Short Paper: #SEVEN, a Sensor Effector Based Scenarios Model for Driving Collaborative Virtual Environment

Guillaume Claude, Valérie Gouranton, Rozenn Bouville Berthelot, Bruno Arnaldi

International Conference on Artificial Reality and Telexistence
Eurographics Symposium on Virtual Environments (2014)
T. Nojima, D. Reiners, and O. Staadt (Editors)

Short Paper: #SEVEN, a Sensor Effector Based Scenarios Model for Driving Collaborative Virtual Environment

G. Claude[†] , V. Gouranton[‡] , R. Bouville Berthelot[§] and B. Arnaldi[¶]

INSA de Rennes, IRISA (- UMR6074), Rennes, France

Abstract

In this paper we present #SEVEN, a sensor effector model that enables the execution of complex scenarios for driving Virtual Reality applications. #SEVEN is based on an enhanced Petri net model which is able to describe and solve intricate event sequences. Our model also proposes several useful features for the design of collaborative scenarios for Collaborative Virtual Environments such as versatile roles and Activity Continuum. We also illustrate its usage it by describing a demonstrator that presents an implementation of our model.

1. Introduction

In a Virtual Reality (VR) application for training, the scenario engine is one of the main subsystems. Its purpose is to orchestrate the events that occur in the Virtual Environment (VE) at runtime. In collaborative virtual environments (CVE), this adaptation must take into account multiple users. To illustrate this assertion, let us take a sequence of actions that must be executed by the same actor even though more than one actor is able to start this sequence. We want our scenario model to be able to describe such constraints.

VE for training are a specific category of VR application that requires the experience of an expert in the targeted area to define the scenarios and the objectives of the training in order to design: what must be done, who has to do it and when it is done. As this expert has the knowledge of the simulated domain and knows what needs to be learned by the trainees, we decided to refer to this person as *"the specialist"* throughout this paper.

In this paper, we are interested in the inner model of a scenario engine for CVE. We decided to exclude the editing formalism of the scenario from the scope of this paper. Indeed, this formalism is not related to runtime but to the authoring of the scenario.

[†] guillaume.claude@irisa.fr
[‡] valerie.gouranton@irisa.fr
[§] rozenn.bouville_berthelot@irisa.fr
[¶] bruno.arnaldi@irisa.fr

Concerning the runtime, we have defined a set of needs:

- The model must allow sequences of actions to be defined at once that must be performed in a **strict order as well as wide open** actions.
- The model must enable the description of complex **event sequences** such as branching or parallelism.
- The model must enable both **actors' actions and the virtual environment behaviours** to be described, depending on the needs of the simulation.
- The model must be able to manage continuous events and discrete events.
- The model must enable complex **collaboration** connections such as team organization and the dynamic assignation of tasks to be described:
  - Each actor has an assigned role in the team related to its capabilities, knowledge and duties.
  - This role can evolve dynamically during the simulation depending on the context.
  - Actions may be related to multiple roles.
  - To be consistent with reality, some sequences of actions must be performed by the actor who initiated them.
- The model must allow the description of scenarios with several **levels of detail**. It allows a scenario to be used with every action specified for the training of beginners and without any guidance from the expert.

2. Related Work

In the literature, the modelling of scenarios for VR applications is divided into two classes: emerging scenario models and predefined scenario models.

2.1. Emerging scenario models

Emerging scenario models do not define precisely what must occur in the environment. They drive the simulation from a set of rules that constrains the behaviour of the different involved agents. The scenario then emerges from these behaviours [PHM*03, LC06]. This approach did not fit our needs as we had to describe the events that must occur during the simulation precisely: the specialist may not be able to describe precisely the next possible events.

2.2. Predefined scenario models

Predefined scenario models focus on the sequence of the events. The specialist defines these sequences when writing the scenario. Some events may have complex arrangements such as parallelism or branching. This class of model uses different kinds of automata to model the arrangement of the events. State machines are widely used to respond to this problem. HCSM [CKP95] proposed a hierarchical state machine structure that was later extended in engines such as the behaviour model HPTS++ [LD02] and the collaborative scenario engine LORA++ [GMA07] in which each state is a software agent. Other works propose the use of Petri nets [Mur89] in behaviour models [WH01]. Smith and Duke [SDM99] propose to consider VE as an hybrid system and use Hybrid Petri nets to model interaction.

In CVE, the arrangement of the events and, more particularly, the actions of actors are even more complex to handle. The solution used in VR is to give roles to the actors. In this case, the role of an actor defines the allowed actions it can perform at a specific point of the scenario. The role theory [BT66] states that roles (its attributions and its duties towards the team and the goals) can change throughout the life of the team. These evolutions must have been foreseen and controlled by the specialist during the scenario specification. This problem is handled by RoB-MALLET [ZY06] and LORA++ by allowing the role of an actor to be changed at runtime. Another problem raised by collaboration is how to define that the actor who has started a task must be the only one allowed to complete this task. HPTS++ handles the problem by being able to model concurrent access to resources (such as the tool needed to perform the actions of the sequence).

2.3. Synthesis

We have found in the literature many of the features we need for our scenario model. Nevertheless none of them matches our criteria:

- HPTS++ proposed the "perception, decision, action" loop to interact with, and control, the VE but it is not collaborative as its main purpose is to model behaviour.
- LORA++ is able to manage complex teamwork but is focused on the actions of the users, not on driving environment
- Others models propose complex modelling of sequence by relying on models such as Petri nets, but again, collaboration is not embedded [WH01].

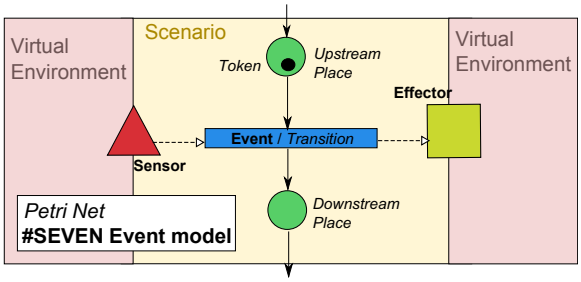

Figure 1: A Petri net enhanced with the event model.

3. Contributions

In the following section we present the #SEVEN scenario model and its features: in section 3.1, we discuss the motivation for using a Petri net-based model; 3.2 describes our model and its main features; 3.3 describes the notion of *Activity Continuum* and its integration in the model;Finally, 3.4 introduces the principle of scenario projection.

3.1. Events sequence

Petri nets provide an expressive yet easy to understand formalism which allows branching (concurrency) and parallelism [BA06] to be modelled. We use hierarchical, safe Petri nets as a tool to model the sequence of the events. The places of a safe net can hold only one token at a time [Mur89]. We use safe nets because we wanted to model the scenario of our VR system as a unique process. The transitions are the events of the scenario. An event is marked as fireable when the transition is sensitized (i.e. when all of the upstream places of the transition hold a token). This net model is enhanced by giving the ability to the places to be Petri nets.

In #SEVEN we have integrated the notions of initial and final places. Initial places define the initial marking [Mur89] of the net when it is initialized. Final places define that the scenario is finished when they all hold a token. Using these two elements, we have added the ability to the places to also be Petri nets. When a place which is a sub-scenario receives a token, it duplicates this token in each of its initial places. From the parent scenario point of view, the place holds a token when all of its final places hold one. At the beginning of the simulation, if a scenario is the root scenario (i.e. it has no parent scenario) then its initial places are marked with a

token. If all of the final places of the root scenario hold a token, the scenario is completed.

3.2. The event model

The **perception** and **action** capabilities of #SEVEN are brought about by extending the transitions of the net as an event. An input element - the sensor - and output element - the effector - are attached to the event. Figure 1 shows the simplest example of a Petri net with the event model.

Perception The purpose of the sensor is to describe the conditions that indicate that an event has occurred. This element listens to the VE until it reaches a specific state or an event occurs. When these conditions are met, the sensor fires its related event if possible (i.e. it is sensitized as a Petri net transition as all its upstream places hold a token). For instance, we have defined specific sensors such as the auto-trigger (fired instantly when its related event is fireable, without any other condition) or the action sensor (fired when a user executes a specific action such as "Pushes the button up" or "Takes the bridles"). There are no forced behaviours for access of the sensors to the environment; it depends on the implementation of the type of the sensor.

Action When an event is fired, it triggers its related effector. The effector's task is to modify the elements of the VE by changing some variables or starting some processes. For instance, an effector can start an animation or change some attributes of an object, such as its inner state.

Execution At each simulation step, the system checks all the sensors one by one. If the conditions of the sensor are met, it tries to trigger its related event. If all of the upstream places hold a token, the event is triggered: it executes its related effector, consumes the tokens in the upstream places and produces a token in the downstream places. Then it stops the loop. At the beginning of the next step, the system continues the checks, starting with the next sensor after the last triggered one. The system stops when it reaches a sensor it has already checked, when no sensor is fired in a loop.

Hybrid system The management of the discrete and continuous components of the simulation is easily handled by the event model. Sensors can transform a continuous signal to a discrete one by applying threshold functions or complex conditions. For example: The sensor is waiting for the winch to reach a defined area. If the coordinates of the winch are not in the specified area (defined by three hysteresis on x,y,z) then the event cannot be activated. When the coordinates match, the sensor triggers the event.

3.3. Activity continuum

In some scenarios, the actor who started an action is forced to complete the started process by executing the following actions. We have proposed an extension of our Petri net implementation which allows this kind of behaviours to be modelled. For this purpose we propose the concept of *Activity Continuum*. An *Activity Continuum* is a set of parameters that is common to all of the actions in a sequence. In this paper we have limited these parameters to identifiers related to the actors.

As in coloured Petri nets [Jen87], we have added some data to the token. These data define the *Activity Continuum* using a set of unique identifiers (one for each actor). The transitions are labelled with functions whose purpose is to define the data on the output tokens from the data on the input tokens.

At runtime, a user can see a sensor as fireable only if its identifier is present on, at least, one token. As the tokens of a sub-scenario are viewed as a unique token by its parent, the token data are equal to the union of the id present on the final places tokens. Knowing this, if the actions available to a user is defined by the sensors fireable for this user, it is possible to restrict the execution of the action to a specific set of actors based on the execution context. Our adaptation of coloured Petri nets does not impact the capabilities of the safe Petri net model we used as a basis. This model adds some predicates on the fireability of the sensors. The scenario labelled with functions has an equivalent Petri net without function which is an unfolding of all of the possible executions of the labelled one (one by identifier).

3.4. Roles and collaboration

In the role theory [BT66], "role" is defined as concepts held by anyone about the behaviours of a person or a position. The theory also states that the role of an actor can evolve with time. In CVE for Training the role defines the skills an actor has and the actions the actor is able to execute during the simulation. In order to fit with role theory, we have embedded in #SEVEN the ability to give a fine-grained description of the actions related to roles. Furthermore, these descriptions of roles can be changed dynamically. To achieve this, we decorate the events with "assignments".

Assignments Assignments are sets of actions an actor can do in the environment. Actor handled events are labelled with potentially multiple assignments. The assignments are then divided between all of the actors. The role of an actor is the union of all of its assignments. An assignment is not specific to an actor and multiple actors can have the same set of assignments and, by extension, the same role.

Dynamic assignment changes Granting or removing an actor's assignments allows the granting or removal of accesses to subnets of the Petri net. We have created a type of effector that is able to add and/or remove assignments from actors depending on their set of assignments. As the assignments of the actor change, its reading of the scenario

changes too and it may be able to access some actions while losing the access to others. The set of actions related to an assignment is the responsibility of the specialist.

3.5. Complete Use Case

Our use case is based on a real situation that implies at least two coworkers in a maintenance procedure (see https://vimeo.com/104937822).

We used #SEVEN in a collaborative VE for the training for the maintenance of an injection moulding machine. This procedure consists of changing the mould of a plastics injection machine; it requires at least two teammates that cooperate to manipulate and fix a new mould which can weigh several tons.

#SEVEN enables us to easily distribute the two roles required by the simulated procedure. Using #SEVEN's assignment feature, we arranged the height assignments attributions that derived from the tasks that compose the procedure to define the roles of the actors. Finally, it has been possible to describe a complex sequence of events that most of the existing scenario models are unable to define simply. Furthermore, we were able to define several *Activity Continuum* (see 3.3) so that the scenario is closer to reality and to the context of the training simulation.

4. Conclusion and Future Work

We propose #SEVEN, a scenario engine model based on hierarchical safe Petri nets, improved by specific features for driving CVE. #SEVEN uses a "perception, decision, action" loop to update itself. Thus, it not only guides users in carrying out their duties, but it also drives the VE by triggering and perceiving changes. #SEVEN allows the writing of scenarios ranging from the constrained to the wide open. Moreover, they can be improved with the dynamic evolution of roles of the actors. Based on these advanced features, the execution of the scenario is modified by taking into account the history of events as well as the state of the environment. Our model is able to express all the features we needed:

- It can define complex events and actor actions sequences.
- It can perceive discrete and continuous events.
- It can describe complex collaboration circumstances.
- It can interact with the virtual environment.

We deliberately focused our efforts on the inner model of the engine, leaving behind the formalism for authoring. Moreover, the *Activity Continuum* feature is for now limited to actors and needs an extension to manage both the actor and the execution context of the action. In addition, there is still work to do on the deadlock detection and correction during both authoring and at runtime.

Acknowledgements

This publication is supported by the S3PM project of the CominLabs Excellence Center. The prototype has been developed with the help of Thomas Boggini.

References

[BA06] BROM C., ABONYI A.: Petri-nets for game plot. In *Proceedings of AISB artificial intelligence and simulation behaviour convention* (2006), vol. 3. 2

[BT66] BIDDLE B., THOMAS E.: Role theory: concepts and research. 2, 3

[CKP95] CREMER J., KEARNEY J., PAPELIS Y.: HCSM: A framework for behavior and scenario control in virtual environments. *ACM Trans. Model. Comput. Simul. 5*, 3 (July 1995). 2

[GMA07] GERBAUD S., MOLLET N., ARNALDI B.: Virtual environments for training: From individual learning to collaboration with humanoids. In *Technologies for E-Learning and Digital Entertainment*, vol. 4469 of *LNCS*. Springer Berlin Heidelberg, 2007. 2

[Jen87] JENSEN K.: Coloured petri nets. In *Petri Nets: Central Models and Their Properties*, vol. 254 of *LNCS*. Springer Berlin Heidelberg, 1987. 3

[LC06] LUGRIN J.-L., CAVAZZA M.: AI-based world behaviour for emergent narratives. In *Proceedings of the 2006 ACM SIGCHI International Conference on Advances in Computer Entertainment Technology* (New York, NY, USA, 2006), ACE '06, ACM. 2

[LD02] LAMARCHE F., DONIKIAN S.: Automatic orchestration of behaviours through the management of resources and priority levels. In *Proceedings of the First International Joint Conference on Autonomous Agents and Multiagent Systems: Part 3* (New York, NY, USA, 2002), AAMAS '02, ACM. 2

[Mur89] MURATA T.: Petri nets: Properties, analysis and applications. *Proceedings of the IEEE 77*, 4 (Apr. 1989). 2

[PHM*03] PONDER M., HERBELIN B., MOLET T., SCHERTENLIEB S., ULICNY B., PAPAGIANNAKIS G., MAGNENAT-THALMANN N., THALMANN D.: Immersive VR decision training: Telling interactive stories featuring advanced virtual human simulation technologies. In *Proceedings of the Workshop on Virtual Environments 2003* (New York, NY, USA, 2003), EGVE '03, ACM. 2

[SDM99] SMITH S., DUKE D., MASSINK M.: The hybrid world of virtual environments. In *Computer Graphics Forum* (1999), vol. 18, Wiley Online Library. 2

[WH01] WILLANS J., HARRISON M.: Verifying the behaviour of virtual environment world objects. In *Interactive Systems Design, Specification, and Verification*, vol. 1946 of *LNCS*. Springer Berlin Heidelberg, 2001. 2

[ZY06] ZHANG Y., YIN J.: A role-based modeling for agent teams. In *Distributed Intelligent Systems: Collective Intelligence and Its Applications, 2006. DIS 2006. IEEE Workshop on* (June 2006). 2