CaveSL: A Large Format Scalable Multi-display System for Social and Scientific Visualization in Second Life

Kip Haynes* Eric Chance
USC's Institute for Creative Technologies

1. Introduction

As virtual worlds have become more popular for education and socialization, many researchers have begun to utilize virtual worlds like Second Life as a novel method for viewing scientific data [Bor08]. However, the typical means of accessing SL is through a single computer screen, which lessens the immersion that is inherent in such a rich 3D world. Because of this, the SL virtual world is a good candidate for adaptation to large scale immersive displays such as a CAVE and other multi projector systems. CaveSL is a freely available modified SL Viewer we developed that allows researchers to utilize large format multi-display systems for social and scientific visualization.

The goal of CaveSL is to provide a solution that would utilize direct communication between SL viewers and have minimal dependence on 3rd party libraries or interfere with the OpenGL rendering pipeline While there are a few ongoing efforts to adapt SL to a stereoscopic display, there are no freely available native adaptations to a CAVE-like or other multi-screen environment at the time of this writing. There are several methods to distribute OpenGL based applications onto multiple computers and displays such as Chromium, [HFA*02] or VRizer [BLZ04]. However, most of these streaming applications either require changes to the rendering code or are no longer available or supported.

2. Approach

Our implementation uses multiple concurrent SL network logins across an unlimited number of machines. A client-server relationship exists between the user's SL viewer client (*leader*) and the other networked viewers (*followers*). Inside the open source SL viewer, each viewer synchronizes whenever updateCamera() is called. Simultaneously the leader broadcasts the position, rotation and focus of the camera via the message passing interface (MPI).

3. Data and Display Synchronization

The "follower" renderers receive the position of the leader camera and synchronization signal. Each follower reads its assigned rotation from a local config file and performs the proper translation and rotation. A customized 2D rotation was written to handle the translation and rotation of the camera about the X axis. Some simple real time configuration tools have been added to assist in achieving the proper orientation and alignment. Additionally, each node reads its own FOV (field of view) info from a local config, because the Second Life FOV adjustment is difficult to set with any accuracy (slider bar).

*e-mail: haynes@ict.usc.edu

The above described approach provides a well synchronized multi-viewer environment using direct communication between the clients. However, Second Life consists of multiple regions or "sims," which are sections of land that are often controlled by different servers. Initial tests revealed that when the leader crosses a sim boundary, the followers remain mapped to the original sim coordinates and show the camera positions in the old sim. This is because the local coordinates of each region or sim in Second Life conforms to a square grid of 0 – 255 meters. This required a customized message on the native SL scripting side, enabling the follower avatars to subtly follow the leader around the environment and into the next sim. Crossing sim boundaries (switching from one simulation server to another) is an elusive and undocumented process in SL. To solve this problem, we created a vehicle object that the follower avatars could sit on and ride in order to follow the leader's camera around. In order to achieve a proper sim crossing, the movement of the leader is always cached and used as a trajectory for the follower vehicle to be able to cross the sim. In most cases this works well, but if the leader zigzags back and forth across a boundary, the follower can miscalculate the trajectory. We are currently exploring better ways of reporting the current sim to the follower nodes.

4. Limitations

Currently, the system is specifically designed for a large, three-screen VR theater and only supports vertical rotation of the follower cameras. However, it would be very simple to add additional rotations for additional display environments. Also, due to the rapid prototype of this project, the user currently has limited control of some of the native Second Life camera functions. This will be made available in a future release.

References

[BLZ04] BERGER F., LINDINGER C., ZIEGLER W.: VRizer - Using Arbitrary OpenGL Software in the CAVE or other Virtual Environments. *IEEE Virtual Reality, Workshop Proceedings* (July 2004)

[Bor08] BOURKE P.D.:Evaluating Second Life as a tool for collaborative scientific visualisation. *Computer Games and Allied Technology* (April, 2008)

[HFA*02] HUMPHREYS G., HOUSTON M., FRANK R., AHERN R., KIRCHNER S., KLOSOWSKI J.: Chromium: A Stream Processing Framework for interactive Rendering on Clusters. *ACM Transaction on Graphics (SIGGRAPH 2002 Proceedings)* 21,3 (2002) pp. 693-702