STREAMING GRAPHICAL CONTENT FOR HIGHLY INTERACTIVE 3D APPLICATIONS

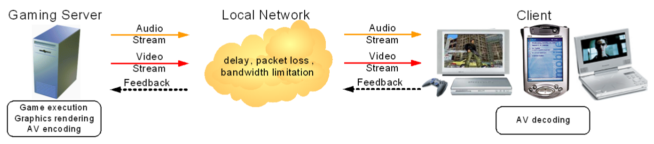
Philipp Fechteler, Benjamin Prestele, Peter Eisert

Fraunhofer Heinrich Hertz Institute, Berlin, Germany
{philipp.fechteler, benjamin.prestele, peter.eisert}@hhi.fraunhofer.de

Abstract

We present a video streaming solution to provide fluent remote access to highly interactive 3D applications, such as games. To fulfill the very low delay and low complexity constraint for this class of applications, several optimizations have been developed. Image preprocessing is implemented on the graphics card to make efficient reuse of the rendered output, as well as the GPU's parallel processing capabilities. H.264/AVC video encoding is accelerated by extracting additional information from the rendering context, which allows for direct calculation of motion vectors and partitioning of macroblocks, thereby omitting the demanding search of generic video encoders. A highly optimized client software has been developed to provide very low delay playback of streamed video and audio, using minimum buffering. In experiments a hardly noticeable delay of less than 40 ms could be achieved.

Keywords: *Games on demand, low delay video streaming.*



Introduction

Remote visualization of interactive 3D applications with high quality and low delay is a long-standing goal. The proposed method is embedded into a larger framework for remote gaming and targets at providing a platform for hotels, internet cafes, and other local environments [1]. Instead of executing the game locally on a PC, it runs on a central server and the output is distributed over a LAN to smaller and cheaper end-devices. Since commercial games shall be supported off-the-shelf, no modifications to the game code can be accepted. Instead, the rendering is intercepted at the graphics API and the output is encoded as video and transmitted to the end-devices. To ensure interactive gaming, it is necessary to achieve very low delay. Furthermore, the encoding complexity should be small, since the encoding is executed in parallel to the game.

Efficient Capturing and GPU Preprocessing

Access to the graphics library (OpenGL/DirectX) is intercepted to capture the frame-buffer after each frame update [2]. In order to accelerate the video encoding, additional render context information is captured during rendering, namely the depth map and transformation matrices. Since the images are generated on the graphics board, the enhanced GPU processing capabilities can be exploited for image processing without any overhead. Inspired by [3], the color conversion and sub-sampling from RGB 4:4:4 to YCbCr 4:2:0 is performed efficiently on the GPU, which also accelerates the GPU$\rightarrow$CPU transfer due to the data reduction by a factor of two.

Enhanced Video Encoding on CPU and GPU

H.264/AVC motion vectors are calculated directly using captured depth maps and transformation matrices [2], thereby omitting the demanding search of generic video encoders. Also, predictions for macroblock partitions are calculated from this information. Additional acceleration is achieved by calculating the motion vectors on the GPU, thereby further reducing the amount of data that has to be transferred from the GPU to the CPU, as well as speeding up the overall processing due to the parallel pipeline of the GPU.

Low Delay Client

Most available H.264/AVC streaming clients are optimized for reliable playback with respect to network jitter. This is achieved by using buffering of up to several seconds, which is unsuitable for interactive applications. To ensure minimum delay, an optimized client software has been developed which uses intelligent buffering for audio and video.

Experimental Results

For evaluation purposes, we have measured the delay between server-side image generation and client-side image presentation, including capturing, encoding, localhost RTP transmission, decoding, and rendering. Experiments at SVGA (800x600) resolution showed a barely noticeable delay of less than 40 ms. The reductions in processing time for the enhanced coding, while preserving coding efficiency, are summarized in the following table at a fixed PSNR image quality of 35 dB:

	enc-time [ms]	bit-rate [MBit/s]
Total Overdose[1]	22.7 (-15.4)%	0.6 (+1.8)%
Extreme Tux Racer[2]	31.8 (-15.5)%	1.3 (+0.4)%

[1] Ego-shooter, 32 bit WindowsXP; [2] Racing game, 64 bit Linux

References

[1] A. Jurgelionis et al. Platform for Distributed 3D Gaming. *Int. J. of Computer Games Technology*, 2009.

[2] P. Fechteler and P. Eisert. Accelerated Video Encoding Using Render Context Information. In *Proc. of ICIP 2010*.

[3] D. Van Rijsselberger. YCoCg(-R) Color Space Conversion on the GPU. In *Proc. of FirW PhD Symposium*, 2005.