Available online at www.sciencedirect.com

SCIENCE DIRECT®

Computers & Graphics 29 (2005) 3–15

COMPUTERS & GRAPHICS

www.elsevier.com/locate/cag

Multimedia integration into the blue-c API

Martin Naef[a,*], Oliver Staadt[b], Markus Gross[a]

[a]Computer Graphics Laboratory, Swiss Federal Institute of Technology, Zurich, Switzerland
[b]Computer Science Department, University of California, Davis, USA

Abstract

In this article, we present the blue-c application programming interface (API) and discuss some of its performance characteristics. The blue-c API is a software toolkit for media-rich, collaborative, immersive virtual reality applications. It provides easy to use interfaces to all blue-c technology, including immersive projection, live 3D video acquisition and streaming, audio, tracking, and gesture recognition. We emphasize on our performance-optimized 3D video handling and rendering pipeline, which is capable of rendering 3D video inlays consisting of up to 30,000 fragments updated at 10 Hz in real time, enabling remote users to meet inside our virtual environment.
© 2004 Elsevier Ltd. All rights reserved.

Keywords: Virtual reality software system; 3D video; Multimedia; Collaborative virtual environments; Telepresence

1. Introduction

The *blue-c* system [1] developed at ETH Zurich provides a novel virtual environment which combines immersive projection with 3D acquisition of the user, allowing remotely located users to meet in a virtual world. blue-c enabling technology includes custom hardware [2] and a new real-time video acquisition and transmission approach [3].

This paper discusses the multimedia integration into the *blue-c application programming interface* (API), a software toolkit that provides easy access to all underlying blue-c technology for the application developer. As opposed to other virtual reality (VR) toolkits that try to separate the VR-specific modules from the graphics rendering and scene graph, the blue-c API tries to integrate much of its functionality into the scene. This helps to keep development interfaces and programming patterns consistent throughout the system without extensive code wrapping efforts. Besides providing access to blue-c-specific technology such as real-time 3D video for telepresence, the blue-c API can also be used as a general-purpose VR toolkit outside the blue-c portals.

This paper focuses on 2D video and performance aspects of our 3D video integration. It also briefly introduces the software architecture. Those aspects that are already covered in detail in [1,3,4] will be omitted. After the system overview, the 2D video system is analyzed. Motivated by the rendering algorithms used for 3D video, we then present a performance-optimized pipeline for turning an incoming dynamic 3D video fragment stream into a vertex array suited for high-performance rendering, discussing the various performance vs. quality trade-offs.

The blue-c application programming environment runs on SGI IRIX™, Linux, and Microsoft Windows™ operating systems. The application code is directly portable between the systems. The API itself has some platform-dependent optimizations to use the available hardware to its full potential.

*Corresponding author.
E-mail addresses: mnaef@acm.org (M. Naef), staadt@cs.ucdavis.edu (O. Staadt), grossm@inf.ethz.ch (M. Gross).

0097-8493/$ - see front matter © 2004 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cag.2004.11.003

The remainder of this paper is structured as follows: Section 2 gives an overview of related work. Section 3 presents the system architecture of the blue-c API. Multimedia services are presented in Section 4, the 3D video service is in more detail in Section 5. We conclude with applications in Section 6 and provide an outlook into the future in Section 7.

2. Related work

Systems for VR are always combinations of many different toolkits and software libraries. The blue-c API is no exception in that respect. This section presents a selection of previous work related to the blue-c API.

Numerous VR application development toolkits have been implemented in the past. CAVElib^{TM} development was started with the initial CAVE^{TM} [5] system and is available as a commercial product (www.vrco.com). It supports device input through the *trackd* system. For rendering, it relies on application-supplied OpenGL code or the Performer scene graph system. Basic networking code for clusters and collaboration is provided, but there is no automatically shared scene graph.

Juggler [6] provides a mature, object-oriented approach to VR. It is very actively supported. As opposed to the blue-c API, but similar to CAVElib^{TM}, Juggler mostly leaves the choice of the rendering system to the application developer and keeps only loose ties to the scene graph. Juggler implements a kernel that keeps several "manager" objects. This concept inspired the blue-c API service structure.

Avango [7], formerly called Avocado, provides similar functionality as the blue-c API. Avango exposes most interfaces to its own scripting language. It is closely coupled to OpenGL Performer, but it changes the scene graph interface to an Inventor-style field system whereas the blue-c API leaves the Performer interfaces unchanged to better support legacy applications. Avango relies on total network ordering and strict locking, which imposes significantly higher requirements onto the underlying network layer.

There is a plethora of toolkits available that provide parts for VR systems, including tracking libraries [8,9], scene graphs [10], networking tools [11], audio servers, etc. Using them together to build a large VR system such as the blue-c, however, requires to learn many different interfaces and concepts, and finding ways to get them to work together smoothly is not always trivial. For the blue-c system, the aim was to provide a holistic, consistent, and well integrated toolkit that provides strong multimedia and basic collaboration support. Unlike other toolkits that separate the scene graph from the rest of the VR system for more flexibility, the blue-c API integrates it into the core for more coherence, allowing to integrate media handling directly into the scene graph without compromising performance.

3. blue-c API system architecture

This section briefly introduces the system architecture of the blue-c API, as presented in [12]. An overview of all components and their main dependencies is given in Fig. 1.

3.1. Core and process management

The blue-c core class is a small kernel that handles system initialization and startup, instantiation and discovery of services, runs the main application loop, and takes care of a clean system shutdown. With the exception of the scene synchronization system that only spawns network transmission threads, most blue-c services spawn individual calculation processes that communicate through a shared arena that is managed by the Performer scene graph system [13].

All process management and locking methods are encapsulated by the API to provide platform

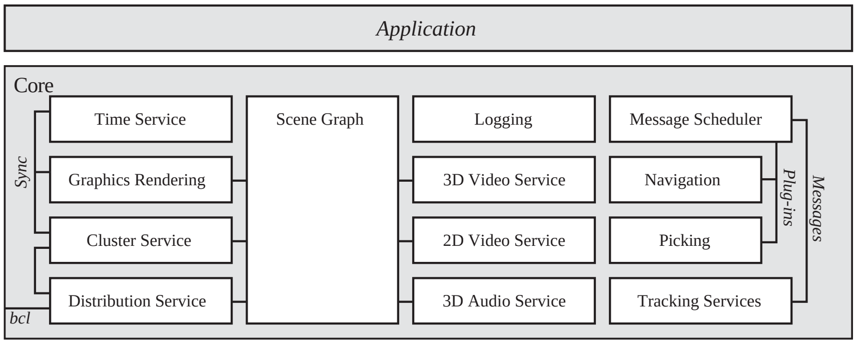

Fig. 1. blue-c API system overview: Services accessing the scene graph, and message scheduler with sources and plug-ins.