Developing VR Applications for the Grid

Christoph Anthes, Roland Landertshamer, Helmut Bressler, and Jens Volkert

GUP, Institute of Graphics and Parallel Processing
Johannes Kepler University, Altenbergerstraße 69, A-4040 Linz, Austria
canthes@gup.uni-linz.ac.at

Abstract. In the recent years advancements in the development of Networked Virtual Environments (NVEs) can be observed in many domains. Although this technology is available, it is still a challenging task to design and produce these virtual worlds. To ease the creation of such environments the inVRs framework was developed.

Besides this also grid environments have advanced from traditional batch processing systems in the area of scientific computation. Nowadays highly responsive and interactive grid jobs can be supported: Real-time Online Interactive Applications (ROIAs) constitute a potential domain.

The edutain@grid middleware provides an approach to use the advantages of Grid technology, like Virtual Organisations (VOs), dynamic resource allocation, etc. and additionally fulfils the requirements of highly interactive real-time applications.

The rtfOdrom application acts as a prototype application to demonstrate the interconnection between Grid computing and Virtual Reality.

1 Introduction

Multi-user environments with participants all over the world have become more common and are well accepted by the industry. Community platforms like Second Life or multi-player games like World of Warcraft are just a few to mention. Networked Virtual Environments (NVEs) are becoming more and more valuable, for training simulations or collaborative visualisations, since the required graphics and real-time needs can be fulfilled due to the advancement in network infrastructure and the development of graphics boards. Collaborators from all over the world try to solve large scale problem by simulations and parameter studies using distributed computing power.

This distributed computing power has become available through Grid computing [9]. Grid architectures provide many other features than just providing computational resources. They offer for example user and resource management in the form of Virtual Organisations (VOs).

One of the main issues in NVEs and Virtual Reality (VR) applications in general is the real-time interactivity, which is seldomly supported by Grid middleware. Traditional Grid computing architectures provide batch processing

E. César et al. (Eds.): Euro-Par 2008 Workshops, LNCS 5415, pp. 352–360, 2009.
© Springer-Verlag Berlin Heidelberg 2009

mechanisms, which can be used comfortably to manage distributed computing power, but they are not usable for interactive data manipulation or visualisation.

The advantages of both domains can be combined by merging functionality of the inVRs framework [2] and the edutain@grid middleware [8]. inVRs is designed to support the creation of efficient NVEs by offering a highly modular architecture and consistent communication mechanisms, while edutain@grid supports Real-time Online Interactive Applications (ROIAs) using a Real-time Framework (RTF) for scalable and interactive communication and the GRIA Grid middleware [15] to support the establishment of VOs.

This paper provides an overview how these different worlds of computational resource management and real-time interactivity can be combined. As an example application rtfOdrom, a VR racing game which is executed on different Grid servers, is described.

The second section gives an overview on the related work, while section three focuses on the architecture of the inVRs framework. The exchangeability of modules and the structure of the network module are drawn out in detail. Section four and five introduce the architectures of the RTF and edutain@grid. The following sections concentrate on the combination of these frameworks and describe rtfOdrom as an example for a real-time application running on the Grid. Finally the last section concludes the paper and gives an outlook into future work.

2 Related Work

Although much research has been done in the fields of Grid computing and Virtual Reality (VR) virtually no approaches exist which try to combine both areas.

Many ways exist for the development of VR applications. Typically three different categories of approaches are common. The first approach is to develop a the VR application from scratch, by using low-level APIs or scene graphs. The second way is the use of fully developed NVEs like DIVE [6] or Graphical User Interfaces (GUIs) like the EON Studio. Finally VEs often make use of frameworks e.g. VRJuggler [12] or DIVERSE [13]. While the first two solutions lack genericity and flexibility, the framework approach can be used to adapt the network capabilities to the needs of Grid computing.

In the area of Grid computing a variety of middleware solutions has been provided. Most of them focus on traditional batch processing approaches. In the CrossGrid project [5] data for a flooding simulation was computed on Grid resources and reduced to real-time display and finally displayed in VR systems like the CAVE [7]. Other approaches like AGJuggler try to make use of the AccessGrid in order to display VEs using Grid resources [11].

The key issues with these approaches is that they either make use of the encrypted communication mechanisms provided by the underlying Grid middleware, or they perform significant offline computation beforehand in order to display a static result or set of results in real-time after the computation. rtfOdrom in combination with the RTF and the inVRs framework allows for non-encrypted communication and thus provides the required real-time capabilities.

3 inVRs Architecture

The inVRs framework offers a clearly structured approach for the design and creation of highly interactive and responsive VR applications in order to improve the development process of VEs and NVEs.

It consists of three independent modules, for interaction, for navigation, and for network communication, two interface layers that allow the abstraction of user input and output display, as well as a system core which stores and manages the state of the VE. An architecture draft of inVRs has been previously described in [2].

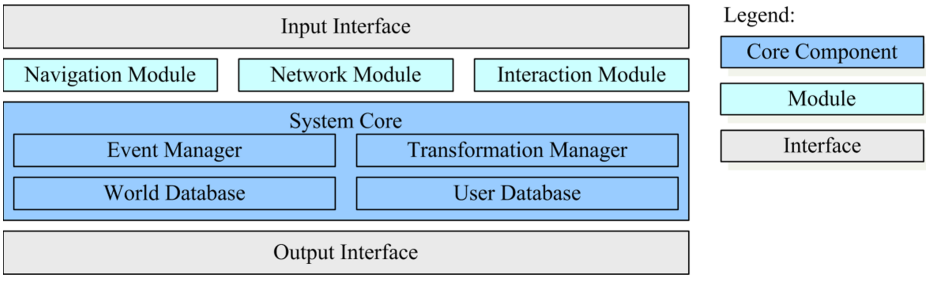

Fig. 1. The Architecture of the inVRs Framework

Figure 1 provides an overview of the inVRs architecture showing the individual components. The flow of data as displayed in this diagram is typically from top to bottom. Input gathered by the devices is parsed by the input interface and exposed to the modules in a data structure describing an abstract controller in order to provide a unified interface. The modules access the abstract controller and generate navigation and data which is processed by the system core managers. The event manger handles discrete reliable events while the transformation manager is responsible for a flow of transformation data packets. More detail on event and transformation management is provided in [1]. Events and transformation data are applied on the user database and the world database which are used to store the state of the VE. The content of these databases is then finally rendered each display frame via the output interface.

In the current implementation inVRs uses OpenSG [14] as a scene graph for rendering the graphical output on a single or multiple stereoscopic displays. Audio output is supported by OpenAL a well known audio library. The input can be retrieved from a variety of sources which could either be regular desktop devices like mice or keyboards, or it can be gathered from VR tracking systems, wands or datagloves.

3.1 Module Exchange

One of the key features behind the design of the framework is its modularity. The need for this becomes clear if we take a look at different application domains.

Some VEs only need a single module e.g. an architecture walkthrough may only need the navigation module. Others such as a networked collaborative safety applications have obviously different requirements. For more interactive VEs like training or phobia treatment applications, both, the interaction as well as the navigation module can be used by interconnecting them through the system core. If an application is designed for multiple users the network module is additionally connected to the core. State changes and entity transformations of the virtual world are in this case transmitted via the network module to the remote participants of the VE.

To achieve this flexibility of module exchange the individual modules are implemented following a plug-in pattern, thus the network module can be easily exchanged in order to support Grid applications. The communication between the managers of the system core and the network module is handled by a high-level interface, which has to be implemented by each inVRs module.

3.2 Network Module

Figure 2 gives an overview on the network module. The communication mechanisms of the system core communicate with the high-level layer of the module via message queues. Geometrical transformations are sent to the network module as unreliable messages and events are sent to be transmitted in a reliable way.

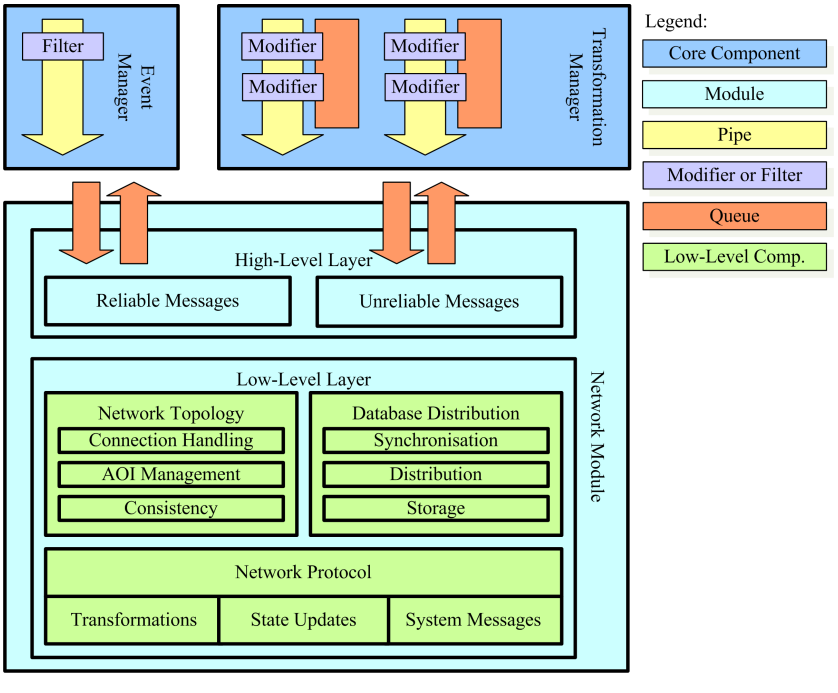

Fig. 2. A High-Level view on the Network Module

The low-level layer shows in a generic view three different aspects which have to be considered for developing an inVRs network application. The message protocol is used for the distribution of geometrical transformations, state updates, and system messages. The network topology handles the connection establishment and communication between the different interconnected nodes, while the database distribution is responsible for storing and managing the states of the virtual world.

This low-level layer of the module has been completely replaced by the communication mechanisms of the Real-time Framework (RTF) in order to be executed on the edutain@grid middleware.

4 RTF Architecture

The RTF was designed to support the scalability of multi-user games by keeping up to the needs of high interactivity.

It is composed of multiple modules which provide controlling, monitoring, communication, streaming, and data storing functionality. Via the controlling module it is possible to startup or shutdown application server instances or migrate server applications from one host to another. The monitoring module can be used to integrate monitoring functionality into application servers. Streaming and data storing functionality are also available for the support of audio and video data transmission and to achieve persistence in networked applications. For the communication between the application clients and the servers the Communication and Computation Parallelisation (CCP) Module is used. This module is integrated into the server and client applications and manages the network connection and communication. It supports different network protocols and provides functionality for the automatic synchronisation of application state information between servers and clients.

The RTF implements a client server architecture which splits up the VE into domains an distributes it over several servers. The VE is populated by entities, which are either static or dynamic objects or avatars representing a single client instance. Entities can travel between domains via portals. On the client side the process of being transferred between domains is transparent to the application even if the destination domain is managed by another server the client's avatar is currently residing on. This is achieved by establishing network connections to potential destination servers when the avatar gets close to a portal to another domain. The migration itself is then achieved by changing the communication destination to the new server.

5 edutain@grid Architecture

The edutain@grid architecture is separated into three different layers, the business layer, the management layer and the real-time layer.

The business layer is the top layer of the system. It is responsible for the management of the different business relationships between the different actors in

the edutain@grid system. From the consumer's point of view this layer provides functionality for login and account management or billing information.

Below the business layer acts the management layer. This layer provides the functionality of resource allocation, monitoring and capacity management. It manages the distribution of the different server applications and supports load balancing mechanisms in order to fulfil Quality of Service (QoS) parameters defined in the business layer.

The third layer in the edutain@grid architecture is the real-time layer. This layer focuses on the real-time communication between the application servers and the client applications. It supports advanced communication functionality like the automatic distribution of game entities or e-learning application data from the servers to the clients. Furthermore this layer supports different network communication protocols and allows to integrate monitoring functionality into the application servers.

The edutain@grid middleware provides interesting features which can be used to support virtual worlds, like a high scalability, which is achieved by the RTF component of the real-time layer. Features like portals provide a user-interface for the end user on many sides of the system (e.g. client, hoster, coordinator).

This middleware has been combined with the inVRs framework in order to connect both worlds.

6 Combining the Three Solutions

In order to interconnect the three approaches the RTF middleware was integrated in the inVRs framework as an individual network module.

The inVRs network module does not follow the principles of a classical client server approach. It is up to the individual modules to maintain a consistent view of the VE of their respective area they are responsible for. The reasoning behind this is that it is more efficient to treat synchronization issues on a per module level rather than on a per application instance level, as different modules may have different needs. The inVRs physics module for example has to distribute the results of the physics simulation to all participants in the VE in order to obtain consistent and vivid object behaviour. However the inVRs architecture does not rule out the possibility of application instances participating in a common network behaving in a different way. In fact the representation of the VE stored locally in the world database of an application instance is designed to be modified by a remote instance without causing any consistency issues. This enables to implement a client server architecture on top of the inVRs framework where the behaviour of entities, both in the RTF and inVRs sense, is organized solely by the server.

Another concept common to the inVRs and the RTF is the idea of having a single user in the VE per application instance. Glinka et al. have described the combination of RTF and legacy applications in [10].

7 The rtfOdrom Example Application

The rtfOdrom is a multi-user VR racing game based on netOdrom, which was originally developed using the inVRs framework, with a simple peer-to-peer interconnection to support two players. A detailed overview of the architecture of the original netOdrom application is described in [3].

In the rtfOdrom is a vehicle corresponds to a RTF-client. The virtual world where the race is taking place is distributed over several RTF servers. The vehicles controlled by the user are represented by users in the inVRs framework. There are also movable obstacles on the racing track which are implemented entities. A physics engine is provided manoeuvring the vehicles and managing vehicle-vehicle and vehicle-obstacle collisions.

The rtfOdrom was primarily designed for testing purposes. Beside focusing on testing RTF components the rtfOdrom was also used to conduct experiments regarding which components of a multi-user application may follow a loose consistency model. Parts of the racing simulation are accessing directly the local world database rather than relying solely on world database updates of the server. In particular it was important to hide the latency associated with user input arising from network lag. Therefore it is inevitable to simulate the vehicle movement within the local application instance which is a behaviour not supported by the RTF framework but corresponds to the approach taken by most inVRs applications.

In order to prevent the clients and servers WorldDatabase state from drifting apart the physics engine has been modified on the client side. Artificial forces and momenta are applied to entities in the local world database in such a way that the state of the server is eventually reached. A description of the implemented physics engine is given by Bressler et. al [4].

8 Conclusions and Future Work

This paper has given an overview of the issues of the combination of Grid computing and VR applications. A brief introduction into the inVRs framework was given which can be incorporated to overcome these issues.

Through the exchange of the inVRs network module with the edutain@grid middleware it was demonstrated that the advantages of grid computing can be used to support interactive and vivid virtual worlds. As an example additional computing resources could be provided in case the computational load on on of the servers used rises above a given threshold.

It is still challenging to display data generated by computational Grid in real-time, but by using automatic geometry reduction mechanisms and storing the post-processed data on ROIA servers it could be even possible to display the data of such applications in real-time.

Acknowledgments

The work described in this paper is supported in part by the European Union through the IST-034601 project "edutain@grid".

References

1. Anthes, C., Landertshamer, R., Bressler, H., Volkert, J.: Managing transformations and events in networked virtual environments. In: Cham, T.-J., Cai, J., Dorai, C., Rajan, D., Chua, T.-S., Chia, L.-T. (eds.) MMM 2007. LNCS, vol. 4352, pp. 722–729. Springer, Heidelberg (2006)
2. Anthes, C., Volkert, J.: Invrs - a framework for building interactive networked virtual reality systems. In: Gerndt, M., Kranzlmüller, D. (eds.) HPCC 2006. LNCS, vol. 4208, pp. 894–904. Springer, Heidelberg (2006)
3. Anthes, C., Wilhelm, A., Landertshamer, R., Bressler, H., Volkert, J.: Net'O'Drom– An Example for the Development of Networked Immersive VR Applications. In: Shi, Y., van Albada, G.D., Dongarra, J., Sloot, P.M.A. (eds.) ICCS 2007. LNCS, vol. 4488, pp. 752–759. Springer, Heidelberg (2007)
4. Bressler, H., Landertshamer, R., Anthes, C., Volkert, J.: An efficient physics engine for virtual worlds. In: medi@terra 2006, Athens, Greece, October 2006, pp. 152–158 (2006)
5. Bubak, M., Holger Marten, J.M., Meyer, N., Noga, M., Sloot, P.A.M., Turala, M.: Crossgrid - development of grid environment for interactive applications. In: PIONEER, Poznan, Poland, April 2002, pp. 97–112 (2002)
6. Carlsson, C., Hagsand, O.: Dive - a platform for multiuser virtual environments. Computers and Graphics 17(6), 663–669 (1993)
7. Cruz-Neira, C., Sandin, D.J., Defanti, T.A., Kenyon, R.V., Hart, J.C.: The cave: Audio visual experience automatic virtual environment. Communications of the ACM 35(6), 64–72 (1992)
8. Fahringer, T., Anthes, C., Arragon, A., Lipaj, A., Müller-Iden, J., Rawlings, C.M., Prodan, R., Surridge, M.: The edutain@grid project. In: Veit, D.J., Altmann, J. (eds.) GECON 2007. LNCS, vol. 4685, pp. 182–187. Springer, Heidelberg (2007)
9. Foster, I., Kesselman, C., Tuecke, S.: The anatomy of the grid enabling scalable virtual organizations. International Journal of High Performance Computing Applications 15(3), 200–222 (2001)
10. Glinka, F., Ploss, A., Gorlatch, S., Müller-Iden, J.: High-level development of multiserver online games. International Journal of Computer Games Technology 2008 16 (2008)
11. Gonzalez, D., Arns, L.: Agjuggler. In: I-Light Symposium (September 2005)
12. Just, C.D., Bierbaum, A.D., Baker, A., Cruz-Neira, C.: Vrjuggler: A framework for virtual reality development. In: International Immersive Projection Technology Workshop (IPT 1998), Ames, IA, USA, May 1998. ICEMT (1998)
13. Kelso, J., Arsenault, L.E., Satterfield, S.G., Kriz, R.D.: Diverse: A framework for building extensible and reconfigurable device independent virtual environments. In: IEEE Virtual Reality (VR 2002), Orlando, FL, USA, pp. 183–190. IEEE Computer Society, Los Alamitos (2002)

14. Reiners, D.: OpenSG: A Scene Graph System for Flexible and Efficient Realtime Rendering for Virtual and Augmented Reality Applications. Ph.D thesis, Technische Universität Darmstadt (May 2002)
15. Mike Surridge, S., Taylor, D., De Roure, D., Zaluska, E.: Experiences with gria – industrial applications on a web services grid. In: First International Conference on e-Science and Grid Computing, pp. 98–105. IEEE Computer Society Press, Los Alamitos (2005)