High Resolution Video Playback in Immersive Virtual Environments

Han Suk Kim1, Jürgen P. Schulze2, and Larry Smarr1,2
Department of Computer Science and Engineering, University of California San Diego, La Jolla, CA1
California Institute for Telecommunications and Information Technology, University of California San Diego, La Jolla, CA2

1. StarCAVE

- StarCAVE infrastructure in Immersive Visualization Laboratory, at CalIT2
- Immersive Virtual Environment in Immersive Visualization Laboratory
- 5 walls with 3 screens each, plus a floor
- 34 JVC HD2K projectors
- Two JVC HD2K projectors generate a stereo image for each screen

2. Objective

Can we play high resolution (e.g. 4K or HD) video clips in 3D virtual environment?

Applicable to:
- Virtual movie theater
- Video Conferencing
- Virtual office
- Video Surveillance

3. Challenge

- Large Scale Video Texture
  - 4K x 2K x 3 bytes per frame
  - 10 minutes, 25 frames/sec = 900GB
- Constant Frame Rate
  - Constant speed of playback (e.g. 29 fps)
- Performance vs. image quality

4. Mesh Generation

We pre-process the video frames in an off-line step to create the mipmaps. We implemented a tiling technique in which the atomic unit to read and write data is a texture of the same size.

The first step of rendering a video is to subdivide the playback screen into a set of tiles, mesh. The mesh is comprised of multiple tiles of different mipmap levels. The goal of subdividing the screen is to allocate the best possible mipmap level to each region with a limited number of tiles. Another goal is to distribute limited resources of disk read to the region as far as possible. The following algorithm describes the mesh generation procedure.

Algorithm 2 Mesh Generation
1. InitializeQueue qScreen = T, B, L, R
2. List output
3. While qScreen is not empty
5. let parent = qScreen
6. if parent is shown too small on screen to be subdivided then
7. output (screen=parent)
8. else
9. for all Child in screen do
10. if Child is inside screen then
11. qScreen.enqueue(Child)
12. end if
13. end for
14. end if
15. end while

5. Memory Management

- Prefetching
- In order to accelerate the data transfer between main memory and texture memory, a separate thread is spawned and dedicated to asynchronous disk I/O operations. Every disk read request is sent to the I/O thread via a message queue and the I/O thread reads data whenever it finds a message in the queue.
- Asynchronous I/O
- Memory Pool and Cache
- We pre-allocate a pool of memory blocks so data loading can save time for allocating memory blocks. The pool of memory blocks consists of a list of blocks, each of which can store the texture data of one tile. The pool is initialized both in main memory and in texture memory.

6. Synchronization

- Video Frame Rate
- Image Frame Rate in Tiled Display 1
- Image Frame Rate in Tiled Display 2

The time for rendering a frame changes over frames and this can cause two types of synchronization problems: synchronization 1) between frames and 2) between CAVE nodes. A synchronized clock across all nodes of our CAVE software offsets frame numbers so that frame number changes neither too fast nor too slow.

7. Result

We have implemented the video playback algorithm for virtual environments running on PC clusters by writing a C++ plugin for the COVISE software framework. ROCKS manages the cluster nodes, each of which is an Intel quad core Dell XP5 computer with 4GB of main memory and dual Nvidia Quadro 5600 graphics cards.

Figure 5: In our experiment we used two different video clip scenarios: one using a single 4K (3840 x 2160 pixels) clip showing the result of a tornado simulation created by NCSA (left) and the other one consisting of 32 low resolution (384 x 192 pixels) video clips (right). Tiers of 128 x 128 textures were used for both clip sets.

Figure 6: Comparison between ideal video frame rate and image frame rate. The solid red line is the ideal timing for each frame, which we set to 25 fps for this experiment. The dotted blue line is from actual measurements. This shows the playback followed the ideal playback speed.

Table 1: This compares rendering performances at three different resolutions. The first experiment was set to achieve as high resolution as possible. The total number of tiles loaded at every frame was 151, which corresponds roughly to the screen resolution. The lowest image frame rate was 1.79 fps. The medium resolution was about 1.45 fps which we can render at 17 fps. The low resolution setting used only 6 tiles to test the prefetching algorithm. It turned out that all the tiles could be prefetched before rendering of an image frame started.

8. Future Work

DXT compression is a lossy compression technique that guarantees a 6:1 compression ratio for three channel data. As most of our system time is spent on the I/O operations between disk storage and physical memory, this would greatly help the system performance.

Reference