the needed data for making a choice of behaviour and update autonomous robot state. Finally, the two Khepera robots execute their commands. It is important to note that the autonomous robot is only controlled by the behavioural unit associated with the simulator of the RSManager, allowing the A^3 system to only work with valid global data. The unit in UIManager just feeds visual simulation. For this application, our updating variables are positions and collision state of the two robots.

The A^3 system is load and linked to the autonomous robot dynamically at configuration file reading. Devices configuration is also defined in the initialisation file. For this application, we have implemented the controller Java classes for the Khepera Robot, for a Logicad3d Spacemouse Classic and for a Microsoft SideWinder Force Feedback Joystick. In this way, we have shown that changes in hardware configuration (in this case, a change of interaction device) do not require any change in the application code.

The construction of this system allows us to verify the architecture and the ideas stated in ASSET. Thanks to the Reflection package of Java, it is possible to dynamically load classes into our system; therefore we can easily integrate new classes and models without modifying the system. Nevertheless, some difficulties have arisen, the most important being collision detection. In fact, in Java3D it is possible to know when collision is produced but it is very difficult to determine collision information. Currently we are using a simple algorithm of collision detection based on bounding spheres while we work on a more appropriate collision engine. This system is a work in progress and we are actually defining a test set that will allows us to evaluate ASSET and ROVE performance.

5. CONCLUSIONS

We have presented in this article ASSET, a Java-based testbed for teleoperation systems development, using virtual reality and artificial intelligence techniques. In this experimental environment, the developer can test new simulation models, behaviors and devices. This feature facilitates the utilization of ASSET in the creation of new telesystems and in making rapid tests and prototypes. Furthermore, to achieve platform independence, the implementation of ASSET is based on Java and Java3D. We have used our system in the building of a sample application, and we have been able to demonstrate the benefits of its flexibility, in particular for the dynamic integration of teleoperated and autonomous entities in the simulation and for the construction of applications independent specific devices. Future work will cover several aspects. At the current time, the most promising directions appear to integrating new interaction devices, enhancing multiuser cooperation, and studying human-machine interaction in supervision tasks.

6. REFERENCES

[Aditya00] A. Aditya and B. Riyanto. Implementation of Java 3D Simulation for Internet Telerobotic System. *Proceedings of IASTED International Conference Modelling and Simulation*, 2000.

[Arai00] H. Arai, T. Takubo, Y. Hayashibara and K. Tanie, Human-Robot Cooperative Manipulation Using a Virtual Nonholonomic Constraint. *Proceedings IEEE International Conference on Robotics and Automation*, 2000.

[Balcisoy00] S. Balcisoy, M. Kallmann, P. Fua and D. Thalmann. A framework for rapid evaluation of prototypes with Augmented Reality. *ACM Symposium on Virtual Reality Software and Technology*, 2000.

[Fong00] T. Fong, C. Thorpe and C. Baur. Advanced Interfaces for Vehicle Teleoperation: Collaborative Control, Sensor Fusion Displays, and Web-based Tools. Vehicle Teleoperation Interfaces Workshop,IEEE *IEEE International Conference on Robotics and Automation*, 2000.

[Ghiasi99] S. Ghiasi, M. Seidl and B. Zorn. A Generic Web-based Teleoperations Architecture: Details and Experience. *SPIE/Telemanipulator and Telepresence Technologies VII*, 1999.

[Hagsan96] O. Hagsan. Interactive Multiuser VEs in the DIVE System. *IEEE Multimedia Magazine*. 3(1), 1996.

[Heguy01] O. Heguy, N. Rodriguez, J-P. Jessel, Y. Duthen, and H. Luga. Virtual Environment for Cooperative Assistance in Teleoperation, *Proceedings of the International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision*. 9(3): II9-II12, 2001.

[Jung99] B. Jung and J-T. Milde. An Open Virtual Environment for Autonomous Agents Using VRML and Java. *4th Symposium on Virtual Reality Modeling Language*, 1999.

[Just98] C. Just, A. Bierbaum, A. Baker and C. Cruz-Neira. VR Juggler: A Framework for Virtual Reality Development. *2nd Immersive Projection Technology Workshop*, 1998.

[Macedonia95] M. Macedonia, D. Brutzman, M. Zyda, D. Pratt, and P. Barham. NPSNET: A Multiplayer 3D Virtual Environment over the Internet. *Proceedings of the AM-1995 Symposium on Interactive 3D Graphics*, 1995.

[Murphy96] R.R. Murphy and E. Rogers. Cooperative Assistance for Remote Robot Supervision. *Presence*. 5(2): 224-240, 1996.

[Nguyen01] L. Nguyen, M. Bualat, L. Edwards, L. Flueckiger, C. Neveu, K. Schwehr, M.D. Wagner and E. Zbinden. Virtual Reality Interfaces for Visualization and Control of Remote Vehicles, *Autonomous Robots*, 2001 (1):9-18, 2001.

[Piguet95] L. Piguet, T. Fong, B. Hine, P. Hontalas and E. Nygren. VEVI: A Virtual Reality Tool for Robotic Planetary Exploration. *Virtual Reality World Conference*, 1995.

[Rickel99] J. Rickel and W. L. Johnson. Virtual Humans for Team Training in Virtual Reality. *Proceedings of the 9th World Conference on AI in Education*, 1999.

[Shaw93] C. Shaw, M. Green, J. Liang and Y. Sun. Decoupled Simulation in Virtual Reality with the MR Toolkit. *Information Systems*. 11(3): 287-317, 1993.

[Staadt00] O. G. Staadt, A. Kunz, M. Meier and M. H. Gross. The blue-c: Integrating real humans into a networked immersive environment. *Proceedings of ACM Collaborative Virtual Environments*, 2000.

[Stanney98] K. M. Stanney, R. R. Mourant and R. S. Kennedy. Human Factors Issues in Virtual Environments: A Review of the Literature. *Presence*. 7(4): 327-352, 1998.

[Tramberend99]H. Tramberend. Avocado: A distributed virtual reality framework. *Proceedings of the IEEE Virtual Reality Conference*, 1999.

[Torguet00] P. Torguet, O. Balet, JP. Jessel , E. Gobetti, J. Duchon and E. Bouvier. CAVALCADE a system for collaborative virtual prototyping. *Journal of Design and Innovation Research - Special Virtual Prototyping*. 2(1), 2000.

[Watsen98] K. Watsen and M. Zyda. Bamboo - A Portable System for Dynamically Extensible, Real-time, Networked, Virtual Environments. *IEEE Virtual Reality Annual International Symposium*, 1998