SIGGRAPH 1994

21st International Conference On Computer Graphics and Interactive Techniques

Orange County Convention Center
Orlando, Florida
July 24-29

Course Notes 17

PROGRAMMING VIRTUAL WORLDS

Organizers

Anselmo Lastra
University of North Carolina

Henry Fuchs
University of North Carolina

Lecturers

Stephen Ghee
Division Limited

Mark Mine
University of North Carolina

Jon Naughton-Green
Division Limited

Randy Pausch
University of Virginia

Schedule

Virtual Reality: Past, Present, and Future *Henry Fuchs*	30 minutes
Technology for Virtual Reality *Anselmo Lastra*	1 hour 15 minutes
Break	15 minutes
Interaction in a Virtual Environment *Mark Mine*	1 hour 15 minutes
Lunch	
Alice: A Rapid Prototyping System for Virtual Reality *Randy Pausch*	1 hour
Break	15 minutes
Programming Virtual Worlds *Steve Ghee and Jon Naughton-Green*	2 hours

Table of Contents

Appendices

Course Description

Anselmo Lastra

For years there has been a great deal of interest in the field of virtual reality. However, there has not been much user-level activity in the field, mostly because it was very hard for people to experiment with VR. The expense and the difficulty of setting up a system kept all but a hardy and persistent few from setting up laboratories. Within the last couple of years, however, a variety of hardware and software have become available. These new products enable interested parties with moderate budgets to set up VR systems for use in areas such as visualization, and human-computer interaction. This course is intended to get those people started working with virtual worlds.

The emphasis throughout the course is on the practical knowledge necessary for novices to begin working in virtual environments. For example, the section on hardware will describe only commercially available technologies, or those soon to be on the market, not research systems. The organization of the course is straightforward. It covers hardware, design, and implementation of virtual environments.

The morning begins with an overview of a typical VR system, followed by a description of the basic hardware components and how to choose them. This section is important because selecting hardware for VR systems is not straightforward. The course notes will show in detail what hardware is available, but the class time will mostly be spent on describing to the attendees what the important hardware specifications mean, and warning them of potential pitfalls.

The second half of the morning is devoted to the design of virtual worlds. Using applications as case studies, we will illustrate modes of interaction in the virtual environment. The intent is to show the students what sorts of things are possible, and to guide them in solving their application problems using virtual worlds techniques. This section of the course will be illustrated with video tape of a variety of applications for modeling and visualization (architectural, medical, scientific) from the University of North Carolina.

The afternoon session shows the students how to implement their virtual worlds using two very interesting development systems. Rather than survey the field of VR software, we decided to describe two systems in some detail. The intent is to provide enough information for the students to get started, but not bias their choice of software unduly. Again, the emphasis is on the practical, how-to aspects of use for application implementation rather than on the choices made during the designs of the development systems.

The first system, Alice, is an interpreted, object-oriented rapid-prototyping environment. A goal for Alice is to allow programmers to build virtual worlds based on a 10 page, two-hour long tutorial. Alice uses the DIVER subsystem to divorce rendering computation from application computation, thus maintaining high rendering frame rates.

The second system, dVS, is a platform-independent software environment for the development of virtual reality applications, based on a distributed, multi-server architecture. dVS supports multiple-host and multiple-user virtual environments. Higher level functions are provided in the form of an object-based toolkit called VCTools.