Software Architectures for Designing Virtual Reality Applications

Rafael Capilla and Margarita Martínez

Department of Informatics and Telematics, Universidad Rey Juan Carlos, Madrid, Spain
{rcapilla,mmartinez}@escet.urjc.es

Abstract. Software architectures are particularly useful when designing complex systems. Apart from facilitating the design, development and evolution processes, software architectures help developers who are new in the domain to understand the design issues involved, reducing the learning effort. In this work we present a software architecture for virtual reality systems. This architecture applies patterns common in other interactive systems, such as the Model-View-Controller, and also identifies new patterns proper of the VR domain, such as the scene graph. In addition, in the proposed architecture we have identified the variability points needed for adapting and evolving such VR systems.

1 Introduction

The challenge of a virtual reality system (VR-system) is to simulate the real world in a virtual environment, making it real for the user who is immersed in the system [8]. Among the characteristics of virtual reality systems that make them complex to develop we can cite the following.

- Use of special hardware devices: head-mounted displays, 3D haptics, etc.
- Complexity of user interfaces and multimodal interaction.
- 3D modeling techniques.
- Complex graphic operations, such as object collision and deformation.
- Presence of the user in the virtual scene.
- Real-time requirements.

We believe that all these factors can preclude a quick development of virtual reality software applications. The challenge to facilitate the construction of such systems is the motivation of this paper. Therefore we have tried to provide some guidance for understanding and designing VR systems from a software engineering perspective. Our proposal analyses the use of software architectures [2] [4] [21] for building virtual reality applications in order to reduce the effort needed in the development process by making more modular and reusable its most relevant parts. The organization of the paper is as follows. Section 2 presents the related work in this field. Section 3 describes our approach for building software architectures for VR systems. Section 4 evaluates our work through the construction of a VR system and section 5 provides the conclusions obtained.
2 Software Architectures in Virtual Reality Systems

We have found only a few proposals in the literature describing virtual reality systems employing software architectures. Schöntage and Eliëns [18] describe the problems for developing distributed VR systems and they mention four architectural styles for classifying object-oriented software under distributed environments. In [19] the DIVA Distributed Architecture Visualization is presented as an example of the use of architectural patterns [5] to achieve a software architecture [20] for building distributed VR systems. Another example is the Dragon system, which is a real-time battlefield visualization virtual environment [10]. The architecture is shown in figure 1.

The Dragon system is composed by two major subsystems: the rendering engine (RE) and a generic entity manager (GEM). RE draws the virtual environment, processes the user input and creates requests and events that are sent to the GEM subsystem. GEM is responsible to collect data from external sources and represent them under a common format. Both subsystems interact through a pair of unidirectional event queues and a shared entity database. DIS (Distributed Interactive Simulation) and NBIS (Joint Maritime Command Interaction) are interfaces of the system. In [3], the authors mention the requirements that software architectures must fulfil for supporting the design of VR systems as well as to perform rapid prototyping. Some of these requirements refer to the modularization and extensibility as a practical point of view in the design process.

In general terms, the lack of flexibility is a common point in many of the VR systems already developed. One of the problems in the development of Large Scale Virtual Environments [14] comes from the use of monolithic architectures, blocking some important aspects such as maintenance, reusability and extensibility among others. Therefore, a more modular design of VR applications based on good software architectures can improve the development and maintenance of these complex systems.