THE FLATWORLD SIMULATION CONTROL ARCHITECTURE (FSCA): A FRAMEWORK FOR SCALABLE IMMERSIVE VISUALIZATION SYSTEMS

Anton Treskunov, Jarrell Pair*, and Bill Swartout
Institute for Creative Technologies
University of Southern California
13274 Fiji Way, Suite 600
Marina del Rey, California 90292

ABSTRACT

Motion Picture sets are traditionally built using decorated modular wall components called “flats”. The FlatWorld project (Pair et al., 2003) at the University of Southern California Institute for Creative Technologies merges this practice with immersive technology by creating a system of displays coupled with physical props which can be scaled to simulate entire buildings and streets. A single room prototype FlatWorld system was developed in 2001. The software developed for this prototype was not scalable beyond the simulation of a single room environment. In 2003, the FlatWorld Simulation Control Architecture (FSCA) was developed to support multiple digital flats in arbitrary configurations. The FSCA facilitates digital flat training scenarios which can be scaled from the simulation of a single room up to a complete city block. The architecture’s flexibility allows it to easily interface with a variety of 3D graphics engines and display devices.

1. INTRODUCTION

The FlatWorld system is intended to be a scalable virtual environment which can simulate the experience of being inside a building interior or outside on a city street. This requirement demands a software solution which can manage dozens of 3D graphics displays along with surround speakers, tracking sensors, and effects devices.

A proof of concept FlatWorld system (Figure 1) was built in 2001. This basic system consisted of two displays simulating a room interior with a physical window and door providing views onto an exterior virtual environment. Though this prototype allowed us to test the core functionality of the FlatWorld concept, it could not support a configuration that varied substantially from the simulation of a one room interior.

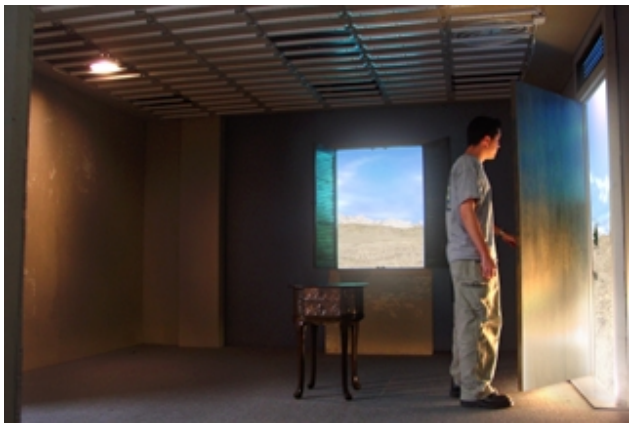

Figure 1: The FlatWorld one room prototype developed in 2001

To move beyond our prototyping work, we required a new software infrastructure that would support multiple system configurations of arbitrary scale along with a wide variety of immersive audio, 3D graphics, and user tracking technologies. To meet these requirements we designed and developed the FlatWorld Simulation Control Architecture (FSCA).

2. RELATED WORK

A number of software libraries have been developed to address the unique requirements of virtual reality application development. CAVELib, WorldToolkit, Avango, and VR Juggler are widely known solutions. Most of these toolkits are oriented to popular immersive display systems such as the immersive workbench and the CAVE (Cruz-Neira et. al, 1993). Many are also distributed as commercial products requiring licensing fees. VR Juggler (Cruz-Neira et. al, 2002) is notable in that it is implemented as an open source toolkit and can be configured to work with different types of immersive displays.

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
00 DEC 2004	**N/A**	**-**

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
The Flatworld Simulation Control Architecture (Fsca): A Framework For Scalable Immersive Visualization Systems	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Institute for Creative Technologies University of Southern California 13274 Fiji Way, Suite 600 Marina del Rey, California 90292**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release, distribution unlimited

13. SUPPLEMENTARY NOTES
See also ADM001736, Proceedings for the Army Science Conference (24th) Held on 29 November - 2 December 2005 in Orlando, Florida. , The original document contains color images.

14. ABSTRACT

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			
unclassified	**unclassified**	**unclassified**	**UU**	**4**	

Though VR Juggler satisfied a number of our requirements, using it would have required a major reworking of our existing code base. We were satisfied with many of the software design decisions used in the prototype system, and we had no need implement them again.

3. SOFTWARE ARCHITECTURE

The FSCA was designed to enable a network-centric system of multiple client displays driven by a single controller application. Each display is abstracted as a unique entity with its own IP address and camera position within the virtual world. In addition to graphics display clients, the FSCA also supports clients which manage tracking devices, audio output, and effects devices (Figure 2). The controller application broadcasts user triggered or scripted event data to all active clients.

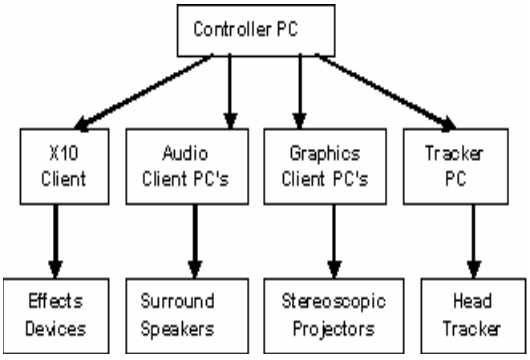

Figure 2: The FSCA controller application broadcasts messages to audio, graphics, tracking, and effects clients.

The FSCA is optimized to run on Windows XP based systems. The FSCA's core libraries are cross-platform making a Linux or Mac OS X port a straight forward task. The processing load is distributed by running the client applications on multiple networked PC's which communicate with the controller application. To date we have developed clients for 3D graphics rendering using game engines, OpenAL based immersive audio processing, and sensory effects device handling using the X10 home automation protocol. Each 3D graphics client renders its scene according to an assigned virtual camera position. By having the capability to tie cameras to specific digital flats, we can easily transform the simulated environment. For example, three digital flats simulating a room interior (Figure 3) can quickly be repositioned and configured to display an outdoor storefront scene (Figure 4).

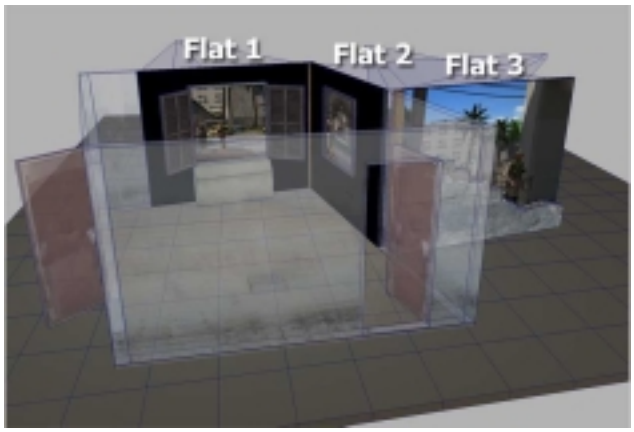

Figure 3: Three digital flats depict a room interior and a view down an exterior alley corridor. Each display is driven by an FSCA 3D graphics client.

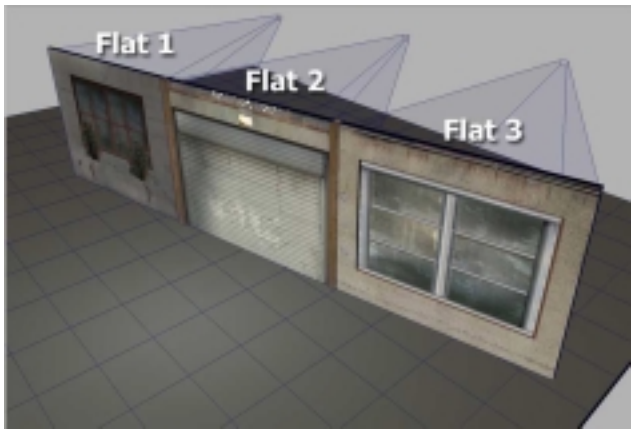

Figure 4: The three digital flats from the previous figure are reconfigured to depict an outdoor storefront scene using the FSCA's virtual camera assignment feature.

3.1 Network Communication

The FSCA's networking component was developed using the Adaptive Communication Environment (ACE), a cross platform, open source library which manages a wide array of concurrent communication tasks. Above ACE we implemented a network message exchange library we call NetMsg. NetMsg is designed such that there is a single master router and multiple slave listeners. The router can broadcast to all registered listeners or to a single listener.

NetMsg uses a multithreaded architecture to handle reconnection tasks, message buffering, and application notifications via C++ functors. Delivery time can be

assigned to each message for events that are time critical. The Network Time Protocol (NTP) is used to synchronize PC clocks. Messages are programmed as C++ classes. Consequently, messages can be added to the system as long as the new message ID's are unique.

3.2 Controller Application Graphic User Interface

The controller application's graphic user interface (GUI) (Figure 5) was built using the open source, cross platform wxWidgets framework. The GUI allows a user to assign virtual camera positions to display clients, select scene databases, trigger animation events, enable input devices, and activate sensors. The 3D scenes are controlled by a sequence of event action states defined in a custom implemented script file.

Figure 5: Controller application graphic user interface shows a 3D scene as viewed on a selected digital flat. Buttons on the lower right trigger scripted events.

4. 3D GRAPHICS RENDERING

With its network-centric infrastructure, the FSCA is not tied to a specific graphics rendering engine. In other words, it is image generator agnostic. An FSCA application can be authored using 3D game technology such as NDL's Gamebryo or Epic's Unreal engine. Alternatively, the FSCA can utilize more traditional 3D simulation engines. Consequently, the FSCA can easily integrate components of legacy simulation and training systems. Our first FSCA application (Figure 6) used OGRE (Object Oriented Graphics Rendering Engine) for 3D graphics rendering. OGRE is also open source and supports Windows, Linux, and Mac OS X.

Figure 6: An OGRE based 3D graphics display client depicts views onto a city street through a physical door and window.

5. FUTURE WORK

In the coming year, we plan to significantly expand the FSCA's core capabilities by implementing new types of clients and adding support for new protocols. Also we are developing FSCA applications to test and validate the architecture's functionality.

5.1 New Sensor and Device Capabilities

We intend to develop a vision sensor client which will enable FlatWorld simulations to act as smart, sentient environments which can dynamically react and adjust to trainee behaviors. We will also improve the FSCA's ability to control effects devices by implementing support for the DMX show control protocol. DMX is a versatile and robust framework which overcomes limitations of the X10 home automation protocol which we are currently using. DMX will expand FlatWorld's multi-sensory effects capability by allowing us to more easily control devices such as strobe lights, fog machines, and air blowers.

5.2 HMD based FSCA application

We are in the process of developing our first FSCA application targeted to primarily use head mounted displays. This application is intended as an experimental platform for treating Iraq war veterans with post traumatic stress disorder.

5.3 FSCA Large Environment Simulation

In 2005, we will be developing a prototype application using several digital flats to simulate an expansive scene depicting both indoor and outdoor spaces (Figure 7, Figure 8, and Figure 9). This application will be appropriate for presence patrol, call for fire, and tactical questioning training.

Figure 7: Depiction of a large indoor/outdoor digital flat simulation driven by the FSCA

Figure 8: Depiction of a simulated city street in a large scale environment FSCA application

Figure 9: 3D rendering of a building interior within the larger simulated environment seen in the previous figures

CONCLUSIONS

The FlatWorld Simulation Control Architecture is a flexible framework for supporting a broad variety of virtual reality software and hardware configurations. The system's versatility allows it to easily build upon legacy simulation systems. Additionally, with the FSCA we can create training applications which can be quickly reconfigured as demanded by the rapidly changing operational environment. Furthermore, the large scale digital flat simulations enabled by the FSCA will give the warfighter the ability to experience an environment before physically being there.

ACKNOWLEDGEMENTS

The authors would like to thank Ernie Eastlund and Matt Liewer for creating the 3D renderings used in this paper.

This paper was developed with funds of the Department of the Army under contract number DAAD 19-99-D-0046. Any opinions, findings and conclusions or recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of the Department of the Army.

REFERENCES

Cruz-Neira, C, Bierbaum, A., Hartling, P., Just, C., and Meinert, K. (2002). VR Juggler – An Open Source Platform for Virtual Reality Applications. *40th AIAA Aerospace Sciences Meeting and Exhibit 2002*, Reno, Nevada.

Cruz-Neira, C., D. Sandin, T. DeFanti. (1993). Surround-Screen Projection-Based Virtual Reality: The Design and Implementation of the CAVE. *In Computer Graphics Proceedings*, Annual Conference Series, ACM SIGGRAPH, New York 1993. pp.135-142.

Pair, J., Neumann U., Piepol D., and Swartout B. (2002). FlatWorld: Combining Hollywood Set Design Techniques with VR. *IEEE Computer Graphics and Applications.* January/February 2003, pp. 12-15.