Finite Difference-Based Sound Synthesis Using GPUs

San Francisco State University

Marc Sosnick msosnick@sfsu.edu

William Hsu whsu@sfsu.edu

San Francisco State University, Department of Computer Science

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. While others have used these physics-based models to generate audio, *none have executed in realtime.* Realtime sound synthesis using these physics-based models will allow the creation of new audio synthesizer instruments. Our proof-of-concept project discussed here shows that it is possible to use these compute-intensive models to generate sound in realtime using GPUs.

GENERATING AUDIO FROM A SYNTHESIZED MEMBRANE

To simulate a membrane, we use a finite-difference scheme, using a truncated second-order Taylor series expansion of the wave equation with dissipation in two dimensions (Equation 1).

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

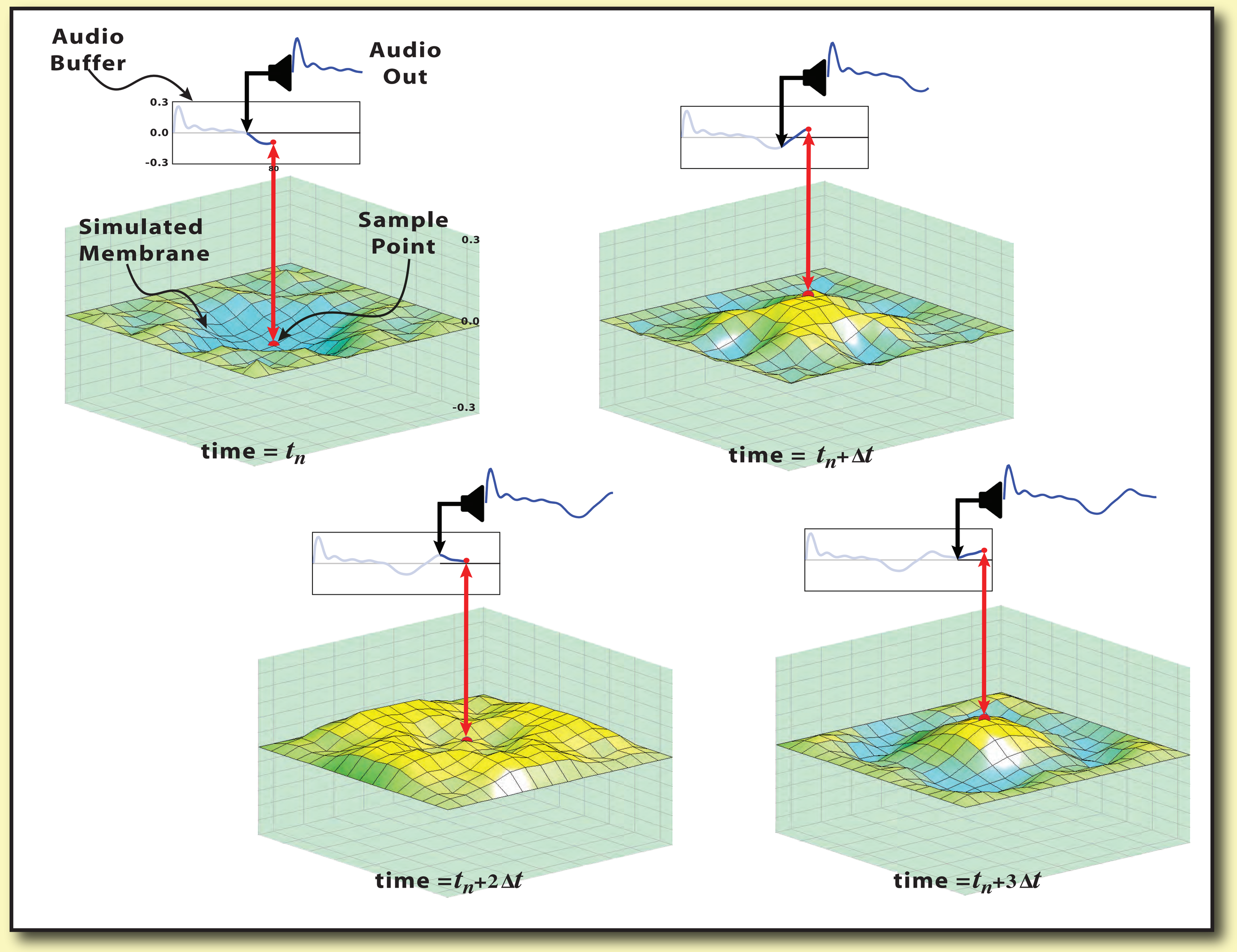

Figure 1. Generating audio from a simulated membrane.

A point is selected on the membrane (Figure 1). The vertical displacement of this point is captured at regular intervals over time. The change in vertical displacement of the membrane then corresponds to the vertical displacement over time of an audio signal. Therefore the motion of the membrane over time determines the audio output. For computational efficiency, audio output samples for consecutive timesteps are buffered before playback.

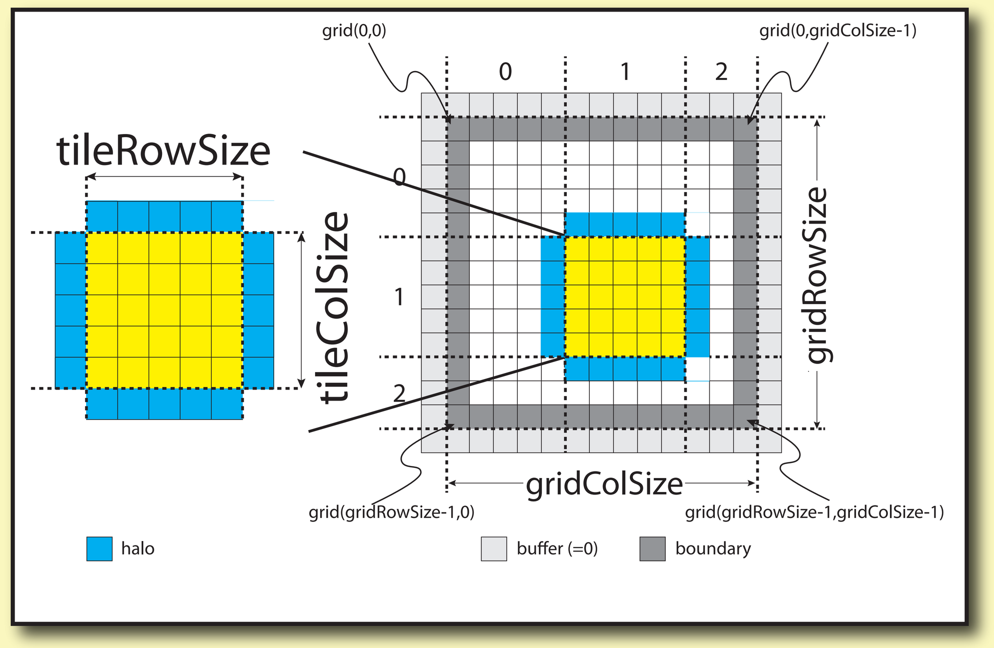

Figure 2. Membrane division for processing.

For each time step, one grid must be updated. To be processed efficiently by the GPU, the membrane (grid) must be divided into tiles, each of which can then be processed independently on the GPU (Figure 2).

SYSTEM ARCHITECTURE

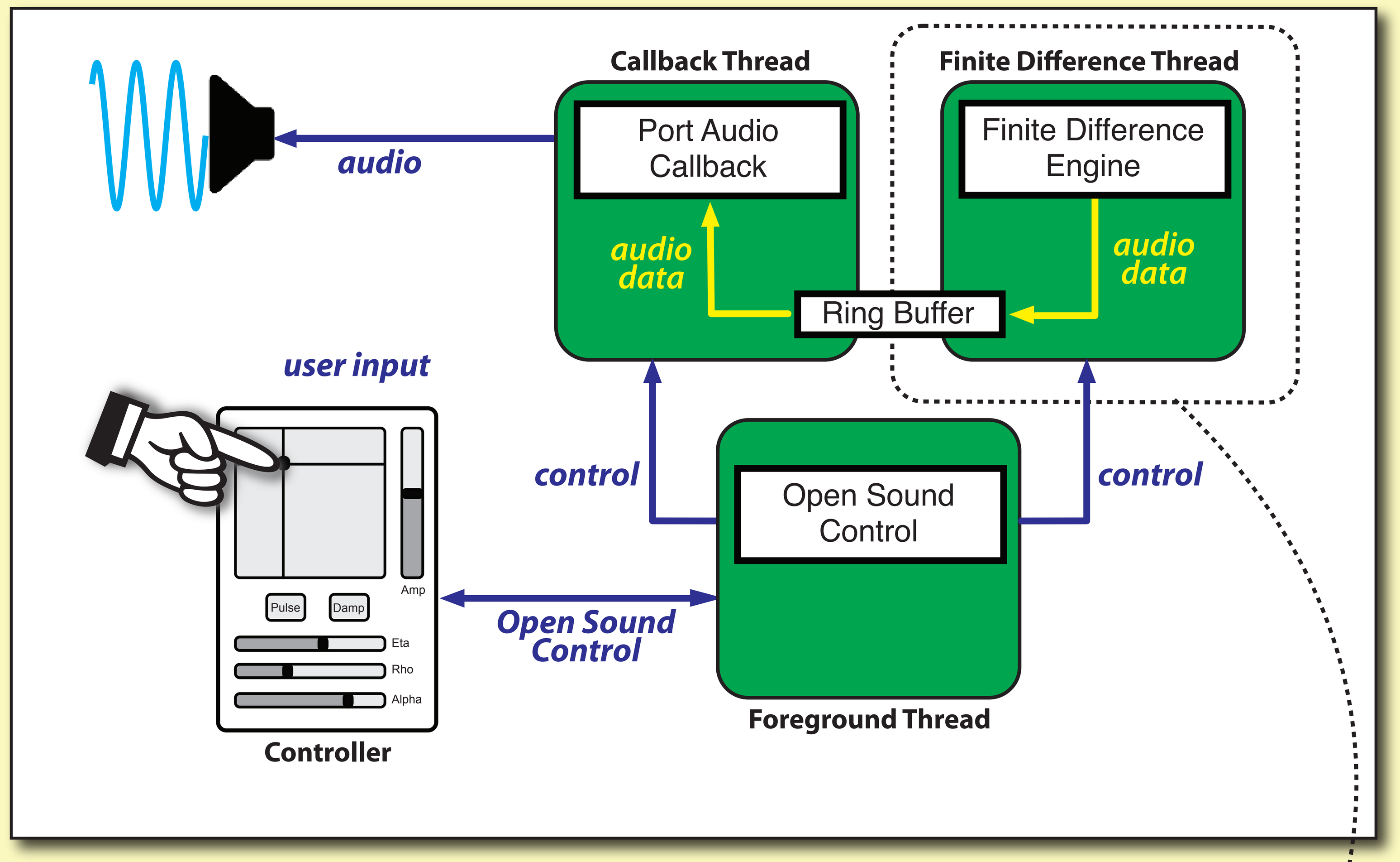

Figure 3. Host thread configuration.

Processing is divided between the host (CPU) and the GPU. For efficiency, The host runs three threads (Figure 3), each with distinct responsibilities. The Callback Thread collects and outputs audio samples, the Finite Difference Thread performs the finite difference simulation. The Foreground Thread handles controller input and program management.

The Finite Difference Engine (Figure 4) is responsible for coordinating membrane excitation, which roughly corresponds to plucking or striking the membrane, as well as continually simulating the vibrating membrane.

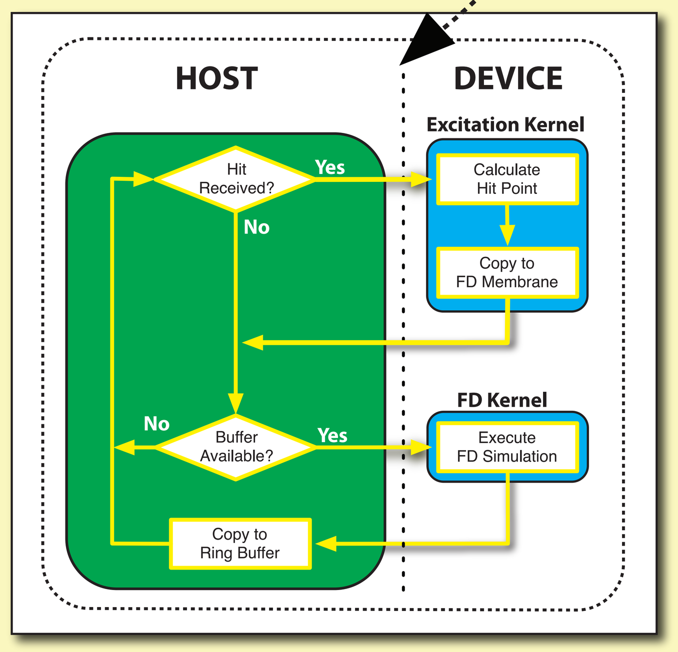

Figure 4. FD engine configuration.

EXPERIMENTAL SETUP

We implemented our Finite Difference Synthesizier (FDS) software in C and C++ using CUDA. We tested FDS on a MacBook Pro with a 2.66 GHz Intel Core i7, with a GeForce GT 330M GPU running OS 10.6.6.

	Grid Size (points)	Kernel Buffer Size (samples)
Setup I	21x21	8
	21x21	512
	21x21	4096
Setup II	15x15	4096
	18x18	4096
	21x21	4096

Table 1. Grid and buffer size configurations tested.

Table 1 shows the grid and kernel buffer size configurations that were tested. Setup I held grid size constant while varying the kernel buffer size. Setup II held the kernel buffer size constant while varying the grid size. We ran and timed the following sequence five hundred times:

1. run the excitation kernel
2. check ring buffer space
3. perform the FD simulation
4. copy the FD simulation output to the ring buffer.

The timings were then averaged. The results are presented in Table 2 and Table 3.

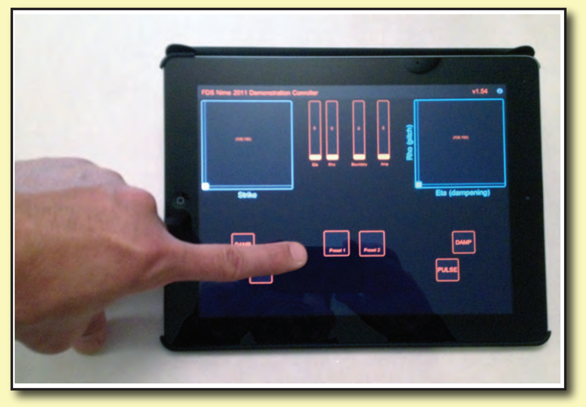

Figure 5. Experimental OSC controller interface.

REQUIREMENTS FOR REALTIME AUDIO

To be considered useful as a realtime audio instrument, jitter and latency must be within acceptable limits. This is known as responsiveness.

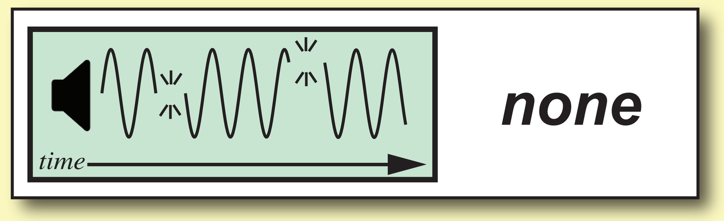

Figure 6. Maximum allowable jitter

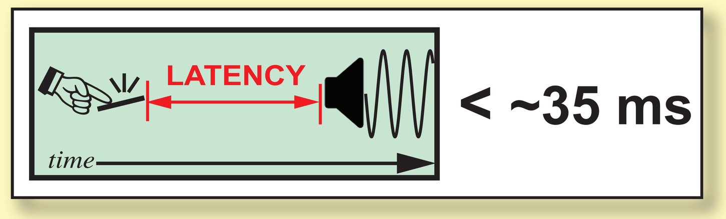

Figure 7. Maximum allowable latency

There can be no jitter (Figure 6), which is usually caused by buffer underruns, and latency (Figure 7) should be near or below 35ms.

EXPERIMENTAL RESULTS

Table 2 and Table 3 summarize our results. Buffer sizes of 8, 512, and 4096 samples correspond to audio output durations of 0.181 ms, 11.6 ms, and 92.8 ms at 44,100 Hz. For the realtime audio tests, all kernel output buffer and grid configurations produced no audio output buffer underruns.

Buffer Size (samples)	Excitation Time (ms)	Finite Difference Time (ms)	Memory Transfer Time (ms)	Total Time (ms)
8	0.04	0.56	0.02	0.62
512	0.03	6.78	0.01	6.82
4096	0.03	34.31	0.03	34.37

Table 2. Results of varying buffer size with a constant grid size of 21x21 points.

Grid Size (points)	Excitation Time (ms)	Finite Difference Time (ms)	Memory Transfer Time (ms)	Total Time (ms)
15x15	0.03	30.26	0.03	30.32
18x18	0.03	31.81	0.03	31.87
21x21	0.03	34.73	0.03	34.37

Table 3. Results of varying grid size with a constant buffer size of 4096 samples.

Satisfactory percussive sounds were produced in qualitative testing. An experimental Open Sound Control (OSC) controller interface (Figure 5) running on an Apple iPad2 was used as the user input controller. It was found that the FDS's output was especially sensitive to changes in the FD parameters η and ρ. Sample recordings of some of these tests are available at http://userwww.sfsu.edu/~whsu/FDGPU.

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.
- By using GPUs it is possible to create a responsive, realtime audio synthesizer instrument using compute-intensive physics-based models.