中華民國八十四年全國計算機會議論文集
Proceedings of National Computer Symposium 1995

PETRI NETS FOR VIRTUAL REALITY SYSTEM: A RULE-BASED APPROACH

Ming-Shing Shieh (謝明興)[1,2] and Din-Chang Tseng (曾定章)[2]

[1]Department of Information Management
Tamsui Oxford University College, Tamsui, Taiwan

[2]Institute of Computer Science and Information Engineering
National Central University, Chung-Li, Taiwan

摘 要

虛擬實境是一個電腦模擬出來的三度空間動態環境，這個環境能夠讓人們想像著沈浸在其中，並且利用全身外在的器官藉著特別的介面設備去與整個環境中的事物相互感受與處理。對於構建虛擬實境物件階層間的關係、周邊介面相關訊息的傳遞、資源持有、甚至在分散式虛擬實境系統下畫面週期性的同步等這些事實、規則、命題的描述極為複雜，因此需要彈性的知識表示結構及執行策略，以因應周邊介面與系統執行核心的互動，而能簡化建構系統的複雜度。本文討論了Petri net的定義、描述、模式，同時提出應用Petri net於虛擬實境系統中處理互斥事件資源持有，及週期性的資源同步控制；並以產出規則建構虛擬實境系統物件關係，用以協助建構Petri net，有助於整個複雜系統全面性的觀察及維護。

關鍵詞：虛擬實境，產出規則，資源持有互斥，週期同步。

Abstract

Virtual reality (VR) is a three-dimensional dynamic environment which is generated and simulated by computer equipment. These environments involve real-time simulations and interactions through multiple sensorial channels that one can be deeply full-body immersion and interaction through high-end user interfaces. To generate such a system, the designer should organize the relationships of objects hierarchy concerning VR system construction. The intercommunication between program and sensory interface, resources management, and even the synchronization strategy all should be taken into consideration when constructing a VR system. These presenting methods such as facts, rules, propositions and so on require more adaptive complexity demand and more flexible representation and execution strategies. Petri nets are graphical and mathematical tools. Models of which providing features for the analysis of behavior properties and performance evaluation, as well as for systematic construction of discrete-event simulators and controllers. So, using Petri nets give the possibility for formally analyzing the behaviors of mutual exclusion of shared resources and periodic synchronization of frame control. In this paper, we give a guidance of Petri nets to model the mutual-exclusive and periodic synchronization function used in a VR system and then a set of production rules is proposed to implement the objects' relations which are composed a VR system. Besides, apply the components such as reachability set to construct Petri nets can help maintain the complex system.

Keywords: Virtual reality, Petri net, production rules, mutual exclusion, periodic synchronization.

1. Introduction

The increased complexity in VR systems, such as software modules, communication among objects, feedback control, etc., causes numbers of problems to inflict the system developers. In the planning stage, when one meets with such complexity of the environment that is composed of special peripheral devices and software, who must solve the problems such as abstract object type, programming methods, communication of objects, interactive modules, etc. In order to improve the immersion property in a

virtual environment, a better design method for VR system is necessary. Here special attention is paid to the design and management of VR systems using Petri nets.

Petri nets, as graphical and mathematical tools, provide variant functions for modeling, formally analyzing and controlling the discrete events in a virtual environment [9]. Petri nets as graphical tools provide a power communication medium between a system and its user interface. Petri nets instead of ambiguous textual description and mathematical notation which are difficult to be understood by the developers can be used to represent the complex interaction models.

As a mathematical tool, a Petri net can be described by a set of linear algebraic equations, or other mathematical models projecting the behavior of the interaction module. This function gives a possibility for formal communication among objects in time intervals or events. Petri nets allow one to perform a formal check of the properties related to the behavior of the underlying system, e.g., precedence relations among events, concurrent operations, appropriate synchronization, repetitive activities and mutual exclusion of shared resources. The major advantages of using Petri-net models are that using such models to analyze behavior properties, performance evaluation, and systematic construction of discrete-event simulators and controllers.

The remains of the paper is described as follows. The Petri-net description, definition, logical representation, models of transitions and places, firing strategies, Boolean analysis and reachability are given in Section 2. Knowledge representation including matrix representation and production rules are discussed in Section 3. Properties of Petri nets for virtual reality, including activities in a virtual environment, environment modeling, mutual exclusion, periodic synchronous layout module in virtual reality, production rule and grammar for parsing production rules in BNF are explored in Section 4. Conclusions are given in Section 5.

2. Petri-net Models

A Petri net is described as a bipartite directed graph which is composed of three types of components: places, transitions and directed arcs. The directed arcs connect places and transitions. Pictorially, places are expressed as circles and transitions are expressed as bars; one example of a Petri net is shown in Fig. 1. In the condensed form, a Petri net can be represented by a transition with a few input and output places. A place is an input place if it has a directed arc connecting to the transition; a place is an output place if it has a directed arc connecting from the transition.

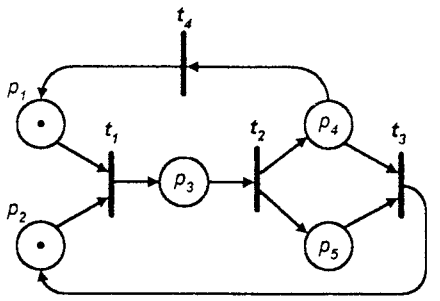

Fig. 1. Representation of a Petri net.

Petri nets are usually used to represent various modeled systems. To model a VR system using Petri nets, places are used to represent objects and causal preconditions, such as object's behavior or the availability of resources, and transitions are used to represent events or potential utilization which lead to effects as an output place, such as the release of the resources.

2.1 Definition of Petri nets

The formal definition of a Petri net is given as follows.

Definition 1: a Petri net is formally defined as a 6-tuple [2-4, 12-13, 18]

$$N_P = (P, T, Q, I, O, M_0),$$

where

1. $P = \{p_1, p_2, ..., p_n\}$ is a finite set of places, and $n \geq 0$;
2. $T = \{t_1, t_2, ..., t_m\}$ is a finite set of transitions, and $m \geq 0$;
3. $Q = \{q_1, q_2, ..., q_k\}$ is a finite set of state tokens, and $k \geq 0$;
4. $I \subseteq (P \times T) \rightarrow N$ is an input function set that defines a set of directed arcs from places to transitions, where N is a set of non-negative integers, and each p_i is an input place of t_j if $(p_i, t_j) \in I$;

5. $O \subseteq (T \times P) \rightarrow N$ is an output function or weight function set that defines a set of directed arcs from transitions to places, where N is a set of non-negative integers, and each p_i is an output place of t_j if $(t_j, p_i) \in O$; and
6. M_0: $P \rightarrow N$ is an initial marking.

An input and an output sets are defined for a transition. A transition which has no input place is called a source transition; a transition which has no output place is called a sink transition. For representing a dynamic system and the behavior of a modeled system, each place can potentially hold either none or a number of tokens. The distribution of tokens over places is called a marking of the Petri net. A token is expressed by a small solid dots as shown in Fig. 1. In general, Petri nets have a dynamic life determined by the tokens inside of a place that activate the place in which they appear [7].

2.2 Models of transitions and places

The presence or absence of a token indicates whether a condition associated with a place. On the other hand, a transition may fire when every input places of the transition contains at least one token. A transition is accredited with having enabled [13] when it fires; otherwise, it is disabled. A Petri net containing tokens is called a marked Petri net. A marking of a Petri net with n places is represented by an $(n \times 1)$ vector M; elements of the vector, denoted as $M(p)$, are non-negative integers representing the number of tokens in the related places [3]. Since a place represents the availability of resources, the number of tokens in the place represents the number of available resources in the place [8, 16, 18]. For instance of the Petri-net model shown in Fig. 1, $M = (1, 1, 0, 0, 0)^T$. The transition t_1 is fired to result in removing tokens from its input places and adding a token to its output place as shown in Fig. 2. The changing distribution of tokens on places represents the occurrence of events in a dynamic modeled system. A scheme of dynamic behavior has been introduced in [11, 18] to deal with the fire mechanism of a transition.

Firing rule: A transition is enabled if its every input place contains tokens and the number of tokens in each input place is equal to or greater than the weight of the directed arc connecting from the transition (i.e., the post-conditions in the output places of the transition). An enabled transition may remove tokens from every input place of the transition; the number of removed tokens equal to the weight of the directed arc connecting the transition.

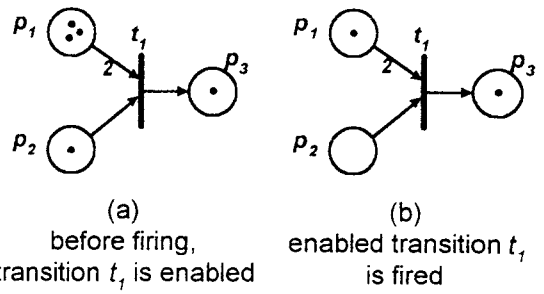

Fig. 2. An example of fired transition in a Petri net.

On the ground of above, a state or marking in a Petri net is changed according to the following:

1) A transition t_i is said to be *enabled* if each input place p_k of t_i is marked with at least $O(p_k, t_i)$ tokens, where $O(p_k, t_i)$ is the weight of the arc from p_k to t_i.
2) An enabled transition may or may not fire depending on whether or not the event actually occurred.
3) A firing of an enabled transition t_i removes $O(p_k, t_i)$ tokens from each input place p_k of t_i, and adds $O(p_k, t_i)$ tokens to each output place p_k of t_i, where $O(p_k, t_i)$ is the weight of the arc from t_i to p_k.

The firing rule illustrated in Fig. 2, transition t_1 is enabled since every input place contains tokens, and $I(p_1, t_1) = 3$, $I(p_2, t_1) = 1$. The enabled transition is fired to remove tokens from every input place, and deposit a token in the output place p_3 and $O(t_1, p_3) = 1$.

Petri nets can be used to model concurrent processes [7]. Functions of each subsystem are semi-independently in the form of asynchronism. Petri nets, which modeled subsystems may send/receive tokens to/from each others. A periodic synchronization may be exceeded among subsystems because one of the subsystems may reach a waiting-state for a token from another subsystem.

3. Knowledge Representation of Petri-net Models

Knowledge representation based on Petri nets is production rule [2, 8, 12]. We denote the places by $C(1)$, $C(2)$,..., $C(K)$, to represent a set of K conditions in a rule-based system. The truths of the conditions, i.e., the tokens in place, are designates by $T(1)$, $T(2)$,...,$T(K)$, which make up a truth state T for the conditions. Each component of the truth state T will be either 1, denoted as enabling, or 0, denoted as disabling.

Postulate: In a Petri net, an activated place passes copies of its truth token along all arcs that depart from that place. An activated transition passes copies of its truth token along all arcs that depart from that transition.

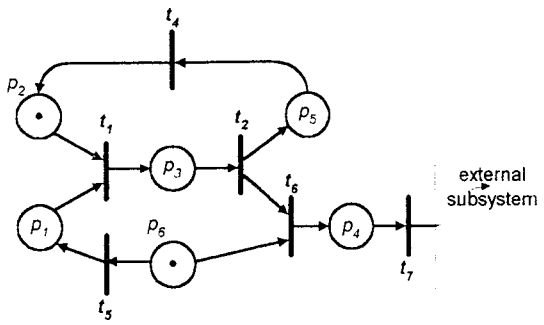

Fig. 3. Example of simple rule-based Petri net.

R1: $(p_1 \; AND \; p_2) \Rightarrow p_3$.
R2: $p_3 \Rightarrow p_5$.
R3: $(p_3 \; AND \; p_6) \Rightarrow p_4$.
R4: $p_5 \Rightarrow p_2$.
R5: $p_6 \Rightarrow p_1$.
R6: $p_4 \Rightarrow P'$. (place in external subsystem)

Fig. 3 shows a simple rule-based system with the following rules:

Suppose now that the truths of p_2 and p_6 are given, i.e., that place p_2 and p_6 are given enabling tokens as shown in Fig. 3. Then transition t_5 is fired, as is transition t_3. Thus p_1 becomes true. Next, the truth of p_1 and p_2 cause transition t_1 to fire so that p_3 becomes true. Consequently, transition t_6 fires, this causes p_4 becomes true and sends a truth token copy to P' in an external subsystem.

4. Properties of Petri Nets for VR

4.1 Mutual exclusion

To run a VR system, the first thing is to read task-dependent user input and then accesses task-level-dependent databases to process corresponding world instances or frames. It is hard to predict all participator's actions and to store all corresponding frames in buffers in advance; therefore, the objects in a virtual environment should be created, modified, and deleted in real time. Human-factors studies indicate that a 30-frames/second refresh rate is needed to display smooth simulations (the time between user action and virtual system feedback) [1, 5]. According to the graphical performance such as frame rate, shading mode, and scene complexity of a virtual environment, the distribution computational load is required owing to the real-time characteristic of a VR simulation. An approach to load a distribution is to divide tasks into several workstations communicating over a LAN. Network distribution has the benefit in distributed simulation without dedicated VR machine. Another benefit is the feasibility of remote access and participation. The facility makes possible multi-user VR simulations. A problem related to such multi-user network-distributed systems is model synchronization, resource of mutual exclusion, and concurrent activities in concert, etc.

Each participant can modify various aspects in a virtual environment at any time. These changes have to be communicated quickly to the other participants to ensure the synchronization and mutual exclusion. Pimentel *et al.* [14] and McCarty *et al.* [10] gave the opinions of shared elements in a virtual environment. These primary elements were remote wand location, orientation, state, object selection state, creation/deletion of building elements, selection of different texture maps, and so on. We here think of a virtual environment as a database accessible in real time. Constraints imposed by data complexity, message routing and mutual exclusion are inherent in each systems.

Petri nets can be used to model properties of a distributed VR system such as mutual exclusion, synchronization, behavior of inheritance, concurrent activities, etc. In modeling VR, using the conditions and events; places represent conditions,

and transitions represent events. A transition has a certain number of input and output places representing the pre-condition and post-condition of the event, respectively. The presence of a token in a place is interpreted as holding the truth of the condition associated with the place [11, 13, 18]. In another interpretation, k tokens are put in a place to indicate that k data items or resources are available. Some typical interpretations of transitions and their input places and output places in a virtual reality environment are shown in Table 1.

Table 1. Some Typical Interpretations of Transitions and Places in VR Environment

Input places	*Transitions*	*Output places*
input signals	signal processor	coordinates
input data	computation	output data
resource needed	task	resource released
preconditions	event	postconditions
conditions	clause in logic	conclusions

In a short-winded VR system which consists of at least two users decorating a room in an immersive virtual environment. Stereo images are sent to every participant's head-mounted displays (HMD). Head position is tracked by a 3D trackers, while the decorative objects are picked and scratched on a wall using a sensing glove. The shared virtual environment is simulated by running a duplicate simulation on every connected machine through a local area network. Any change in one station must be sent to all other stations to maintain the consistence of the system. In this short-winded system, the fancy ornamented goods are mutual exclusion.

In a networking VR system, each object has a sole name, some of the objects "inhabit" on the local machine and others on the remote one. Since networking, simulations are executed asychronously, and coordination messages, which control remote objects, are sent during each simulation cycle. For keeping simulation synchronized, communication may be keep low, and each message is made as compact as possible and is sent only when objects changed position.

The short-winded VR system is represented by a Petri-net model as shown in Fig. 4 and Table 2. In this system, two remote gloves G_1 and G_2 perform pick-and-affix operations to access adornments from a pursuing division at any time and hang the adornments on a wall in a virtual house. In order to avoid collision, it assumed that only one remote glove can access the workspace at a time. Besides, it is assumed that the pursuing division contains a buffer with a limited space for adornments.

In the Petri-net model, places p_1, p_2, p_3, p_4 and transitions t_1, t_2, t_3, t_4 simulate behavior and activities of remote glove G_1, while places p_5, p_6, p_7, p_8 and transitions t_5, t_6, t_7, t_8 simulate behavior and activities of remote glove G_2. Transitions t_1 and t_5 indicate concurrent activities of G_1 and G_2 reading sensor data. Either these two transitions are fired in parallel or not. Transitions t_2 and t_6 perform temporarily checking the concurrence of object's coordinates and update objects with data represented by tokens in place p_{12}; the tokens represent the tasks performed by gloves G_1 and G_2 outside the pursuing division.

The access to the pursuing division requires synchronization of the activities of the remote gloves refraining from collision. The synchronization is achieved by a Petri sub-net involving places p_9, p_{10}, p_{11} and transitions t_3, t_4, t_7, t_8. If t_3 is enable, firing t_3 brings the effect to act places p_4 and p_{11}; that is, glove G_1 performs pick-and-affix operations accessing adornments from a pursuing division, and this action disables t_7 simultaneously. Moreover, it is assumptive that the pursuing division has a limited buffer. Consequently, if place p_9 is empty, transition t_3 is disabled. It means that only one glove can access adornments from a pursuing division at a time.

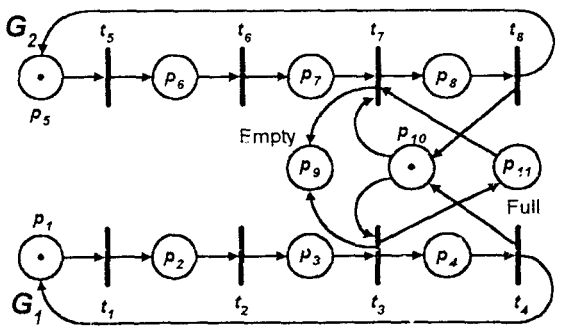

Fig. 4. Petri-net model for a distributed short-winded VR system.

Table 2. Explication of Places and Transitions in the Petri-net Model for the Distributed Short-winded VR System

Place	Explication
P_1	Glove G_1 reads sensor data
P_5	Glove G_2 reads sensor data
P_2	Glove G_1 updates objects with data
P_6	Glove G_2 updates objects with data
P_{12}	Glove G_1 or G_2 performs tasks outside the pursuing division
P_3	Glove G_1 waits for the access to the pursuing division
(P_7)	Glove G_2 waits for the access to the pursuing division
P_4	Glove G_1 (G_2) performs pick-and-affix operations
(P_8)	accessing adornments from a pursuing division
P_9	adornment is empty in a buffer
P_{11}	adornment is ready to access in a buffer
P_{10}	mutual exclusion of accessing adornments from a pursuing division
Transition	Explication
t_1 (t_5)	Glove G_1 (G_2) requests updating objects with data
t_2 (t_6)	Glove G_1 (G_2) requests accessing to the pursuing division
t_3 (t_7)	Glove G_1 (G_2) accesses adornments from a pursuing division
t_4 (t_8)	Glove G_1 (G_2) leaves the pursuing division

Now, we are going to approach the projection of a VR knowledge-control representation onto a Petri net. We first start by identifying the places of the Petri nets with the propositions of the VR control knowledge by means of the function α

$$\alpha: \quad P \Rightarrow PR.$$

$$p_k: \quad \alpha(p_k) = P_k \quad k \in N.$$

where $PR = \{P_k\}$ is the set of propositions representing VR control knowledge. According to the definition 1, we define the input function over set T

$$I: T \Rightarrow \wp(P).$$

and output function over set T

$$O: T \Rightarrow \wp(P).$$

which associate to each transition the set of places which constitute its input and output, respectively.

Control knowledge representation of VR for the Petri-net formalism is shown in Fig. 5. Immediate reachability set and reachability set for each place p_k can be used for coverability analysis and net construction.

4.2 Periodic synchronization

R1: IF s_1' *is* S_1	*R8: IF* g_7' *is* G_7	α: $P \Rightarrow PR$	I: $T \Rightarrow \wp(P)$	O: $T \Rightarrow \wp(P)$
THEN b_1' *is* B_1	*OR* ((p_{22}' *is* P_{22})	$s_1 \Rightarrow s_1'$ *is* S_1	$t_1 \Rightarrow \{s_1\}$	$t_1 \Rightarrow \{b_1, g_1, g_2\}$
AND g_1' *is* G_1	*AND* (g_8' *is* G_8)	$s_2 \Rightarrow s_2'$ *is* S_2	$t_2 \Rightarrow \{b_1\}$	$t_2 \Rightarrow \{g_3\}$
AND g_2' *is* G_2.	*AND* (p_2' *is* P_2))	$s_3 \Rightarrow s_3'$ *is* S_3	$t_3 \Rightarrow \{g_2\}$	$t_3 \Rightarrow \{p_1\}$
R2: IF b_1' *is* B_1	*THEN* p_3' *is* P_3.	$b_1 \Rightarrow b_1'$ *is* B_1	$t_4 \Rightarrow \{g_3\}$	$t_4 \Rightarrow \{p_1\}$
THEN g_3' *is* G_3.		$g_1 \Rightarrow g_1'$ *is* G_1	$t_5 \Rightarrow \{g_1\}$	$t_5 \Rightarrow \{p_1\}$

Fig. 5. Example of the application of the Petri-net formalism to a generic KB.

For modeling VR activities, places described in Section 3 can be functionally classified into several categories as follows:

1) *Resource places* are used to model the activity in VR system such as robots, machines, and carrier. The kinds of these places offered depend on the application categories of VR. If a resource is marked, the corresponding resource is available.

2) *Operation places* are employed to indicate the status of resource usage. A marked operation place indicates that a specific operation is being performed. These places are often associated with certain units of operation time.

3) *Induction places* indicate the signals obtained from sensors that are sent to or are received from the other submodels to express some close-at-hand events. The source of the signals may be yielded from glove, trackers, 3D mouse, or some other feedback equipment.

4) *Intermediate places* are used to model the process flow of each action. If an *intermediate place* is marked, it represents that the last action is accomplished and is ready for the next action.

Now, we want to construct the Periodic Synchronous Layout Module for VR execution environment as described in Fig. 5. For clarity, the notations of places used in Fig. 6 are explained as follows:

1) **b***k*, $k \in N$, represents the ensemble, i.e. the unitary object, at a workstation. A token in the place **b***k* means the integral movement of **b***k* is enabled.

2) **s***k*, $k \in N$, represents the sensors at a workstation. If the place **s***k* is marked, it means a trigger event is occurred.

3) **g***k*, $k \in N$, represents the performance of graphical objects. If the place **g***k* is marked, it means graphical objects, **g***k*, are ready to perform some activity.

4) **p***k*, $k \in N$, represents the content of tasks. If the place **p***k* is marked, it means the task, **p***k*, has been accomplished.

5) **r***kl*, *k*, *l* $\in N$, with an arc connecting region *k* and region *l* in the layout directed graph, represents the path from region *k* to region *l*. If there is a token in the place **r***kl* means that an safe movement wants to move region *k* to region *l*.

6) ***cp*** represents the status of completing the movement along a segment path or accomplishing a task.

We construct the Periodic Synchronous Layout Module for VR system as shown in Fig. 6. It is remarkable that Periodic Synchronous Layout Module has already embedded a control rule into it so that any collision will be never occurred.

Grammar to translate Fig. 6 is shown in Table 3.

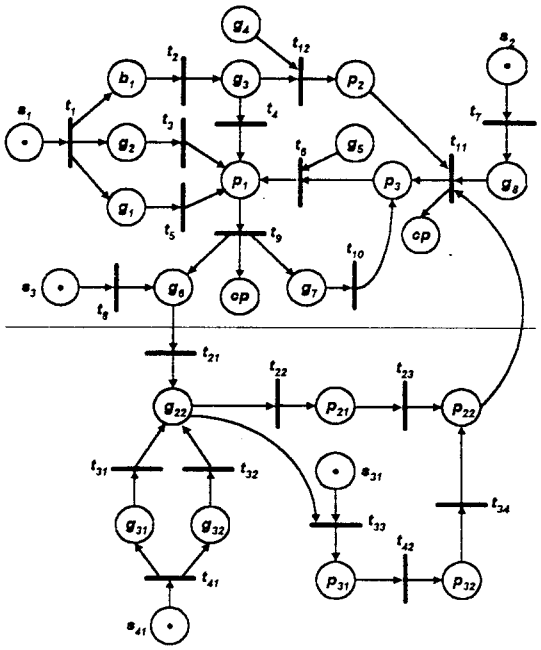

Fig. 6. Periodic Synchronous Layout Module.

Table 3. The Grammar for the VR Production Rules Expressed in ***BNF***

VR_control_knowledge_base ::=
control_rule | *VR_control_knowledge_base*
control_rule ::= 'R' *rule_no* ':' 'IF' *antecedent*
'THEN' *consequent* '.'
rule_no ::= *digit* *digit* | *rule_no*
antecedent ::= *proposition* | *proposition*
'AND' *antecedent*
consequent ::= *proposition* | *proposition*
'AND' *consequent*
proposition ::= *proposition_clause* |
proposition_clause 'OR' *antecedent*
proposition_clause ::= *attribute* *predicate* *value*
attribute ::= *strings* | *strings* *attribute*
predicate ::= 'is'
value ::= *strings* | *strings value*
digit ::= '0' .. '9'
strings ::= *(any string, up to* 30 *characters,*
surrounded by blanks)

5. Conclusions

In this paper, we proposed a generalized Petri-net model for modeling and controlling discrete events in a virtual environment. Petri nets are shown to have advantages to model a system which has the property of mutual exclusion and periodic synchronization. A transition is triggered only if the values in its input places/tokens match the conditions of a number of associated rules.

The need for real time operation in VR applications can be a drawback due to the large number of complex activities in a VR environment. This requires more structural complexity demand and more flexible representation and execution strategies. We have used the Petri-net formalism in order to obtain a formal structure that permits the definition of VR activities to help carry out inferences in complex situations. This process is based on a Petri-net approach to compose inference rules.

An important issue related to this study is the choice of Petri sub-net and rules for transitions. A planning for planning problem may be important to ensure the property of reversibility of the mutual sub-net during execution. The coordination and scheduling for processing VR activities should be incorporated into plans.

References

[1] Burdea, G. and P. Coiffet, *Virtual Reality Technology*, John Wiley & Sons, NY, 1994.

[2] Bugarin, A. J. and S. Barro, "Fuzzy reasoning supported by Petri nets," *IEEE Trans. Fuzzy Systems*, Vol.2, No.2, pp.135-150, 1994.

[3] Cao, T. and A. C. Sanderson, "A fuzzy Petri approach to reasoning about uncertainty in robotic systems," *in Proc. of 1992 IEEE Int. Conf. on Robotics and Automation*, Vol.1, pp.317-322, 1993.

[4] Chen, S. M., J. S. Ke, and J. F. chang, "Knowledge representation using fuzzy petri nets," *IEEE Trans. Knowledge and Data Engineering*, Vol.2, No.3, pp.311-319, 1990.

[5] Kalawsky, R. S., *The Science of Virtual Reality and Virtual Environments*, Addison-Wesley, reading, MA, 1993.

[6] Latta, J. N. and D. J. Oberg, "A conceptual virtual reality model," *IEEE Computer Graphics & Applications*, Vol.14, No.1, pp.23-29, 1994.

[7] Looney, C., "Expert control design with fuzzy rule matrices," *Int. J. Expert Syst.:Res. Appl.*, Vol.1, No.2, pp.159-168, 1988.

[8] Looney, C., "Fuzzy Petri nets for rule-based decisionmaking," *IEEE Trans. Systems, Man, and Cybernetics*, Vol.18, No.1, pp.178-183, 1988.

[9] Mccarragher, B. J., "Petri nets modelling for robotic assembly and trajectory planning," *IEEE Trans. Industrial Electronics*, Vol.41, No.6, pp.631-640, 1994.

[10] McCarty, W. D., S. Sheasby, P. Amburn, M. R. Stytz, and C. Switzer, "A virtual cockpit for a distributed interactive simulation", *IEEE Computer Graphics & Applications*, Vol.14, No.1, pp.49-54, 1994.

[11] Murata, T., "Petri nets: Properties, analysis, and applications," *IEEE Trans. Fuzzy Systems*, Vol.2, No.4, pp.295-301, 1994.

[12] Pedrycz, W. and F. Gomide, "A generalized fuzzy Petri net model," *in Proc. IEEE*, Vol.77, No.4, pp.541-580, 1989.

[13] Peterson, J. L., *Petri Nets, Theory and the Modelling of Systems*, Prentice Hall, Englewood Cliffs, NJ: Prentice Hall, 1981.

[14] Pimentel, K. and B. Blau, "Teaching your system to share," *IEEE Computer Graphics & Applications*, Vol.14, No.1, pp. 60-65, 1994.

[15] Sun, T. H., C. W. Cheng, and L. C. Fu, "A Petri net based approach to modeling and scheduling for an FMS and a case study," *IEEE Trans. Industrial Electronics*, Vol.41, No.6, pp.593-601, 1994.

[16] Tazaki, E. and K. Yoshida, "A fuzzy Petri net model for approximate reasoning and its application to medical diagnosis," *in Proc. of 1992 IEEE Int. Con. on Systems, Man, and Cybernetics*, Vol.1, pp.627-631, 1992.

[17] Venkatesh, K., M. Zhou, and R. J. Caudill, "Comparing ladder logic diagrams and Petri nets for sequence controller design through a discrete manufacturing system," *IEEE Trans. Industrial Electronics*, Vol.41, No.6, pp.611-619, 1994.

[18] Zurawski, R. and M. Zhou, "Petri nets and industrial applications: A tutorial," *IEEE Trans. Industrial Electronics*, Vol.41, No.6, pp.567-583, 1994.