GPU-based Audio via the VGA Port

Jörn Loviscach*
Hochschule Bremen (University of Applied Sciences)

1 Introduction

Graphics processors catch increasing attention from the audio community [Trebien and Oliveira 2008]. The playback, synthesis, and processing of sound—in particular of a huge number of audio tracks playing simultaneously—is a task well suited for massive parallelization. However, handling audio on the GPU requires reading the resulting data from the graphics card and transporting them to the audio adapter. This causes state changes in the GPU and forces the CPU to wait for the completion of the read operation.

This work demonstrates that the detour via the audio adapter can be avoided. To this end, analog audio is sent through the graphics card's VGA port in an encoded form. An inexpensive electronic circuit built from standard operational amplifiers and CMOS logic ICs handles the decoding and outputs an analog stereo line signal. The graphics card is operated at the standard resolution of 800×600 pixels with typical settings for the *horizontal* blanks. The settings for the refresh frequency and for the *vertical* blank are special, however, as is detailed below. The triple buffer of the graphics card takes over the role of the ring buffer commonly used to ensure a continuous stream of data to the audio card. In contrast to the latter, the triple buffering is handled fully transparent to the programmer.

2 Implementation

The first major technical obstacle lies in the graphics card's DAC resolution of 8 bits per channel, which is poor when compared to the 16 bits of CD quality. This problem can be overcome through dithering: Every line of the VGA image is interpreted as one single audio sample. Within this line, the eight-bit value fluctuates slightly to yield an effectively higher resolution in the average. The second major technical obstacle is that today's typical graphics cards require a vertical blank interval of several lines to function reliably. Otherwise—as experiments with cards from different manufactures proved—the operating system may hang or the first lines of the image may randomly show the previous frame's content.

The inevitability of the vertical blank implies that not every line of the output can correspond to an audio sample. The audio decoder has to implement a buffer that is filled in advance and then can supply audio samples during the vertical blank. To keep this buffer short and the decoder electronics simple, one can introduce a regular pattern of lines with and without sample data as follows: The vertical blank (sync interval plus porches) is set to 40 lines, meaning that one frame comprises 640 lines. 128 audio samples are distributed onto these 640 lines as illustrated in Figure 1. To achieve a sampling rate of approximately 44,100 samples per second, the refresh rate is set to 344 frames per second.

The graphics card sends the audio samples in packets of 16. The decoder spread each of these packets evenly over 80 lines of the image, employing an analog storage based on 16-line (de-)multiplexer circuits. The red and the blue channel of the VGA connector are used for the left and right audio channels, respectively. The green channel carries a signal that controls the timing of the analog buffer and is used as a reference to compensate noise. For details see the electronic attachment to this page.

*e-mail: joern.loviscach@hs-bremen.de

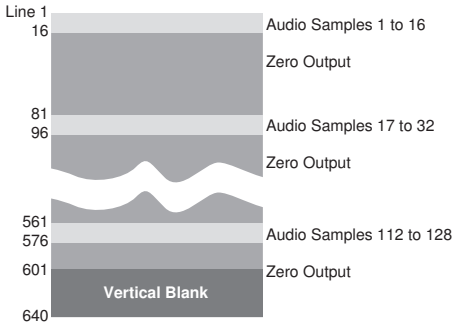

Figure 1: *A frame of 800×600 pixels contains 128 audio samples.*

To create the appropriate signal on the graphics card, the software renders a frame-filling rectangle with a custom pixel shader. This shader reads the audio data from a two-channel texture with 16 bit resolution per channel. The high-frequency noise pattern for the dithering is supplied through another texture. Electrical differences between the 16 storage places of the analog buffer and cross-talk from the digital control lines inevitably lead to tonal noise at one 16th of the sampling frequency, slightly less than 3 kHz. This noise can be suppressed by adding a canceling waveform to the input signal. This waveform is composed of sinusoidal waves at this frequency and integer multiples of it. Setting the amplitudes and phases of these waves is interactive and has to happen only once.

3 Results and Outlook

Experiments were done on the VGA port of an Nvidia GeForce 6800GT with a standard display connected to the primary DVI output for operation. The frequency response of the system extends from 20 Hz to 20 kHz within ±3 dB. The added noise-canceling waveform widens the dynamic range from 49 dB to 66 dB. The latency time measured as reaction time to an impulse sent via a handshake pin of the serial port is 10 ms, as expected for triple-buffer operation at 344 fps. Nvidia's performance counter pixel_shader_busy is at 35 %, which is mostly due to the low number of pixel processors; all other GPU load indicators are below 5 %.

Future work will address using a DVI port as a means to output audio. This digital output allows the decoder to assemble the bits of the three color channels into one 24-bit word per pixel, hence achieving studio-grade resolution without sacrificing one complete line of the image for every audio sample. The DVI-based solution requires significantly more complex decoding logic, but may easily offer 1000 audio outputs in parallel, a number that is useful for wave field synthesis. Such applications that generate massive amounts of data on the graphics card are particularly promising, as the direct output means than only small volumes of data have to be transferred between the CPU and the GPU—in either direction.

References

TREBIEN, F., AND OLIVEIRA, M. M. 2008. Real-time audio processing on the GPU. In *ShaderX 6: Advanced Rendering Techniques*, W. Engel, Ed. Charles River Media, 583–604.