A REAL OBJECT CONTROL SYSTEM FOR USE IN MANUFACTURING TELEPRESENCE ENVIRONMENTS.

Claiton de Oliveira
Department of Mechanical Engineering
School of Engineering of São Carlos
University of São Paulo, São Carlos, SP, Brazil
claiton@sc.usp.br

Suellen Galle
Department of Computation
State University of Londrina, Londrina, PR, Brazil
su_galle@yahoo.com.br

Jandira Guenka Palma
Department of Computation
State University of Londrina, Londrina, PR, Brazil
jgpalma@uel.br

Arthur José Vieira Porto
Department of Mechanical Engineering
School of Engineering of São Carlos
University of São Paulo, São Carlos, SP, Brazil
ajvporto@sc.usp.br

Abstract. *This paper presents a real object control system for use in telepresence systems, which is composed by a Real Object Class Library (ROCL) and a Real Object Control Software (ROCS). Through the real object control system, the object functioning simulation within a virtual environment at the same time as a real environment, makes possible the visualization of the events in a multiuser environment, in such a way that occurred interferences in real objects are reflected in virtual objects and vice versa. The creation of a virtual environment and a prototype control for use in telepresence environments must obey to criteria of size scale, space, speed and time for the results to be as accurate as it possible. The class representation of the real and virtual objects into the object-oriented paradigm results in clarity of the control conception of each object, simplifying the communication between the designer and the simulation system as well as the system updating through the easiness of new object addition. The system is a module of a telepresence environment authoring system which intends to facilitate the interface of virtual reality in manufacturing telepresence environments. The ROCL was tested and evaluated by using the ROCS to control some real prototypes.*

Keywords: *manufacturing environments, telepresence, virtual environments, real object control system*

1. INTRODUCTION

The telepresence systems consist of a refinement of the teleoperation: as the latter is the action of using a machine - teleoperator - with capacity to act at a distance under human control (Gevarter, 1984), the former makes possible, not only the operation but also the sensation of being in a environment distant from the operator (Gibilisco, 1994).

The virtual reality systems allow real world simulations in an environment or virtual world, being this the name given to the digital world created from computer graphic techniques. Once it can interact and explore this world through input and output devices, it becomes a virtual environment (Vince, 1995).

The application of the virtual reality technology and the implementation of synthetic environments allow the approach of geographically dispersed people through the data communication, sounds and images. It promotes the integration of its computational resources through data and application interoperability, inserted in a context similar to real (Palma, 2001).

The implementation of Virtual Environments (VE) demands computing technical knowledge, what makes difficult its creation and manipulation by users of other areas, that use Virtual Reality (VR) in their knowledge domain. Thus, this paper describes a real object control system for use in telepresence systems, which is composed by a Real Object Class Library (ROCL) and a Real Object Control Software (ROCS).

Through the real object control system, the object functioning simulation within a virtual environment at the same time as a real environment, makes possible the visualization of the events in a multiuser environment, in such a way that occurred interferences in real objects are reflected in virtual objects and vice versa.

The creation of a virtual environment and a prototype control for use in telepresence environments must obey to criteria

of size scale, space, speed and time for the results to be as accurate as it possible. The class representation of the real and virtual objects into the object-oriented paradigm results in clarity of the control conception of each object, simplifying the communication between the designer and the simulation system as well as the system updating through the easiness of new object addition.

The system is a module of a telepresence environment authoring system which intends to facilitate the interface of virtual reality in manufacturing telepresence environments. The ROCL was tested and evaluated by using the ROCS to control some real prototypes.

The remainder of this paper is divided as follows: section 2 describes the discrete event simulation with interface in virtual reality. The prototypes used in the system are described in section 3. Section 4 describes the real object class library. The real object control is describe in section 5 and finally section 6 presents the conclusions.

2. DISCRETE EVENT SIMULATION WITH INTERFACE IN VIRTUAL REALITY

This section brings a brief description about the main concepts related to discrete event system simulation through the use of virtual reality.

2.1 Discrete event simulation

Discrete event systems are systems in which state changes, or events, occur at discrete instants of time (Schriber, 1999). Discrete event system simulation are focused on events that occur in the real environment and on the development of equations that describe the conditions that induce the occurrence of these events. In this context, the simulation consists of forecasting and accomplishing changes in the state of the system, by means of the succession of events in the environment. From this, it is possible to obtain some conclusions and present recommendations about the modeled system, indicating solutions for the problems in study.

2.2 Petri net modeling tools

In order to simulate some process or system, it is necessary to construct a mathematical model, either symbolic or schematical that describes the system behavior. This model must take in account all the relevant aspects of the problem in analysis, even though in most cases is necessary to make a simplification of the problem, to make feasible the specification of a model which describes it. There are a lot of modeling tools, with diverse characteristics: deterministic, stochastic, dynamic, statics symbolic and mathematical among others.

One of the main formal modeling tools is known as Petri Nets (PN) (Murata, 1989). The theory of Petri Nets was introduced in 1926 by Carl Adam Petri, and consists of a graphical mathematical tool extremely effective for modeling and analysis of discrete event systems. A Petri net is a bipartite directed graph and with weighted arcs, composed primary of two structural elements: places that represent conditions and transitions that represent events. The Petri net elements can be seen in the Fig. 1.

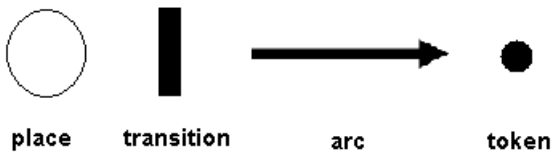

Figure 1. Basic Petri net Elements

A transition has a certain number of input and output places which represent respectively, the preconditions and postconditions of the events. A transition is enabled when the input places have sufficient tokens, according to its weight, which indicates that the necessary data or resources for that transition are available.

The Fig. 2 presents an example of Petri net modeling of an automated guided-vehicle system (AGV) used in manufacture.

The metamodel in which this work is based uses Petri net as modeling tool, once the PN concept attend to the necessary requirements for modeling discrete event systems (Murata, 1989).

2.3 Petri nets and virtual environments

The Petri nets has been successfully used to model, control and analyze event discrete systems, that are characterized by concurrence, parallelism, deadlocks and conflicts, which are also characteristics of virtual environments.

The dynamic of a virtual environment and its participant objects can be managed by a control system as Petri nets, responsible for integrating each virtual object to the virtual environment and controlling all the events of the environment.

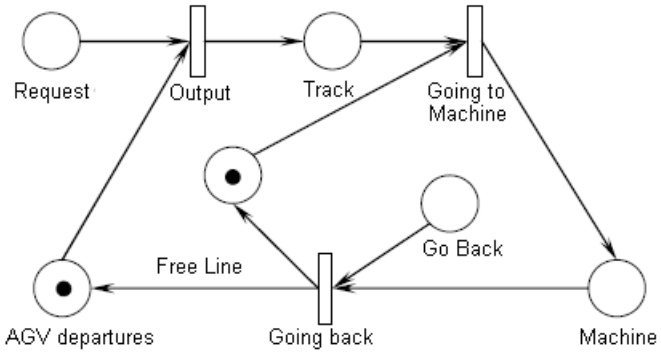

Figure 2. AGV Petri net modeling example

The integration of the Petri nets with the virtual environment consists of linking the places of the PN with the values of the attributes of activation from the VE and the transitions of the PN with the methods from the VE, creating a connection table. This connection table allows that all the objects instantiated in the VE can have its on attributes and methods associated to the elements of the PN, as shown in Fig. 3.

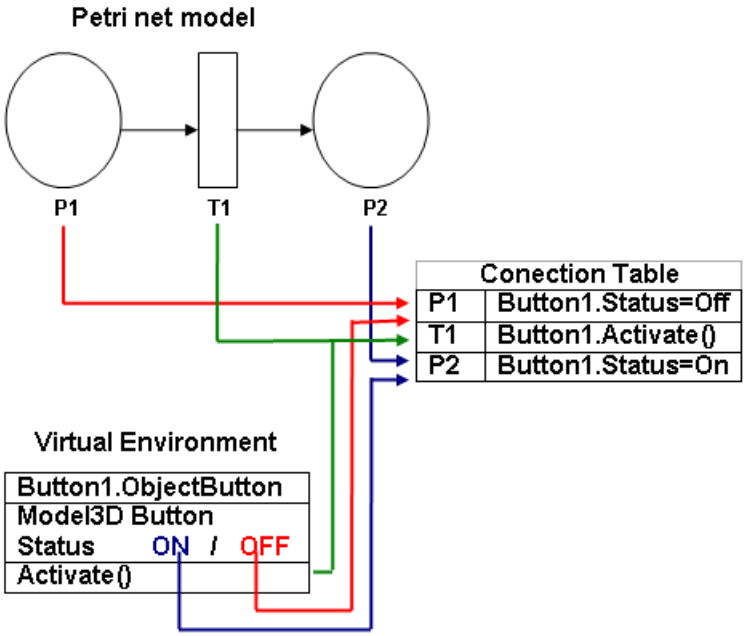

Figure 3. Link between Petri net and object

Thus, this connection table model is used to identify each element of the objects that can be activated or updated during run-time.

2.4 Use of virtual reality in simulation

By grouping some virtual reality definitions, it can be said that it is an advanced interface technique, where the user can perform immersion, navigation and interaction in a three-dimensional synthetic environment generated by computer, using multisensorial channels (vision, hearing, tact, etc.) (Earnshaw, 1995).

In the simulation area, virtual reality was introduced in the animation module, which is the subsystem of the simulation software responsible for establishing connection among the diverse components of the model and executing the simulation. With the VR use, the user can interact with the components of the system during his “virtual functioning”, as well as, immerge into the modeled environment, giving more realism and allowing a richer exploration of the system.

Traditionally, in order to perform discrete event simulation there are a few steps to be followed which are:

1. to prepare the input variables;
2. to select the parameters;

3. to execute the simulation;

4. to revise the results after the execution.

There some commercial systems that have been developing such experiments as the "Virtual Proving Ground" (NATC, 2007). This technology relies on modeling and simulation to create realistic testing environments, and can aid the user in the structural design of vehicle assemblies. Once the system can simulate the force and displacement inputs of the ground surface model to the vehicle system model, predictions of vehicle response and durability with widely varying ground surfaces and operational modes can be useful to the user before and during the initial design phase and in the design modification process.

The main problem of this type of system is that they are complex, which means they possess a high number of interrelated variables, what makes difficult its exploratory analysis. The analysis process requires several cycles to reach reliable information that illustrate aspects of interest in the system. A new approach consists of allying traditional computational concepts with virtual environments interactivity to guide the simulation, allowing verifications in run time and instantaneous responses of the system for obtaining the results (Kesavadas, 2000).

3. PROTOTYPES USED IN THE SYSTEM

The creation of a virtual environment and a prototype control for use in telepresence environments must obey to criteria of size scale, space, speed and time for the results to be as accurate as it possible. The class representation of the real and virtual objects into the object-oriented paradigm results in clarity of the control conception of each object, simplifying the communication between the designer and the simulation system as well as the system updating through the easiness of new object addition.

As mentioned previously, the Real Object Class Libray (ROCL) is a module of a telepresence environment authoring system which intends to facilitate the interface of virtual reality in manufacturing telepresence environments. This way, in order to test and evaluate this authoring system it was used the ROCS plus a Real Object Control Software (ROCS) to control some real prototypes. These prototypes are: a derrick, 2 buttons and an apron feeder.

The derrick prototype, which can be seen in the Fig. 4, is connected to the parallel port of the computer and fed from a 12-volt power supply. It is composed by two step motors:

- Motor 1: responsible for horizontally turning the arm of the derrick to the right or the left. To each step of the motor 1, the derrick arm turns 1,5 degree;
- Motor 2: responsible for raising or lowering the magnet at a 1,67 cm/s speed. Moreover, it is also possible to turn on or off the magnetization of the magnet.

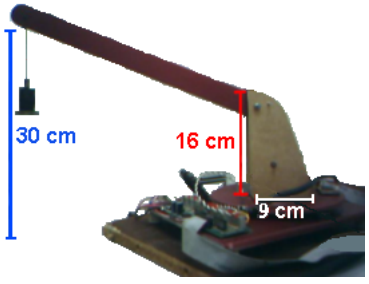

Figure 4. The derrick prototype

The prototype "Button" is a touch sensor that receives action from the "real world" and sends to the parallel port. When it is pushed it transmits 5 volts to the interface, indicating its condition (on or off). In this project two buttons were used: button 1 and button 2, which can be seen in Fig. 5. The buttons can be pushed separately (one of each time) or simultaneously (the two at the same time).

Figure 5. The button prototype

The prototype “Apron feeder”, which can be seen in the Fig. 6, is connected to the parallel port of the computer, and it is fed from a 12-volt power supply. It possesses DC motor capable of turning its axle and to put into motion the apron feeder. In the same way that it occurs with the derrick the apron feeder actuation is made through the sending of a hexadecimal number to the parallel port.

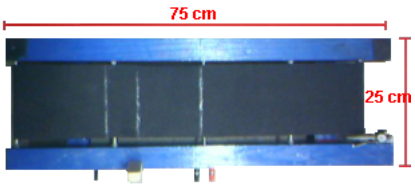

Figure 6. The apron feeder prototype

4. REAL OBJECT CLASS LIBRARY

The Real Object Class Library (ROCL) has the purpose of providing control methods of real prototypes with will be used in the simulation of a considered system. These real objects are motorized prototypes that compose a real environment (physical) for simulation, as well as virtual objects modeled in 2D or 3D will compose a virtual environment.

The two environments, virtual and real, interact in such a way that all the actions executed in an environment be executed identically and at the same time in another environment, that is, the occurred interferences in real objects are reflected in virtual objects and vice versa.

The real and virtual object classes were designed under the object oriented programming paradigm (OOP), and are organized in a way that a parent class called “Object” possesses as children each one of the objects of the library. For each object, there are two classes that are the implementation of these objects in the virtual and real scope, as it can be observed in the Fig. 7. That is, the super-class “Object”, which through pointers to the functions and attributes members of the implemented objects, provides a single interface for all the other objects, real or virtual.

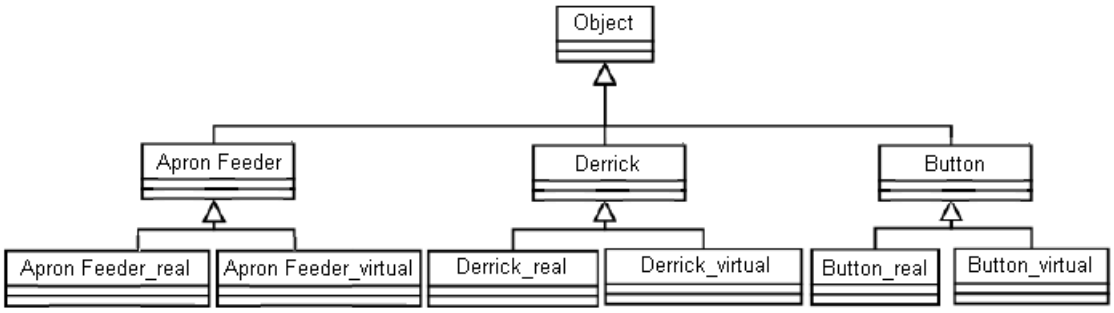

Figure 7. Structure diagram of the real and virtual classes

As shown in the Fig. 7, the class correspondent to the object type “Derrick” is an abstract class, that is, a class which is not used to create objects and only exists to serve as base for its derived subclasses. This way, each class possesses the virtual methods that are only implemented in its respective subclasses. So, each method can be implemented with different commands, in accordance with the distinct necessities of the real and virtual subclass. This standard is followed by all the three classes presented in the Fig. 7: Apron Feeder, Derrick and Button.

With this type of organization of object classes, the addition of prototypes to the system is obtained with the elaboration of the new specific class for the object in question. The methods of each subclass (real and virtual) will follow the requirements dictated by the virtual methods of its parent class and the controls related to the super-class “Object” will be inherited. Thus, the system update is facilitated and there is an increase of the simulation possibilities with the addition of new objects to the library.

5. REAL OBJECT CONTROL SOFTWARE

The Real Object Control software (ROCS) was elaborated for use and test of the Real Object Class Library (ROCL). The composition of the software and the library results in a Real Object Control System.

The software (ROCS) offers an option menu for use of the functionalities of each three types of prototypes (Derrick, Button and Apron Feeder). It is used essentially for tests of the library (ROCL) and the prototypes.

The system executes the motorized prototype control for discrete event simulation in real environments so that it can provide a visualization of movements, allowing error detection and possibilities of improvements, before an implantation in a definitive environment of use.

The use of the structure of classes where the real and virtual subclasses implement the virtual methods of its respective super-class, directly provided the integration of the real object classes to the virtual object classes. And thus it facilitated the joining of other parts of the major project and the performance of the tests of the virtual and real environment together.

As it was expected, the two environments, virtual and real, interacted in such a way that all the actions executed in one environment could be executed identically and at the same time in the other environment. Each occurred interference in the real objects were reflected in the virtual objects and vice versa.

6. CONCLUSIONS

The complexity in the real and virtual environment construction for simulation, inhibits the application of the VR technology and telepresence. This proposal facilitates the use of real prototypes and the RV interface in the discrete event simulation, allowing both the simulation and the user interaction.

The control system of motorized real prototypes for discrete event simulation in real environments, provides the previous visualization of the movements and allows error detection or visualization of possibilities of improvements, even before an implantation in a definitive environment, where it will not be able to occur failures.

The creation of the real environment and the prototypes for use in telepresence environments must obey to criteria of size scale, space, speed and time so that the results be as most accurate as possible. This way, the simulation performed in the environment with the prototypes will produce a situation identical the one that will occur in the real environment of use.

The representation of the real and virtual classes under the oriented object paradigm results in a clear conception of the control of each object. This simplifies the communication between the designer and the update and simulation system through the easiness of new object addition.

7. REFERENCES

Earnshaw, R. A. et al, 1995, “Virtual Reality applications”, London, Academic Press, 328p.

Gevarter, W. B., 1984, “Artificial Intelligence, Expert Systems, Computer Vision and Natural Language Processing”, New Jersey: Noyes Publications, 226p.

Gibilisco, S., 1994, “The McGraw-Hill Illustrated Encyclopedia of Robotics and Artificial Intelligence”, New York: TAB Books, 420p.

Kesavadas, T., Sudhir, A., 2000, “Computational Steering in Simulation of Manufacturing Systems”, In: Proceedings IEEE International Conference on Robotics and Automation, Vol.3, p. 2654-2658.

Murata, T., 1989, “Petri Nets: Properties, Analysis and Applications”, Proceedings of IEEE, Vol. 77, No. 4, pp. 541-574.

NATC Nevada Automotive Test Center - Virtual Proving Ground, September 1st 2007, <http://www.natc-ht.com/Virtual_Proving_Ground.htm>.

Palma, J., 2001, “Metamodelo para a Modelagem e Simulação de Sistemas a Eventos Discretos, Baseado em Redes de Petri e Realidade Virtual. Uma Aplicação em Sistema de Manufatura”, Tese (Doutorado), EESC - Escola de Engenharia de São Carlos.

Schriber, T.J., Brunner, D.T, 1999, “Inside Discrete-Event Simulation Software: How It Works and Why It Matters”, Winter Simulation Conference Proceedings, Vol. 1, pp. 72-80.

Vince, J., 1995, “Virtual reality systems”, Cambridge, Addison-Wesley, Reading, MA, USA.

8. RESPONSIBILITY NOTICE

The authors are the only responsible for the printed material included in this paper.