Efficient 3D Audio Processing on the GPU

Emmanuel Gallo, Nicolas Tsingos

Efficient 3D Audio Processing with the GPU

Emmanuel Gallo and Nicolas Tsingos
REVES/INRIA Sophia-Antipolis*

Introduction

Audio processing applications are among the most compute-intensive and often rely on additional DSP resources for real-time performance. However, programmable audio DSPs are in general only available to product developers. Professional audio boards with multiple DSPs usually support specific effects and products while consumer "game-audio" hardware still only implements fixed-function pipelines which evolve at a rather slow pace.

The widespread availability and increasing processing power of GPUs could offer an alternative solution. GPU features, like multiply-accumulate instructions or multiple execution units, are similar to those of most DSPs [3]. Besides, 3D audio rendering applications require a significant number of geometric calculations, which are a perfect fit for the GPU. Our feasibility study investigates the use of GPUs for efficient audio processing.

GPU-accelerated audio rendering

We consider a combination of two simple operations commonly used for 3D audio rendering: variable delay-line and filtering [1, 4]. The signal of each sound source is first delayed by the propagation time of the sound wave. This involves resampling the signal at non-integer index values and automatically accounts for Doppler shifting. The signal is then filtered to simulate the effects of source and listener directivity functions, occlusions and propagation through the medium. We resample the signals using linear interpolation between the two closest samples. On the GPU this is achieved through texture resampling. Filtering is implemented using a simple 4-band equalizer. Assuming that input signals are band-pass filtered in a pre-processing step, the equalization is efficiently implemented as a 4-component dot product. For GPU processing, we store the sound signals as RGBA textures, each component holding a band-passed copy of the original sound. Binaural stereo rendering requires applying this pipeline twice, using a direction-dependent delay and equalization for each ear, derived from head-related transfer functions (HRTFs) [1]. Similar audio processing can be used to generate dynamic sub-mixes of multiple sound signals prior to spatial audio rendering (e.g. the perceptual audio rendering of [5]).

We compared an optimized SSE (Intel's Streaming SIMD Extensions) assembly code running on a *Pentium 4 3GHz* processor and an equivalent *Cg/OpenGL* implementation running on a *nVidia GeForce FX 5950 Ultra* graphics board on AGP 8x. Audio was processed at 44.1 KHz using 1024-sample long frames. All processing was 32-bit floating point.

The SSE implementation achieves real-time binaural rendering of 700 sound sources, while the GPU renders up to 580 in one time-frame (≈22.5 ms). However, resampling floating-point textures requires two texture fetches and a linear interpolation in the shader. If floating-point texture resampling was available in hardware, GPU performance would increase. We have simulated this functionality on our GPU using a single texture-fetch and achieved real-time performance for up to 1050 sources. For mono processing, the GPU treats up to 2150 (1 texture fetch)/ 1200 (2 fetches and linear interp.) sources, while the CPU handles 1400 in the same amount of time. Thus, although on average the GPU implementation was about 20% slower than the SSE implementation, it would become 50% faster if floating-point texture resampling was supported in hardware. The latest graphics architectures are likely to significantly improve GPU performance due to their increased number of pipelines and better floating-point texture support.

The huge pixel throughput of the GPU can also be used to improve audio rendering quality without reducing frame-size by recomputing rendering parameters (source-to-listener distance, equalization gains, etc.) on a per-sample rather than per-frame basis. This can be seen as an audio equivalent of per-pixel vs. per-vertex lighting. By storing directivity functions in cube-maps and recomputing propagation delays and distances for each sample, our GPU implementation can still render up to 180 sources in the same time-frame. However, more complex texture addressing calculations are needed in the fragment program due to limited texture size. By replacing such complex texture addressing with a single texture-fetch, we also estimated that direct support for large 1D textures would increase performance by at least a factor of 2.

Can the GPU be a good audio DSP ?

Our first experiments suggest that GPUs can be used for 3D audio processing with similar or increased performance compared to optimized software implementations running on top-of-the-line CPUs. The latest GPUs have been shown to outperform CPUs for a number of other tasks, including Fast Fourier Transform, a tool widely used for audio processing [2].

However, several shortcomings still prevent an efficient use of GPUs for mainstream audio processing applications. Due to limitations in texture-access modes and texture-size, long 1D textures cannot be indexed easily. Infinite impulse response (recursive) filtering cannot be implemented efficiently since past values are usually unavailable when rendering a given pixel in fragment programs. As suggested in [2], including persistent registers to accumulate results across fragments might solve this problem.

Slow AGP readbacks might also become an issue when large amounts of audio data must be transfered from graphics memory to the audio hardware for playback. However, upcoming PCI Express support should solve this problem for most applications.

Finally, our results demonstrate that game-audio hardware, borrowing from graphics architectures and shading languages, may benefit from including programmable "voice shaders", enabling per-sample processing, prior to their main effects processor.

References

[1] Durand R. Begault. *3D Sound for Virtual Reality and Multimedia*. Academic Press Professional, 1994.

[2] I. Buck, T. Foley, D. Horn, J. Sugerman, and P. Hanrahan. Brook for GPUs: Stream computing on graphics hardware. *ACM Transactions on Graphics, Proceedings of SIGGRAPH 2004*, August 2004.

[3] J. Eyre and J. Bier. The evolution of DSP processors. *IEEE Signal Processing Magazine*, 2000. See also http://www.bdti.com/.

[4] T. Funkhouser, J.M. Jot, and N. Tsingos. Sounds good to me ! Computational sound for graphics, VR, and interactive systems. *SIGGRAPH 2002 course #45*, 2002.

[5] N. Tsingos, E. Gallo, and G. Drettakis. Perceptual audio rendering of complex virtual environments. *ACM Transactions on Graphics, Proceedings of SIGGRAPH 2004*, August 2004.

*{Emmanuel.Gallo|Nicolas.Tsingos}@sophia.inria.fr
http://www-sop.inria.fr/reves/projects/GPUAudio