Efficient Finite Difference-based Sound Synthesis Using GPUs

Marc Sosnick, Department of Computer Science Faculty Advisor: William Hsu, Department of Computer Science

San Francisco State University

INTRODUCTION

- We have been exploring the use of the general-purpose high-performance computing capabilities of GPUs to perform sound synthesis using compute-intensive physics-based models in realtime. Until now, realtime synthesis using these models has not been practical using only CPUs.
- Others have used these physics-based models generate audio[1,2], *but none have executed in realtime.*
- Realtime sound synthesis using these physics-based models will allow the creation of new audio synthesizer instruments.
- We discuss our findings from our proof-of-concept work, intended to find if it is possible to use these compute-intensive models to generate sound in realtime using GPUs.

SYNTHESIS METHOD

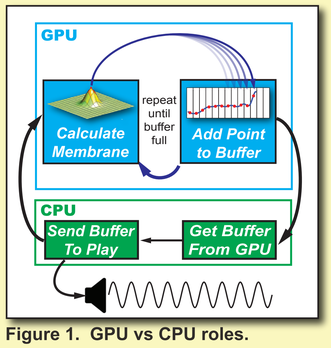


Figure 1. GPU vs CPU roles.

The CPU is used to coordinate buffers between the GPU and the audio driver (Figure 1). The audio sample buffer is filled by the GPU, and when full passed back to the CPU.

The GPU simulates a membrane in 3-dimensions, using the vertical displacement at a point on the membrane as the value for the audio sample (Figure 2). Equation (1) is repeated for each sample generated.

To simulate the membrane, we use a finite-difference scheme, using a truncated second-order Taylor expansion of the wave equation with dissipation in 2-dimensions[1,3,5]:

$$u_{i,j}^{n+1} = \left[1+\tfrac{\eta\Delta t}{2}\right]^{-1}\left\{\begin{array}{c}\rho\left[u_{i+1,j}^{n}+u_{i-1,j}^{n}+u_{i,j+1}^{n}+u_{i,j-1}^{n}-4u_{i,j}^{n}\right]\\ +2u_{i,j}^{n}-\left[1+\tfrac{\eta\Delta t}{2}\right]u_{i,j}^{n-1}\end{array}\right\} \quad (1)$$

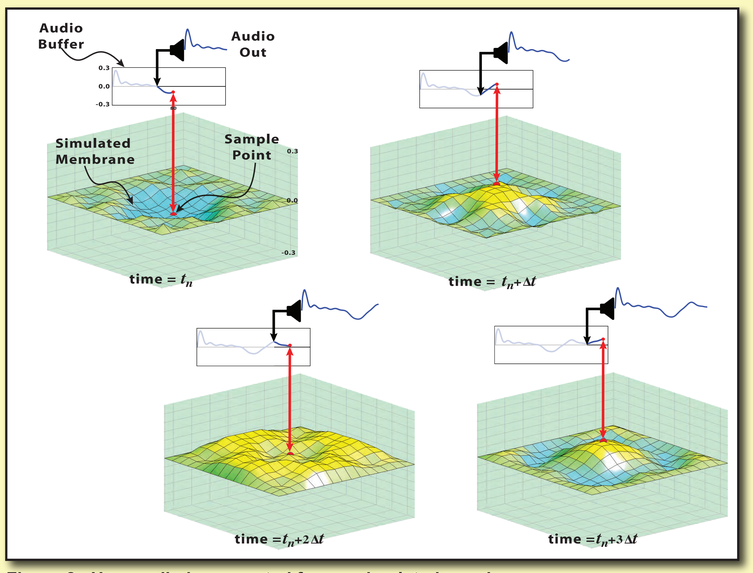
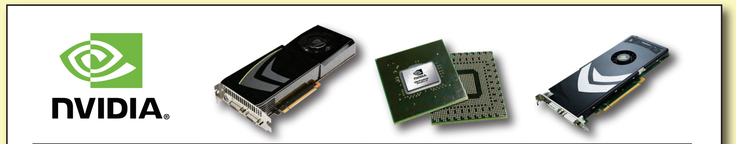


Figure 2. How audio is generated from a simulated membrane.

EXPERIMENTAL SETUP



Graphics Card	GTX285	9400M	8800GT
CPU @ Clock Rate	Intel Core 2 Quad @ 2.5 GHz	Intel Core Duo @ 1.86 GHz	Intel Quad Xeon @ 3 GHz
GPU Cores @ Clock Rate	240 cores @ 1.48 GHz	16 cores @ 0.80 GHz	112 cores @ 1.5 GHz

Table 1. System configurations tested

We implemented our software in C++ using Nvidia's CUDA[6] extension to program the GPUs. We tested our software on three different systems (Table 1), equipped with midrange graphics with GPU computing capability.

REQUIREMENTS FOR REALTIME

To be considered useful as a realtime instrument, jitter and latency must be within acceptable limits[4]. This is known as responsiveness. There can be no Jitter (Figure 3), which is usually caused by buffer underruns. Latency (Figure 4) should be below 30 ms.

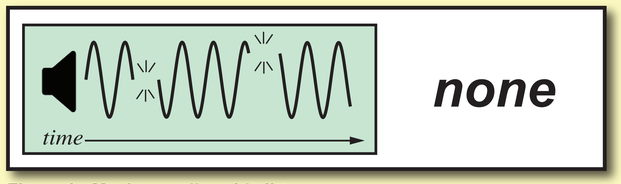


Figure 3. Maximum allowable jitter

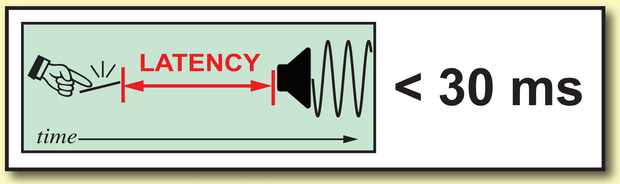


Figure 4. Maximum allowable latency

EXPERIMENTAL RESULTS

We timed execution on the CPU and GPU with a variety of buffer sizes and grid sizes (Figures 5, 6). Grid size is the resolution or size of the simulated membrane.

We checked for jitter, also using a variety of buffer and grid sizes. This is a binary test, where any buffer underrun error was considered jitter.

System	Processor	Configuration: Buffer (samples)	Configuration: Grid (points)
GTX285	CPU	≥ 4096	*
		*	≥ 20 x 20
	GPU	∅	∅
9400M	CPU	= 4096	*
		≤ 1024	*
	GPU	≤ 1024	*
8800GT	CPU	≤ 1024	= 21 x 21
	GPU	≤ 1024	= 21 x 21

Table 2. Results of jitter testing

EXPERIMENTAL RESULTS (CONT'D)

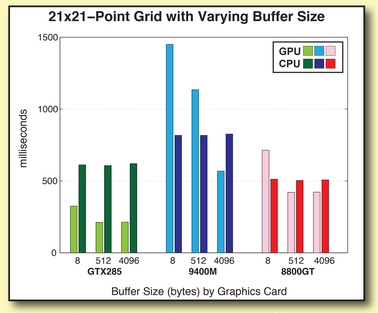


Figure 5. Results of varying buffer size

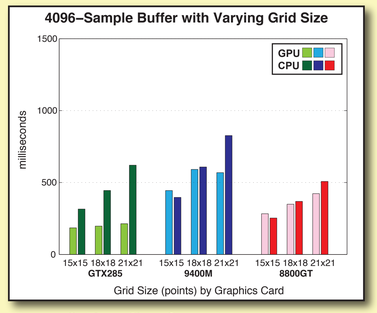


Figure 6. Results of varying grid size

CONCLUSIONS

- It is possible to generate realtime audio using GPUs and finite-difference simulations.
- Larger grids better leverage GPU computing power.
- Choice of buffer and grid sizes is important to responsiveness.
- Memory bandwidth is not a major consideration, especially with more advanced graphics cards.
- It should be possible to create a responsive, realtime synthesizer instrument using compute-intensive physics-based models.

FUTURE WORK

- Develop and optimize parallel algorithm to process arbitrarily large or dense grids.
- Write code in OpenCL to leverage heterogeneous computing environments and embrace industry standards.
- Package code into a modular, production-quality synthesis package.

REFERENCES

[1] A. Adib: "Study Notes on Numerical Solutions of the Wave Equation with the Finite Difference Method," *arXiv:physics/0009068v2 [physics.comp-ph]. 4 October 2000.* Downloaded from http://arxiv.org/abs/physics/0009068v2 on April 15, 2010.

[2] S. Bilbao: "A finite difference scheme for plate synthesis," *Proceedings of the International Computer Music Conference*, pp. 119-122, 2005.

[3] B. Land: "Finite difference drum/chime," Downloaded 4/15/2010 from http://instruct1.cit.cornell.edu/courses/ece576/LABS/f2009/lab4.html.

[4] N. P. Lago, F. Kon: "The Quest for Low Latency," *Proceedings of the International Computer Music Conference*, pp. 33-36, 2004.

[5] E. Motuk, R. Woods, S. Bilbao, J. McAllister: "Design Methodology for Real-Time FPGA-Based Sound Synthesis," *IEEE Transactions on Signal Processing*, Vol. 55, No. 12, pp. 5833 – 5845, 2007.

[6] *Nvidia CUDA Programming Guide, version 2.3.1. 8/26/2009.* Downloaded 4/21/2010 from http://developer.download.nvidia.com/compute/cuda/2_3/toolkit/docs/Nvidia_CUDA_Programming_Guide_2.3.pdf.

ACKNOWLEDGEMENTS

This research would not have been possible without generous support from William Hsu, Niki Jorgensen, Blair Whitmer, Dragutin Petkovic, Mike Wong, Carlo Matar, and Marcia and Myron Sosnick.