Journal of Visual Languages and Computing 21 (2010) 136–156

Contents lists available at ScienceDirect

Journal of Visual Languages and Computing

journal homepage: www.elsevier.com/locate/jvlc

A methodology to specify three-dimensional interaction using Petri Nets

Rafael Rieder[a,*], Alberto Barbosa Raposo[b,1], Márcio Sarroglia Pinho[a,2]

[a] Pontifícia Universidade Católica do Rio Grande do Sul – PUCRS, Faculdade de Informática – FACIN, Avenida Ipiranga, 6681, Prédio 32, Sala 607, CEP 90619-900 Porto Alegre, RS, Brazil

[b] Pontifícia Universidade Católica do Rio de Janeiro, PUC-Rio Grupo de Tecnologia em Computação Gráfica, TECGRAF Rua Marquês de São Vicente, 225, Prédio Belisário Velloso, CEP 22453-900 Rio de Janeiro, RJ, Brazil

A R T I C L E I N F O

Article history:
Received 5 November 2007
Received in revised form
28 October 2009
Accepted 14 January 2010

Keywords:
Interaction tasks
Petri Nets
Specification
Code generation

A B S T R A C T

This work presents a methodology to formally model and to build three-dimensional interaction tasks in virtual environments using three different tools: Petri Nets, the Interaction Technique Decomposition taxonomy, and Object-Oriented techniques. User operations in the virtual environment are represented as Petri Net nodes; these nodes, when linked, represent the interaction process stages. In our methodology, places represent all the states an application can reach, transitions define the conditions to start an action, and tokens embody the data manipulated by the application. As a result of this modeling process we automatically generate the core of the application's source code. We also use a Petri Net execution library to run the application code. In order to facilitate the application modeling, we have adapted Dia, a well-known graphical diagram editor, to support Petri Nets creation and code generation. The integration of these approaches results in a modular application, based on Petri Nets formalism that allows for the specification of an interaction task and for the reuse of developed blocks in new virtual environment projects.

1. Introduction

The development process of Virtual Reality (VR) applications still uses *ad-hoc* modeling and implementation techniques, with very little standardization and almost no formalism. This can be noticed particularly in efforts devoted to scientific applications, leading, in most cases, to code rewriting and hindering application analysis before its implementation.

In order to better understand a VR application, especially its possibly intricate interaction flow, it is very helpful to use some kind of formal method like Petri Nets (PN), the Unified Modeling Language (UML), or Finite State Machines (FSM), which can describe the system's function and components. These methods provide not only a better understanding but also a preliminary evaluation of each phase of the system's operation. Moreover, a model-based description facilitates the automatic generation of the core application code from graphical representations. The model-based code generation approach already has produced interesting results in other domains, such as the development of user interfaces for mobile computing [17,35], and the development of web applications [5,21].

Besides formal specification tools, some researchers have sought to develop taxonomies to document and specify virtual environments (VEs) at an abstraction level closer to the user's view instead of the programmer's concept of the application. VEs use nontraditional devices and techniques for three-dimensional interactions that need to be carefully planned, since different physical cues

* Corresponding author. Tel.: +55 51 3320 3611; fax: +55 51 3320 3621.
E-mail addresses: rafael.rieder@pucrs.br (R. Rieder), abraposo@tecgraf.puc-rio.br (A.B. Raposo), pinho@pucrs.br (M.S. Pinho).
[1] Tel.: +55 21 2512 5984; fax: +55 21 3527 1848.
[2] Tel.: +55 51 3320 3611; fax: +55 51 3320 3621.

1045-926X/$ - see front matter © 2010 Elsevier Ltd. All rights reserved.
doi:10.1016/j.jvlc.2010.01.002

are mapped during a computer simulation to allow the user's tasks to be performed directly in a three-dimensional spatial context. Some taxonomies seek, for example, to identify the interaction process phases [6], to classify interaction techniques [7,18,25], and to organize the system's control [10]. These approaches split the systems into smaller parts, identifying behavior patterns and allowing to encapsulate them into classes that are capable of executing relevant functionalities. They also allow reusing these classes in other projects and combining them to build a new interaction technique, for example.

The use of both formalisms and taxonomies aims to better define the interaction processes, reducing the time spent for the design and implementation of VEs. Therefore, an integration of both approaches can join the best of each: system specification according to the user's level of expertise, evaluation in the early stages of the development process, and detailed information on each phase of the software development process.

This paper describes a methodology that supports the design and implementation of software modules, which represent the interaction process phases. Our methodology integrates three modeling approaches: PN formalism [19], the interaction technique decomposition taxonomy created by Bowman and Hodges [6] and object-oriented programming concepts. The combination of these elements allows for the description of interaction tasks, the sequence of interaction processes being controlled by PNs with the codes generated automatically.

The PNs are used to graphically and formally represent the VE behavior patterns, based upon the phases of the interaction process according to Bowman's taxonomy. The adoption of these approaches provides a model that can be easily coded using a set of C++ classes. A PN simulator is used to control the program's execution flow.

The choice of PNs to specify tasks in VEs emerges when we start using Bowman's methodology because the interaction tasks can be easily understood as transitions, while the states reached by the application can be understood as places in the PN. With this approach, the definition of independent modules to represent the system's functionalities is straightforward. The logical separation into modules is important, for example, to develop interaction technique frameworks, or to facilitate an automatic code generation from a tested and validated model.

By integrating a taxonomy for interaction techniques, the formalism of PN and automatic code generation, the present work addresses the entire development cycle of a three-dimensional interaction. This cycle begins at the design stage, based on Bowman's interaction taxonomy, then moves to validation and debugging using PN simulation, and ends up with automatic code generation. This approach is different from existing works because the design process is focused on the user's view and on task decomposition, which allows analyzing the system in different levels of detail, making the communication between development teams and end-users more effective during the entire project. Existing approaches are specialized in specific parts of the cycle, as will be discussed in the section about related work.

This text is organized as follows: first we present a literature review in Section 2. Section 3 describes the developed methodology. In Section 4 we present a case study applying the methodology, and we show the necessary steps from application modeling to code generation for a VE. This section also discusses the possibility of hierarchical modeling using our methodology. In Section 5, we describe two case studies that illustrate the use of our approach in realistic settings, such as cooperative manipulation and innovative three-dimensional interaction applications. Section 6 presents a brief discussion about the use of the developed methodology, whereas Section 7 describes the goals we want to achieve with future work. Section 8 concludes the paper highlighting the potential of our approach.

2. Related work

Smith and Duke [28] point out that the lack of formal descriptions during the development process of virtual environments inhibits the identification of similarities among different interaction techniques, leading to the "reinvention" of existing techniques. Furthermore, according to Navarre et al. [20], informal descriptions are prone to ambiguities during the implementation process.

Different mechanisms have been proposed by the VR community to describe and implement interaction techniques, seeking to understand the dynamic behavior patterns of the applications and allowing the standardization of important functions.

This section presents an overview of the methodologies used to specify and implement the interaction process in VEs. Techniques and frameworks used to define the base of this work are briefly mentioned, focusing on the goals, advantages and disadvantages of each one of them.

2.1. Approaches for interaction technique specification

Interaction technique specification is an important task from the perspective of both users and designers. Users require interaction techniques that allow them to complete interaction tasks in a particular application, and designers have to build systems that make the required interactions possible [27]. The existing approaches for interaction technique specification that are more closely related to this work are presented in the following paragraphs.

HyNet [31] is a specification methodology for interaction techniques that integrates three modeling approaches. High-level PNs represent the formal base for the specification, defining the application's semantics and allowing a graphical representation of the application's events (the discrete part of the application). Differential Algebraic Equations handle the continuous behavior pattern of the application, and Object-Oriented Concepts help enhance the expressiveness of the methodology, generating concise and compact models.

Based on HyNet, the Flownet methodology [32] was developed to describe dynamic behavior patterns in VEs.