Individual Project Report

Matthew Reuben Thomas U1154964

Introduction

The goal of this project was to produce a piece of software which could efficiently synthesize high-quality impulse responses for virtual environments. The software had to run on commodity hardware, available to most musicians, while also producing high-quality output at a reasonable speed. In addition, the software had to be capable of rendering impulses suitable for basic auralization, using HRTF-based techniques.

Literature Review

Many different implementations of modelling reverbs exist. Some of these are commercially available, for example the Odeon family of room acoustics software (discussed in Rindel (2000)), and the Catt-Acoustic program (Dalenbäck (2010)). Many more exist only as research projects without commerical implementations - the programs described by Röber, Kaminski, & Masuch (2007), Savioja, Lokki, & Huopaniemi (2002), Schissler et al. (2014) and Taylor et al. (2012) (to name a few) do not appear to be available in any form to the general public.

Physically-based reverb algorithms can largely be divided into two categories, namely wave-based and geometrical algorithms, each of which can be divided into subcategories.

Savioja et al. (2002) notes that wave-based methods include the *finite element method*, *boundary element method*, and the *finite-difference time-domain model*. These methods produce very accurate results at specific frequencies, however they are often very costly. In these models, the scene being modeled is divided into a mesh, where the mesh density is dictated by sampling frequency. Calculations must be carried out for each node on the mesh, as a result of which, computational load increases as a function of sampling frequency. This in turn makes higher frequencies very computationally expensive to calculate.

Geometric methods may be stochastic or deterministic, and are usually based on some kind of ray-casting. In these models, sound is supposed to act as a ray rather than a wave. This representation is well suited to high frequency sounds, but is unable to take the sound wavelength into account, often leading to overly accurate higher-order reflections (Rindel (2000)). These methods also often ignore wave effects such as interference or diffusion.

The most common stochastic method is ray tracing, which has two main advantages: Firstly, an implementation can take influences from the extensive body of research into graphical raytracing methods. Secondly, raytracing is an 'embarrassingly parallel' algorithm, meaning that it can easily be distributed across many processors simultaneously, as there is no need for signalling between algorithm instances, and there is only a single 'final gather' (Stephens (2013)).

A common deterministic method is the image-source method. This algorithm is conceptually very simple, and therefore fairly straightforward to implement. However, it quickly becomes very expensive, with complexity proportional to the number of primitives in the scene raised to the power of the number of reflections. For scenes of a reasonable size, a great number of calculations are required, making soley image-source based methods impractical for most purposes.

Many systems implement 'hybrid' algorithms, which may combine any of the above methods. Combining wave-based and geometric methods has an obvious benefit - the wave-based simulation can be used to generate a low-frequency response, while a geometric method can generate the higher-frequency portion. Similarly, combining ray tracing and image-source modelling allows for the use of accurate image-source simulation for just the early reflections, with faster stochastic modelling of later reflections.

The majority of the systems mentioned are realtime - the programs mentioned by Röber et al. (2007), Schröder (2011), Schissler et al. (2014), Savioja et al. (2002), and Taylor et al. (2012), along with the Catt-Acoustic software, are all capable of producing real-time auralizations. Offline renderers also exist. The Odeon software and the system described by B. Kapralos, Jenkin, & Milios (2004) are both examples of offline renderers.

Realtime systems play an important role in audio engines for interactive games or virtual-reality systems, where it is often necessary for virtual environments to provide realistic sonic cues to the interacting user. However, such systems must be capable of producing impulse responses very rapidly, at least tens of times per second. Savioja et al. (2002) and B. Kapralos et al. (2004) note that it is generally only possible to use geometric methods to simulate early reflections at real-time rates, as modelling later reflections becomes too computationally expensive. Schissler et al. (2014) mentions that in benchmarks for their real-time system, they traced 1000 rays from each source to an arbitrary

depth of around 10 reflections. Cheng (2014) describes a real-time geometric system for modelling sound in 3D games, which is capable of rendering 4096 rays to a depth of 3 reflections in real-time.

In raytracing, as with any other stochastic modelling method, better quality is achieved primarily through taking more samples. Clearly, the constraints that real-time processing places upon auralization methods mean that impulses generated in this way will be inaccurate, and likely of insufficient quality for music production.

The final notable difference between implementations is to do with hardware rather than software. Many modern implementations of raytracing algorithms take advantage of the computer's graphics processing unit (GPU), which provides a huge number of discrete processors that can run the same algorithm in parallel. Though the processors are individually much less powerful than the typical central processing unit (CPU), great speedups are possible due to the sheer number of processes that can be run simultaneously. Raytracing algorithms have the added advantage that most of the data supplied to the algorithm is 'static' - all the information about the scene, such as geometry, source position, mic position, and materials can be reused for each ray that is fired. The only variant parameter is the initial ray direction. This means that the static data can be copied to graphics memory once, and then referred to by every raytracer instance. Due to the popularity of raytracing techniques in graphics, a variety of papers are available concerning the implementation of raytracers on graphics hardware.

On the other hand, many raytracing implementations just run on the CPU. It is still possible to achieve reasonable speedups over naive implementations by using single-instruction multiple-data (SIMD) instructions or multiple threads. Advantages of this approach are mostly to do with tooling. GPU code must be written in a speciality language - 'general purpose' options include Cuda and OpenCL, though Cuda code will only run on Nvidia GPUs. Alternatively, a graphics shader language such as GLSL can be coerced into interacting with non-graphical data. These speciality languages often have very sparse programming environments, and much better tools are available for working with and debugging CPU code than GPU code. Additionally, every platform is guaranteed to have a CPU on which the program can run, but not every system will have a compatible GPU.

Implementation

For this project, I opted to build an offline raytracing algorithm, which I later adapted into a hybrid raytracing/image-source model. I decided to keep the costliest and 'most parallel' parts of the algorithm on the GPU where possible, doing only mixdown and postprocessing on the CPU.

Evaluation

Bibliography

Antani, L., Chandak, A., Savioja, L., & Manocha, D. (2012). Interactive sound propagation using compact acoustic transfer operators. *ACM Transactions on Graphics*, *31*(7).

Battenberg, E., & Avižienis, R. (2011, September). Implementing real-time partitioned convolution algorithms on conventional operating systems. Proceedings of the 14th International Conference on Digital Audio Effects (DAFx-11).

Belloch, J. A., Ferrer, M., Gonzalez, A., Martínez-Zaldívar, F. J., & Vidal, A. M. (2012). Headphone-based spatial sound with a gpu accelerator. Proceedings of the International Conference on Computer Science.

Belloch, J. A., Gonzalez, A., Martínez-Zaldívar, F. J., & Vidal, A. M. (2011). Real-time massive convolution for audio applications on the gpu. *The Journal of Supercomputing*, *58*(3).

Cheng, Z. (2014). Making games sound as good as they look: Real-time geometric acoustics on the gPU. Retrieved from http://on-demand.gputechconf.com/gtc/2014/presentations/S4537-rt-geometric-acoustics-games-gpu.pdf.

Cowan, B., & Kapralos, B. (2008, November). Spatial sound for video games and virtual environments utilizing real-time gpu-based convolution. Proceedings of the 2008 Conference on Future Play: Research, Play, Share.

Cowan, B., & Kapralos, B. (2009). Real-time gpu-based convolution: A follow-up. Proceedings of the 2009 Conference on Future Play on @ GDC Canada.

Dalenbäck, B.-I. (2010, May). Engineering principles and techniques in room acoustics prediction. Proceedings of the 2010 Baltic-Nordic Acoustics Meeting.

Dammertz, H., Hanika, J., & Keller, A. (2008). Shallow bounding volume hierarchies for fast simd ray tracing of incoherent rays. Proceedings of the Nineteenth Eurographics conference on Rendering.

Gaster, B. R. (2010). The opencl c++ wrapper api. Retrieved from https://www.khronos.org/registry/cl/specs/opencl-cplusplus-1.1.pdf

Goldsmith, J., & Salmon, J. (1987). Automatic creation of object hierarchies for ray tracing. *IEEE Computer Graphics and Applications*, *7*(5).

Hradský, P. (2011, January). *Utilising opencl framework for ray-tracing acceleration* (Master's thesis). Czech Technical University in Prague.

Hulusic, V., Harvey, C., Debattista, K., Tsingos, N., Walker, S., Howard, D., & Chalmers, A. (2012). Acoustic rendering and auditory-visual cross-modal

perception and interaction. *Computer Graphics Forum*, *31*(1).

Kapralos, B., Jenkin, M., & Milios, E. (2004, October). Sonel mapping: Acoustic modeling utilizing an acoustic version of photon mapping. Proceedings of the 3rd IEEE International Workshop on Haptic, Audio and Visual Environments and Their Applications.

Müller-Tomfelde, C. (2001). Time-varying filter in non-uniform block convolution. Proceedings of the COST G-6 Conference on Digital Audio Effects (DAFX-01).

Noisternig, M., Katz, B., Siltanen, S., & Savioja, L. (2008a). Framework for real-time auralization in architectural acoustics. *Acta Acustica United with Acustica*, *94*(6).

Noisternig, M., Katz, B., Siltanen, S., & Savioja, L. (2008b). Framework for real-time auralization in architectural acoustics. Retrieved from http://auralization.tkk.fi/Auralization_framework

Nvidia. (2009). Opencl programming guide for the cuda architecture. Retrieved from http://www.nvidia.com/content/cudazone/download/OpenCL/NVIDIA_OpenCL_ProgrammingGuide.pdf

Raghuvanshi, N., Narain, R., & Lin, M. C. (2009). Efficient and accurate sound propagation using adaptive rectangular decomposition. *IEEE Transactions on Visualization and Computer Graphics*, *15*(5).

Rindel, J. (2000). The use of computer modeling in room acoustics. *Journal of Vibroengineering*, *4*(3).

Röber, N., Kaminski, U., & Masuch, M. (2007, September). Ray acoustics using computer graphics technology. Proceedings of the 10th International Converence on Digital Audio Effects.

Savioja, L., Lokki, T., & Huopaniemi, J. (2002). Interactive room acoustic rendering in real time. Proceedings of the 2002 IEEE International Conference on Multimedia and Expo.

Schissler, C., Mehra, R., & Manocha, D. (2014). High-order diffraction and diffuse reflections for interactive sound propagation in large environments. *ACM Transactions on Graphics*, *33*(4).

Schröder, D. (2011, February). *Physically based real-time auralization of interactive virtual environments* (PhD thesis). RWTH Aachen University.

Stephens, R. (2013). *Essential algorithms: A practical approach to computer algorithms*. John Wiley & Sons.

Taylor, M., Chandak, A., Mo, Q., Lauterbach, C., Schissler, C., & Manocha, D. (2012). Guided multiview ray tracing for fast auralization. *IEEE Transactions on Visualization and Computer Graphics*, *18*(11).

Tsakok, J. A. (2009, August). Faster incoherent rays: Multi-bVH ray stream tracing. Proceedings of the Conference on High Performance Graphics 2009.

Wei, C., Gain, J., & Marais, P. (2012, March). Interactive gpu-based octree generation and traversal. Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games Costa Mesa, CA March 9 - 11, 2012.