Proceedings of the 10th IEEE International Workshop on Robot and Human Communication, pp. 50-55, 2001.

A New Integration Method of Constructing an Interactive Virtual Environment for the Collaboration between Virtual Agent and Real Human

Hirotake Ishii Keisuke Endou Kazumasa Sharyo
Daisuke Komaki Nobuyuki Ichiguchi Hidekazu Yoshikawa

Graduate School of Energy Science, Kyoto University
Gokasho, Uji, Kyoto, JAPAN 611-0011

Abstract

A new collaborative environment has been constructed recently, in which a human-shaped virtual agent can communicate with a real human through both verbal and nonverbal ways. The collaborative environment is very promising as the next generation communication environment, but there exists a problem the workload for the construction is too large for the practical use. In this study, the system for simulating the collaborative environment is divided into several subsystems according to its function, and for each subsystem a new construction method to reduce the construction workload has been developed individually. In this paper, newly developed construction methods are described such as an object-oriented method for constructing virtual spaces, an affordance-based body motion synthesizing method and a modeling method of the virtual agent's behavior using Petri-net.

1 Introduction

Virtual Reality technology can be applied for various kinds of industrial and commercial fields such as amusement, education, design support. Especially, a new collaborative environment has been constructed recently, in which a human-shaped virtual agent is located, and the virtual agent can communicate with a real human through both verbal and nonverbal ways. The virtual agent plays a role of the substitute of real training instructor, cooperative worker, secretary and so on.

For example, Jeff Rickel and his colleagues have developed a pedagogical agent called "Steve" that supports the learning process[1]. Steve agents can demonstrate skills to trainees, answer trainee questions, watch the trainees as they perform the tasks, and give advice if the trainees run into difficulties. Steve agents enable the trainee to learn complicated tasks even if the trainee is alone and a real instructor or the other trainee cannot participate in the training. Therefore, such kind of training environment is very promising as the next generation training environment, but there exists a problem the workload is too large to construct the training environment for the practical use of real field training.

Then, in this study, new construction methods have been proposed to reduce the workload for constructing such a next generation collaborative environment. Concretely, the system for simulating the collaborative environment is divided into several subsystems according to its function and for each subsystem a new construction method has been developed individually.

In this paper, new construction methods are described for synthesizing the virtual agent's body motion as 3 dimensional computer graphics, designing virtual spaces as collaborative environments and modeling the virtual agent's behavior using Petri-net[2].

2 System Configuration for Simulating Collaborative Virtual Environment

In this study, the authors concentrate on the development of the method for constructing the collaborative environment for machine-maintenance task, such as assembling and disassembling machines. Because the machine-maintenance task is one of the most complicated tasks and if the collaborative environment for the machine-maintenance task can be constructed, it would be possible to construct the collaborative environments for the other tasks.

To simulate the collaborative environment for the machine-maintenance task, several kinds of functions should be realized as follows:

1. Both of the real human and the virtual agent can manipulate virtual objects just like in the real world,
2. The virtual agent can demonstrate the complicated tasks to educate the real human, and
3. The real human and the virtual agent can communicate each other through verbal and nonverbal ways.

In consideration of the above-required functions, the system for simulating the collaborative environment should be designed as shown in Figure 1. The system consists of 7 subsystems as follows:

1. Speech Recognition Subsystem (Subsystem 1)

   To make it possible for a real human to communicate with the virtual agent through a verbal way, it is necessary to recognize the utterance of the real human.

2. Gesture Recognition Subsystem (Subsystem 2)

   To make it possible for a real human to manipulate virtual objects, it is necessary to measure the

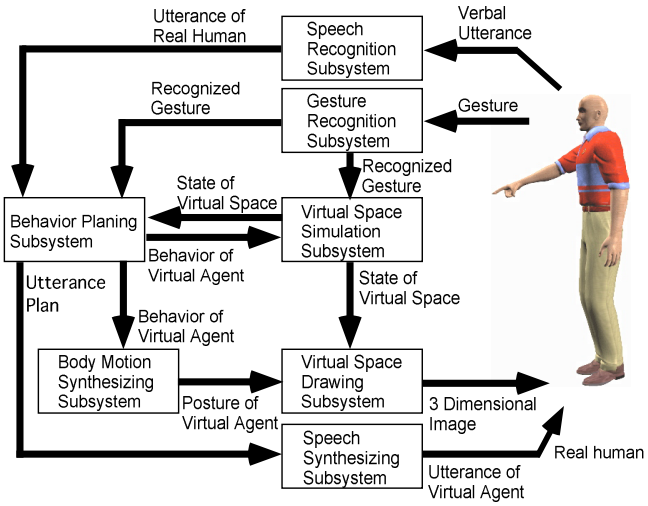

Figure 1: System Configuration for Simulating Collaborative Virtual Environment.

real human's gesture by using datagloves, polhimus sensors and so on.

3. Virtual Space Simulation Subsystem (Subsystem 3)

   To make it possible for both the real human and the virtual agent to manipulate virtual objects just like in the real world, the simulation of the interaction between virtual objects should be based on the related physical laws.

4. Behavior Planing Subsystem (Subsystem 4)

   It is necessary to decide the virtual agent's behavior in according to the states of the virtual space and the real human's gesture. For example, if the real human commits a wrong action, the virtual agent should point it out and let the real human correct the error.

5. Body Motion Synthesizing Subsystem (Subsystem 5)

   To make it possible for the virtual agent to show the demonstration of the complicated tasks, it is necessary to synthesize the virtual agent's body motion as 3 dimensional computer graphics in real time.

6. Virtual Space Drawing Subsystem (Subsystem 6)

   The Virtual Space Drawing Subsystem generates 3 dimensional images by using the information about the current state of the virtual space and the current posture of the vitual agent.

7. Speech Synthesizing Subsystem (Subsystem 7)

   The Speech Synthesizing Subsystem synthesizes the utterance of the virtual agent.

In the case of constructing a new collaborative environment for the different kinds of tasks, it is necessary to reconstruct the subsystems 3, 4 and 5. And the reconstruction of these subsystems with the conventional methods needs a lot of cost, so a new method is required in order to reduce the workload for constructing these subsystems.

Therefore, in this study, new construction methods have been developed with which a collaborative environment for various kinds of tasks can be constructed easily. Concretely, for the construction of the Subsystem 3, an object-oriented construction method for designing virtual spaces has been developed. And for the construction of the Subsystem 4, a new modeling method has been developed for modeling the virtual agent's behavior using Petri-net. Moreover, for the construction of the Subsystem 5, a body motion synthesizing system "AHMSS" has been developed which was designed based on the idea derived from the concept of affordance. In the following sections, these new construction methods are described.

3 Development of the Body Motion Synthesizing System

In this study, a body motion synthesizing system (AHMSS) has been developed with which the virtual agent's motion can be synthesized based on the concept of affordance. By applying the idea derived from the concept of affordance, it is possible to reduce the workload for synthesizing the virtual agent's motion. In this section, how to apply the concept of affordance to design the body motion synthesizing system is described.

3.1 Principle of the System Design

A human has a lot of joints and each joint has the freedom of motion from one to three degrees. So a human has a large number of posture variables. To synthesize the human motion, all of the joint's angles must be specified. Numerous algorithms for synthesizing human motions can be found in literature[3], but all of them are limited to use for synthesizing a particular motion. Then the conventional method of developing a system using computer animation of virtual agents has been like this; first, what kinds of the virtual agent's motion should be synthesized for realizing the system is decided, and then the algorithms and the data for synthesizing those kinds of virtual agent's motion are constructed into the system. Although the motion of the virtual agent can be synthesized by such ways, the system is necessary to be reconstructed again when you would like to change the environment.

As one solution to this problem, there is the concept of affordance[4]. The affordance was introduced by psychologist James Gibson and he defined the affordance as "a specific combination of the properties of substance and its surfaces taken with reference to an animal." According to this concept, an action of a human is triggered by the environment itself where the human exists unlike the afore-mentioned way of interpretation that the human would behave in accordance with the model of the environment the human already possesses in advance.

When this way of thinking would apply to the development of the body motion synthesizing system, the algorithms and the data for synthesizing the virtual agent's motion should be composed not in the virtual agent's brain but in the virtual objects located in the virtual space. And the algorithms and the data should be transferred from the virtual object to the synthesizing system at the time when they become necessary.

Based on the discussions mentioned above, the authors make it the first policy of the system design that

the algorithms and the data necessary for synthesizing the virtual agent's motion are composed in the database not for the virtual agent but for the virtual objects. This design method makes it possible to use a new algorithm for synthesizing a variety of body motions without reconstructing the synthesizing system.

3.2 Mixing Two Kinds of Body Motion

If the distance between a cup and a chair is short enough, a human can grasp the cup while sitting on the chair. In this way, a human can perform an action which relates to two different objects at the same time.

In this study, to make it possible to synthesize the mixed motion, the authors suppose that a human does not accept the affordance uniformly; the degree of the affordance intensity varies in according to every part of the human body. For example, a cup affords the motion "grasp a cup" strongly toward the arm while weakly toward the other part of the body. And a chair affords the motion "sit on a chair" strongly toward the legs and the hip while weakly toward the rest. If the distance between a cup and a chair is short enough, a human accepts the affordance from both of the cup and the chair. And because the degree of the affordance intensity from the cup is stronger than that from the chair, the arm performs the action "grasp a cup". Likewise, because the degree of the affordance intensity from the chair is stronger than that from the cup, the legs and the hip performs the action "sit on a chair". All together, the human acts "grasp a cup while sitting on a chair".

Based on the discussions mentioned above, the AHMSS is designed to synthesize the mixed motion. Concretely, the concept of the motion weight is introduced. The motion weight is the numerical value which represents the priority of the human motion over the other motion and it is set to each joint of human's body. In the AHMSS, the motion weight is prepared to every action the virtual objects can afford.

The mixed motion can be obtained by using the following equation:

$$\theta(t) = \frac{\theta_1(t)W_1(t) + \theta_2(t)W_2(t)}{W_1(t) + W_2(t)} \quad (1)$$

where $\theta(t)$ is the joint angle of the mixed motion at time t, $\theta_n(t)$ is the joint angle of the original motion n at time t and $W_n(t)$ is the motion weight of the joint for the original motion n at time t $(0 < W_n(t) \leq 100)$. With this approach various kinds of the virtual agent's motions can be synthesized easily by changing the combination of actions and the beginning time of the mixing.

3.3 Support Application for Deciding the Motion Weight

By using the motion weight, it becomes possible to mix two kinds of human motions. But it is very difficult to decide the value of the motion weight, because the appropriate value varies according to the kind of motions.

Then, in this study, the support application for deciding the value of the motion weight has been developed. With this support application, the value of the motion weight can be tuned through Graphical User Interface as shown in Fig. 2. The application user can check the mixed motion repeatedly with various values of the motion weight.

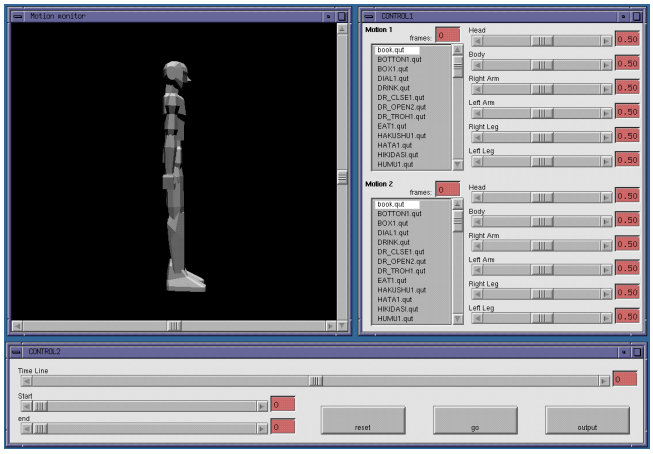

Figure 2: Support Application for Deciding the Motion Weight.

3.4 Examples of the Synthesized Body Motion

Figure 3 shows the example motion of the virtual agent who picks up a cup while walking. In this case, the virtual agent is afforded from both of the floor and the cup. The floor affords the motion "walking" strongly toward the legs while weakly toward the other part of the body. And the cup affords the motion "grasp a cup" strongly toward the arm while weakly toward the other part of the body. Then these affordances are mixed by using the motion weight which was prepared in advance.

4 New Method for Constructing Collaborative Virtual Environment

In this section, the object-oriented method for constructing virtual spaces (OCTAVE) is described. In the OCTAVE method, there are two major classes; property class and virtual object class. One instance of the property class corresponds to one property of the virtual object, and one instance of the virtual object class corresponds to the virtual object. And the virtual object class is defined as the set of the property class. Figure 4 shows the configuration of the virtual object designed with the OCTAVE method. And as shown in Fig. 5, the property class consists of the following 4 elements:

1. A Petri-net for representing the discrete states of the class and their transitions,
2. Plural variables for representing the continuous state of the class,
3. Processes for simulating the property of virtual objects, and
4. The member classes which add various properties to the parent class.

4.1 Petri-net

In the OCTAVE method, a class has two kinds of the internal state, discrete state and continuous state. The discrete state changes according to the events occurred in the virtual space. Each discrete state is defined with

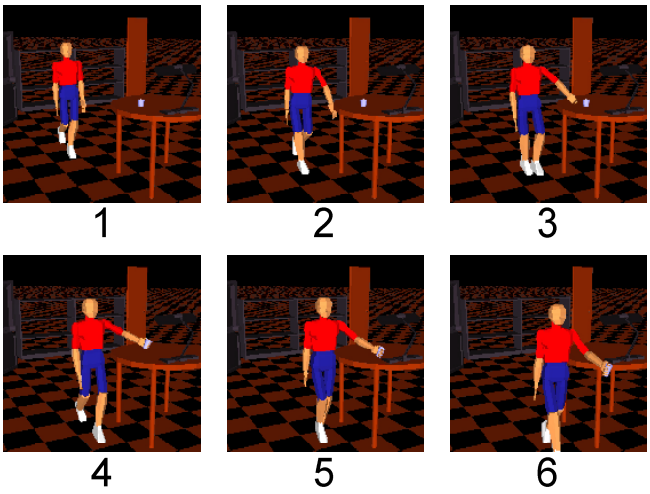

Figure 3: The example snapshots of the virtual agent who picks up a cup while walking (Mixed motion).

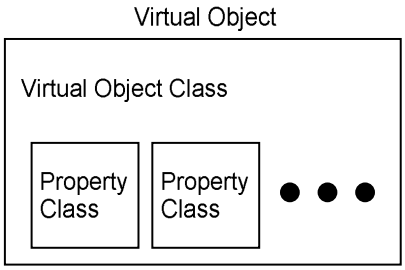

Figure 4: Configuration of the virtual object designed with the OCTAVE method.

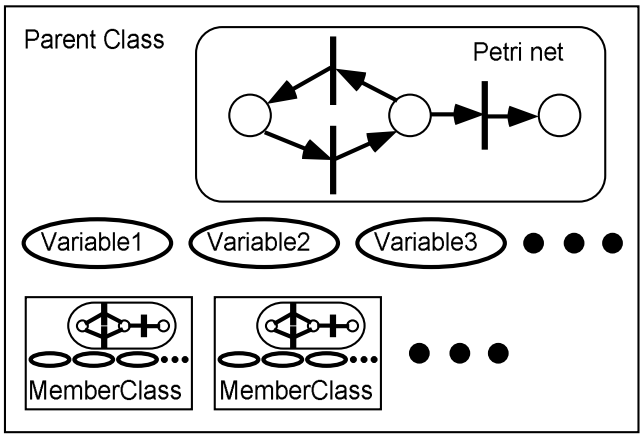

Figure 5: Configuration of the property class.

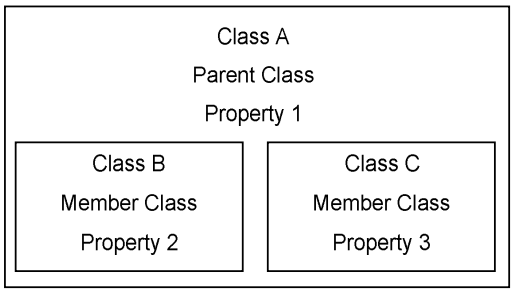

Figure 6: Relationship between a Parent Class and a Member Class.

the process for simulating the property of virtual objects. And when the discrete state changes, the process for simulating the property also changes. For example, when the state of the class is "grasped by the other object which has the property 'able to grasp an object' ", the process for simulating the virtual object's movement is executed so that the virtual object follows the other object. And when the state of the class is "still", the process to fix the virtual object's location and posture is executed.

With the OCTAVE method, the discrete state and its transitions of the class are defined by constructing Petri-net. Concretely, one place in the Petri-net corresponds to one state of the class and the existence of a token in the place represents that the class is in the state of the corresponding place. And one transition in the Petri-net corresponds to one event occurred in the virtual space. Therefore, the transition of the state of the class is modeled as the transition of the tokens in the Petri-net.

The important point of the OCTAVE method is that the property of the virtual objects is designed not by indicating the other objects directly, but by indicating the property of the other objects. For example, to design the situation that a pen is located on a desk, the state of the pen is represented not as 'located on the desk', but as 'located on the object which has the property that some objects can be located on'. By this design methodology, it becomes possible to design virtual objects independently of the other virtual objects.

4.2 Variable

With the OCTAVE method, some variables are used to represent the continuous state of the class which cannot be represented by the Petri-net alone. For example, location, posture, size of virtual objects would change continuously with time.

4.3 Member Class

With the OCTAVE method, a member class is used to add a property to the parent class. Figure 6 shows the relationship between a parent class and a member class. In the Figure 6, the class A, B, C has the property 1, 2, 3 respectively, and the class B and C are the member class of the class A. In this case, the class A has the properties 1, 2, 3 at the same time. By using the relationship between a parent class and a member class in this way, the parent class can be defined by using the definition of the other classes defined in advance. This provision makes it possible to reduce the workload for designing a new class.

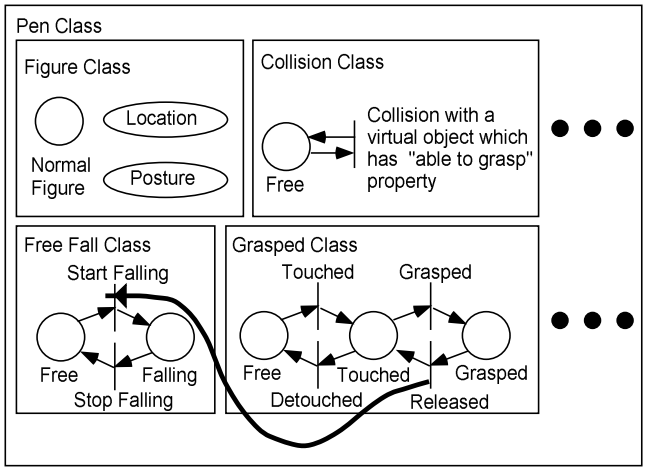

Figure 7: Virtual pen designed with the OCTAVE method.

4.4 Simulating the Interaction among Virtual Objects

With the conventional method for constructing virtual spaces, a virtual object is indicated directly to represent the relationship between two virtual objects. For example, "a pen is located on a desk". But with the OCTAVE method, a property of a virtual object is indicated so that it can make it possible to construct virtual objects independently of the other virtual objects. So, a new function is necessary to simulate the interaction among virtual objects. In this study, a particular function is prepared to mediate the information between virtual objects. This function corresponds to "message passing" of the object-oriented concept. For example, to simulate the collision between virtual objects, a particular function is prepared to calculate the location of the virtual objects, detect the collision and inform the occurrence of the collision to the collided virtual objects. With this function, the simulation of the collision between virtual objects can be realized.

4.5 Example

Figure 7 shows the virtual pen designed with the OCTAVE method. The virtual pen consists of Figure Class, Collision Class, Free Fall Class, Grasped Class and so on. The Figure Class adds the property to the Pen Class that the virtual object has shape, location and posture. The Collision Class adds the property that the virtual object collides with the other virtual object. The Free Fall Class adds the property that the virtual object falls if released. And the Grasped Class adds the property that the virtual object can be grasped by the other object. The variables such as location and posture are calculated by the simulation of the virtual space. For example, when a token exists in the place "Falling", the variable of the location is calculated to simulate the falling state of the virtual object. The arc from the transition "Released" to the transition "Start Falling" means that after the transition 'Released' fires, the transition "Start Falling" fires.

5 Modeling Virtual Agent's Behavior

To make it possible for the virtual agent to educate and support the real human in the collaborative environment, it is necessary that the virtual agent has various kinds of knowledge and can utilize them effectively. In the field of Artificial Intelligence, there are a large number of researches on modeling the virtual agent's knowledge, but the development of the human-like efficient knowledge hasnot been accomplished yet. Therefore the researches tend to be focused on how to develop the human-like efficient knowledge. But the research on how to make it easy to model the virtual agent's knowledge is also important for the practical use.

In this study, to make it easy to model the virtual agent's behavior, the modeling method using Petri-net has been proposed and the support application for constructing Petri-net specialized for modeling virtual agent's behavior has been developed.

5.1 Required Knowledge for Educating and Supporting the Real Human

To make it possible for the virtual agent to educate and support the real human for the machine-maintenance work, the virtual agent needs to have the various kinds of knowledge as follows:

- Physical knowledge

  Physical knowledge is the knowledge which origin is the physical laws. For example, "if a pen is released, it falls down" and "to take off a cover fixed with a nut, it is necessary to take off the nut first" are the Physical Knowledge. This knowledge is necessary to make it possible for the virtual agent to demonstrate and cooperate the maintenance work.

- Spatial Knowledge

  Spatial Knowledge is the knowledge concerning the spatial relationship among virtual objects. For example, "to grasp a pen with a hand, it is necessary to be in the place where the pen is reachable" is the Spatial Knowledge.

- Knowledge about the target machine of the maintenance work

  This knowledge is particular to the target machine of the maintenance work such as the name of the machine, the right procedure for the maintenance work and so on.

- Communication Knowledge

  Communication Knowledge is the knowledge which is necessary when the virtual agent communicates with the real human through verbal and nonverbal ways. For example, in the case of asking the name of the virtual object, it is necessary for the virtual agent to point the target object first and utter "What is this?".

In this study, as the first step of the development, the method for modeling the virtual agent's behavior using Petri-net has been developed. The representation of the virtual agent's behavior using Petri-net enables us to design the virtual agent's behavior visually. Of course, only by the employment of Petri-net, all the knowledge mentioned above cannot be modeled. But the procedural knowledge, which can be modeled with Petri-net

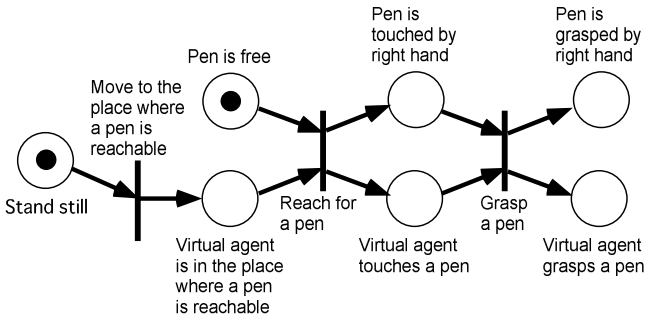

Figure 8: Petri-net representation of the work "grasp a pen with a hand".

easily, is the basis of all knowledge. In the following subsections, the method for modeling virtual agent's behavior using Petri-net and the support application for constructing Petri-net specialized for modeling virtual agent's behavior are described.

5.2 Modeling Virtual Agent's Behavior using Petri-net

In the method for modeling virtual agent's behavior using Petri-net, one place in the Petri-net corresponds to one state of the virtual object or the virtual agent. And the existence of a token in the place represents that the virtual object or the virtual agent is in the state of the corresponding place. And one transition in the Petri-net corresponds to one action of the virtual agent.

For example, Figure 8 shows the Petri-net representation of the work "grasp a pen with a hand". In this case, the pen can be in the state "free", "touched by a hand" and "grasped by a hand". And the virtual agent can be in the state "stand still", "in the place where a pen is reachable", "touches a pen" and "grasps a pen". And Figure 8 means the virtual agent and the pen is in the state "stand still" and "free" respectively.

The place "stand still" and "in the place where a pen is reachable" is the input and output place of the transition "Move to the place where a pen is reachable" respectively. So when the virtual agent acts the action "Move to the place where a pen is reachable", the state of the virtual agent changes from the state "stand still" to the state "in the place where a pen is reachable".

By analyzing the reachability of the Petri-net shown in Fig. 8, it can be found that to become the state of the virtual agent "Virtual agent grasps a pen", the transitions "Move to the place where a pen is reachable", "Reach for a pen" and "Grasp a pen" need to fire sequentially. In this way, by analyzing the reachability of the Petri-net, the virtual agent's behavior can be decided.

5.3 Support Application for Constructing Petri-net

In this study, the support application for constructing Petri-net has been developed with which Petri-net can be designed visually through Graphical User Interface according to the method described in the subsection 5.2. Figure 9 shows the interface of the support application. By using this support application, a large Petri-net can be constructed in short time.

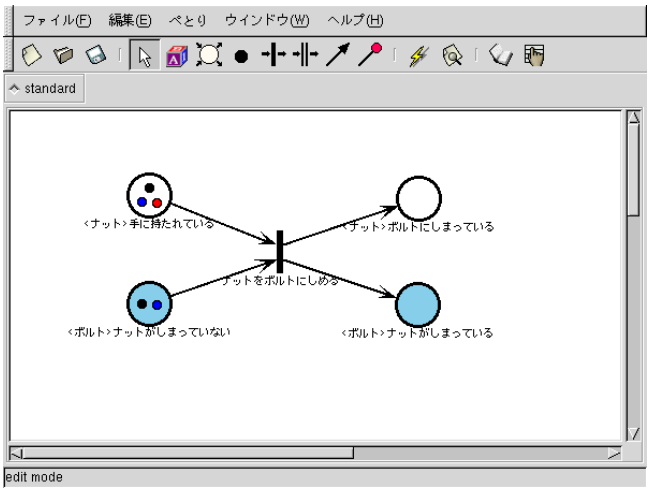

Figure 9: Interface of the Support Application for Modeling Virtual Human's Behavior.

6 Conclusion and Future Work

In this study, new methods for constructing the Collaborative Virtual Environment have been developed to reduce the construction workload. For the construction of the virtual space, the object-oriented method has been proposed to make it possible for the virtual objects to be constructed by combining the components which are independently constructed in advance. And for the synthesis of the virtual agent's body motion, the affordance-based design approach has been proposed and the body motion synthesizing system AHMSS has been developed. Moreover, for the modeling of the virtual agent's behavior, a new modeling method using Petri-net has been proposed and the support application for constructing Petri-net has been developed.

The proposed methods are implemented individually as the different application. Therefore it is necessary to integrate these methods and implement as single application as the future work.

References

[1] Rickel, J. and Johnson, W., "Animated Agents for Procedural Training in Virtual Reality: Perception, Cognition, and Motor Control", *Applied Artificial Intelligence*, Vol. 13, pp. 343-382, 1999.

[2] Peterson, L., "Petri net Theory and the Modeling of System", *Prentice-Hall*, 1981.

[3] Badler, N., Philips, C. and Webber, B., "Simulating Humans: Computer Graphics, Animation, and Control", *Oxford University Press*, 1999.

[4] Gibson, J., "The Ecological Approach to Visual Perception", *Houghton Mifflin Company*, 1979.