SYNTHETIC-NATURAL CAMERA FOR DISTRIBUTED IMMERSIVE ENVIRONMENTS

Krystian Ignasiak, Marcin Morgoś, Władysław Skarbek

Multimedia Laboratory, Institute of Radioelectronics, Warsaw University of Technology

ABSTRACT

The concept of synthetic-natural hybrid (SNH) camera and microphone is used to design fully immersive environments with SNH scene distributed among distant locations in a network. SNH camera and microphone in natural way integrate three algorithmic aspects in such software design: audio visual (AV) scene view generating, view rendering and view controlling by user. At user terminal SNH camera and microphone is implemented by RendGen (rendering and generating) application which is a client of the specific immersive application server SNHgen, synchronizing views and events in the whole system. RendGen application is actually a collection of plug-ins which could be either general or specific to the particular location, too. A software methodology is developed to define SNHgen servers and integrate plug-ins with RendGen clients. Our approach is illustrated on the basis of virtual classroom application.

1. INTRODUCTION

Immersive environments become gradually important multimedia paradigm used in advanced applications such as distance surgery, virtual classroom, virtual museum tours, 3D games, etc.

Fully immersive environments not only integrate synthetic data (3D graphics) with natural audio-video (AV) streams but deal as well with interactive aspects in local (always) and in networked (frequently) circumstances.

The synthetic-natural, i.e. hybrid (SNH) scenes in which users are immersed at rendering add new challenges for software design especially in networked case. Besides classical AV synchronization problem we encounter rendering complexity and interactivity coordination problems.

Rendering time complexity is reduced significantly by using Open-GL graphics cards with advanced extensions. However, additional dimension for rendering complexity is added when source data used to build the immersive scene is of various modality, e.g. image based rendering versus classical 3D graphics rendering based on triangle meshes and associated textures. When SNH scene has different parts with such different rendering modalities, the integration of them at the rendering engine creates several technical problems, especially if different rendering parts are developed by different programming teams.

In section 2 we show how the concept of synthetic-natural hybrid (SNH) camera and microphone is used to design a software platform for fully immersive environments with SNH scene distributed among distant locations in a network. Namely, the coherent client server architecture is presented. Section 3 defines functionalities for selected software modules (plug-ins) developed in the system while section 4 illustrates the approach in a virtual classroom application.

2. SOFTWARE ARCHITECTURE

SNH camera and microphone in natural way integrate three algorithmic aspects in software design for immersive environments: SNH AV scene view generating, view rendering and view controlling by user. At user terminal SNH camera and microphone is implemented by RendGen (rendering and generating) application which is a client of the specific immersive application server SNHgen, synchronizing views and events in the whole system. RendGen application is actually a collection of plug-ins which could be either general or specific to the particular location, too.

In order to create a software platform for immersive environments we defined the software architecture which is illustrated in Figure 1.

The main element of this architecture is HybridGenerator which is implemented in HybridGenerator C++ class. The HybridGenerator: provides clients with the scene description and animation of the scene, manages events within the application, manages the interaction within the scene (between users, between objects and between users and objects), provides the programmer with network communication methods and a uniform way of event management.

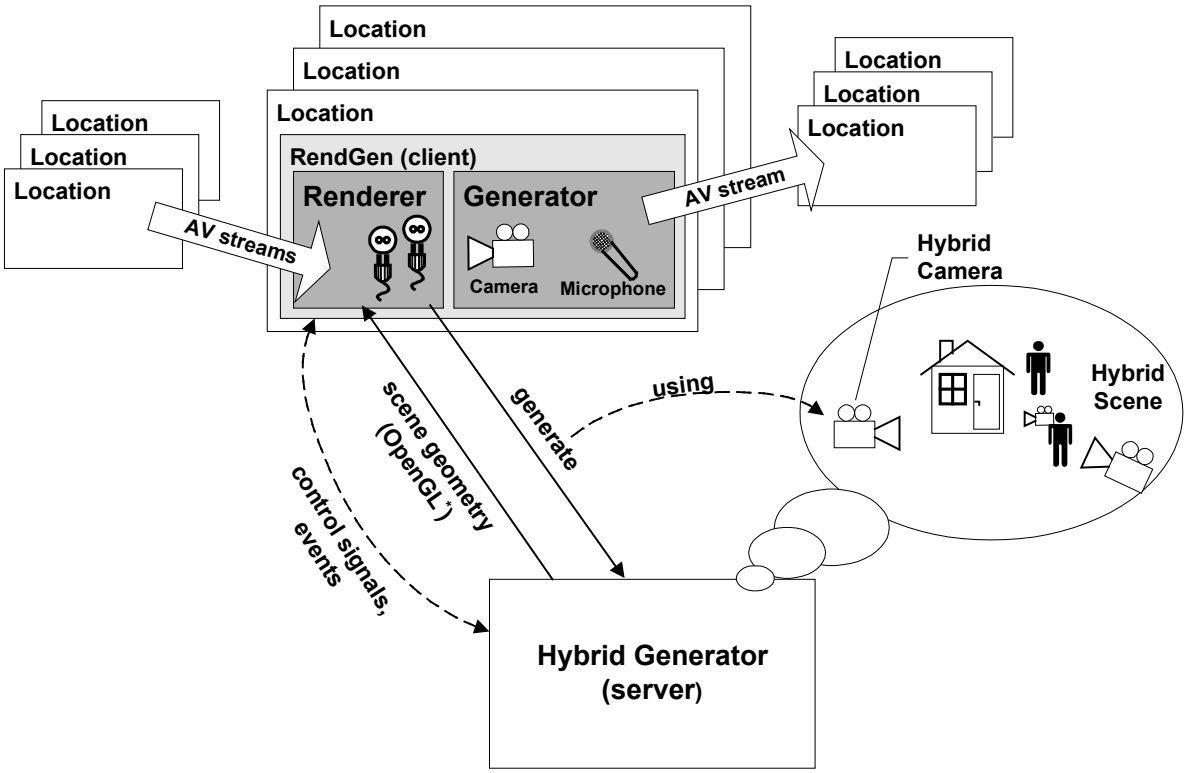

Figure 1 Architecture of rendering engine.

The HybridGenerator acts as the server for RendGen client application which is the universal browser for the end user. RendGen (**Rend**erer-**Gen**erator) consists of two modules:

1. **renderer** – it displays the scene in a viewport; renderer asks HybridGenerator to send the scene geometry as a list of OpenGL commands to be executed in a client application; renderer combines the scene geometry together with other signals (e.g. live video) coming from other locations;
2. **generator** – it manages data sources available at the client side, e.g. video cameras, and sends the signals to different locations, like other RendGens, so that it participates in generating the whole scene by HybridGenerator.

The knowledge of how to combine the scene geometry together with live AV streams from other locations is managed by the HybridGenerator.

3. SELECTED PLUG-INS DESCRIPTION

There are twelve plugins available in this version of Rendering API: Viewport, AVPlayer, Capture, ImageLoader, MemBufManager, Model3DLoader, MulticastSender, MulticastReceiver, ScreenCapture, TextTerminal, VideoCompressor, VideoDecompressor.

The functionality for selected plug-ins is as follows:

1. **Viewport**. Hybrid Generators use this plugin for displaying 2D/3D graphics, animations of the whole scene, which are based on OpenGL. Viewport plugin creates OpenGL MDI child window in the client area of RendGen application window. So far the subset of 46 OpenGL commands has been implemented. These commands are chosen to cover some of the most typical OpenGL operations on 2D/3D graphics.
2. **AVPlayer**. Provides application with an easy programming interface to use AVI movie files in a composed scene and to display them within Viewport plugin window (e.g. as the "texture" of 3D object). This plugin creates an ImageResource and fills this resource with current data from the movie file. The resource is available for any other plugins running within RendGen.
3. **Capture**. Provides application with an ability to obtain video streams from any capture

device registered at the operating system level. Such a video stream can be used as a "texture" for objects or a background for the scene. The captured data, i.e. the created resource, is available for any other plugin running within RendGen.

4. **ImageLoader**. Loads given file resource (of type IMAGE) into memory and creates ImageResource, which is then available for other plugins. Only Windows BMP image file format is supported so far.
5. **MemBufManager**. Memory Buffer Manager – the function of this plugin is to manage shared memory. The memory can be shared between other plugins and remotely between plugins and HybridGenerator. This plugin is mainly provided for transferring large OpenGL data sets, like vertices, indices, colors, normals, buffers, and any other data the application deals with.
6. **TextTerminal**. The plugin provides an easy and flexible bi-directional text communication mechanism between user and HybridGenerator. It may also be used to provide users with text chat functions. This plugin creates MDI child window in the client area of RendGen.

The communication between elements of system relies on events handling. Events are sent from a plugin to HybridGenerator, from one plugin to another plugin, etc. in both directions.

Applications created with the proposed Rendering API may use standard, universal plugins as well as specific ones, created only for the particular task and application. Those plugins are created by subclassing the RendGenPlugin C++ class. In order to do that, several methods of this class need to be implemented such as: createInstance, onEvent, onShow, closePlugin, isClosed. The plugin communicates with RendGen application, other plugins and HybridGenerator using the same uniform event management mechanism (for instance with sendEvent and onEvent methods), used by RendGen to communicate with remote HybridGenerator.

Plugin may also share and exchange some data with other plugins in the form of various Resource objects. Plugins have access to the methods of RendGen application that are provided to control and synchronize usage of resources shared between plugins.

4. VIRTUAL CLASSROOM APPLICATION

Virtual classroom application was chosen as a main test application for Rendering API for Immersive Environments, since it combines all types of possible interaction within the scene, as well as main features of immersive environment. These features are necessary to provide the user with high level of realism, e.g. combining a real video stream with artificial objects at a variable level of virtual and artificial world presence.

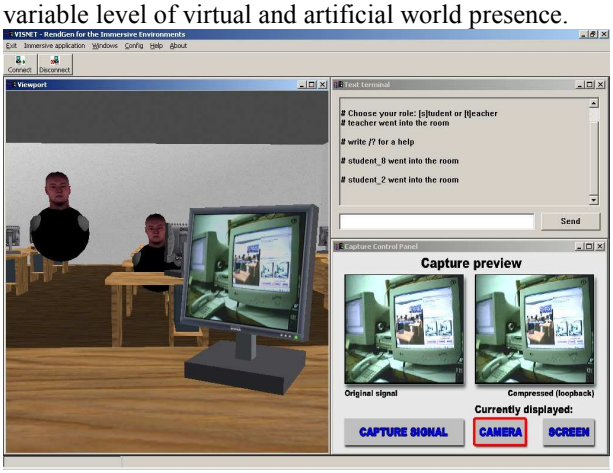

Figure 2 Screenshot of virtual classroom GUI for teacher user role.

Virtual classroom presented to the user contains various 3D object such as blackboard, desks, computers, monitors as well as 3D models of teacher and other students. The teacher's interface of virtual classroom application, has an option to capture video stream from output of any application (or the whole desktop) running on the same computer (MS Excel, PowerPoint, Paint for instance). Additionally teacher's interface provides an opportunity to capture video signal from any capture device (and video camera) available in the operating system (see Figure 2). Those video streams are compressed, transmitted through network using multicast protocol, and finally rendered on the virtual blackboard or virtual monitor's screens, so that any student in the same virtual classroom can see them.

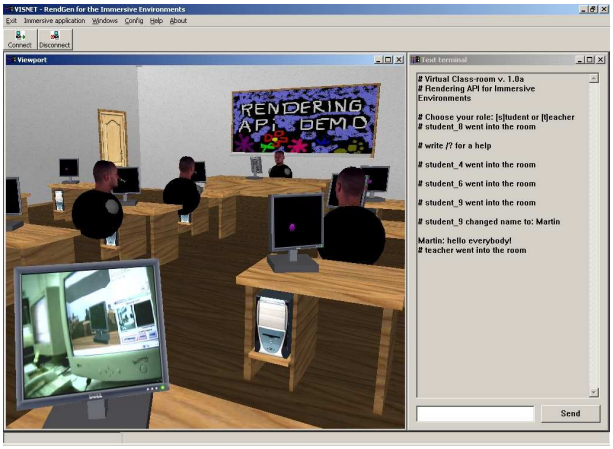

Figure 3 Screenshot of virtual classroom GUI for student user role.

Users, that means teacher and students, can look around and move in this virtual environment. The specific attribute of this application is that video streams are the most important part of the presentation layer, because they carry on the essence and the content of so called *virtual lesson*. The other elements of presentation layer are less important, that's why 3D models of teacher and students are rather symbolic (however they contain naturally looking 3D scans of human faces) as well as other 3D models (classroom's equipment), see Figure 3.

In this application the CPU time is consumed mainly by the compression, decompression and conversion of the video streams.

Virtual classroom uses all standard plugins of the Rendering API as well as one additional plugin developed specially for it, i.e. Model3DLoader. This plugin loads 3D face models in our own format, however if some additional and common used formats would be implemented in this plugin, it can become one of the standard plugins for Rendering API.

5. CONCLUSIONS

The API for applications using concept of synthetic-natural hybrid (SNH) camera was presented. This API allows for building efficient distributed applications which were tested on the basis of virtual classroom. The API allows for easy combining the natural and synthetic elements forming a fully immersive environment.

Acknowledgement The work presented was developed within VISNET, a European Network of Excellence (http://www.visnet-noe.org), funded under the European Commission IST FP6 programme.

6. REFERENCES

[1] Shreiner D., M. Woo, J. Neider, T. Davis, *OpenGL Programming Guide: The Official Guide to Learning OpenGL, Version 1.4, Fourth Edition,* Addison-Wesley, 2003.

[2] Sullivan, W.G., Terpenny, J.P. "A Virtual Classroom Experiment for Teaching the Economic Principles of Engineering Design", Proc. *31st ASEE/IEEE Frontiers in Education Conference*, Reno, USA, October 10 – 13, 2001.

[3] Turoff M., "Designing a Virtual Classroom," Proc. *International Conference on Computer Assisted Instruction ICCAI'95*, Hsinchu, Taiwan, March 7-10, 1995.