Designing a Dedicated Audio Processing Unit

ETABA ASSIGANA[*], *AES Student Member*
(ea6088a@american.edu)

American University, Washington D.C., USA

Modern video games contain thousands of individual sound files: music, sound effects, ambiences, and dialog. Most of these are processed and manipulated in some way during playback and it takes a good amount of CPU (central processing unit) headroom to handle all of that processing, even with the power CPUs offer today. I believe that creation of a successful APU that could calculate sonic transformations of sounds based on situational properties such as travel medium, obstructions/occlusions, travel distance, environmental absorption rates, temperature, refraction, and other acoustic properties could be the next frontier in immersive entertainment, for gaming and beyond. This paper assesses several components of realistic acoustic modeling, recent methods and research regarding their implementation, and seeks to determine the feasibility of combining them into one acoustic modeling system that could be implemented onto a discrete audio processing unit chip(set).

0 INTRODUCTION

Accurate acoustic simulation is a complex and computationally-expensive task that essentially involves multiple recursions through calculations that derive snapshots of sound wave propagation originating from a source sound, and convolutions of them into aural content that can be played back. Channel-based "two-dimensional" sounds notwithstanding, the current engineering paradigm for "three-dimensional" audio processing and playback in gaming is an object-based acoustic environment simulation based on simplified propagation properties that can be more efficiently computed as part of the shared processing of a single central processing unit architecture.

But what if processing power wasn't restricted by the need to share it? As an example, consider the graphics processing unit (GPU). It arose out of a need to offload CPU cycles necessary to display increasingly more complex visuals in games. The creation of custom graphics acceleration hardware dates back to the 1970's; though the term GPU didn't arrive until the 80's. But it wasn't until 1999, the dawn of a new millennium, that Nvidia's GeForce 256 kicked off a graphical revolution in gaming, being marketed as "the world's first GPU" [1]. Since then, graphical computation has exploded by orders of magnitude. Additionally, people have expanded the concept of the GPU into the GPGPU, general purpose graphics processing unit, which allows units to perform CPU computations that aren't necessarily graphical in nature.

0.1 Question

Research into using GPGPUs for audio processing and simulation is an ongoing pursuit and will be discussed in this paper; however, a fundamental limitation of GPGPU pipelines is that they require data migration into a graphical form in order to process said data [2]. Thus, the question I ask is: can the idea of parallel processing as developed for (GP)GPUs be translated into the creation of a new, custom audio processing unit?

1 MOTIVATION

My interest in this topic stems from my interest in audio for video games: music, sound design, and implementation. With the nascent proliferation of virtual and augmented reality (VR and AR, respectively), games are increasingly seeking to encapsulate players entirely into their virtual worlds. Sound is a big part of what some define as "presence", as lead PlayStation™ console architect Mark Cerny can attest to. In fact, it's important enough that Sony is working with Advanced Micro Devices (AMD) on a custom chip from their Ryzen™ series APUs[1] that will implement acoustic ray tracing and 3D spatial audio [4].

As advanced as the PlayStation™ "5" (presumed title) will be, the gap remains in the accurate rendering of sound versus the traditional interpretation of it in games. This can be further generalized to other interactive and non-interactive media, both linear and nonlinear. As an ex-

[*]To whom correspondence should be addressed e-mail: eaba@live.com.

[1]AMD's *APU* is a term for what they call an accelerated processing unit, consisting of a CPU and GPU on a single die [3].

ample, consider the horror genre of film, television, and video games. Offering fans of the genre the opportunity to immerse themselves in a 360-degree virtual reality experience with the realistic acoustics of the horrific environments contained within them—especially near-field acoustics—would offer a revolution in virtual presence. Notable and legendary games such as *Hellblade: Senua's Sacrifice*, *Dead Space*, *Resident Evil*, *Silent Hills*, and so many others would take on a whole new dimension of impact beyond their visuals.

Beyond visual entertainment media, realistic real-time acoustic simulation can be applied to many other fields. Music is an example. We could simulate and recreate live concerts in venues that could never contain the number of listeners who may want to attend a given show. An audio engineer could record a studio performance of an artist and apply the acoustics of any desired venue using their geometric and environmental properties instead of simulating purely from pre-recorded impulse responses (of which an infinite amount would need to be captured to truthfully reproduce an environment). Combined with binaural filtering and head tracking, a performance of a concert that never actually occurred could be achieved.

As an avid consumer of video games, particularly on consoles, I have lived and played long enough to experience the evolution from 8-bit blips and beeps over monophonic outputs into 16-bit stereo sound, to now experiencing uncompressed 5.1 surround sound and binaural sound. I would like to see an ultimate goal of environmental sound propagation being completely recreated in virtual simulations (unrealistic environments notwithstanding). To that end, I propose one potential method for realizing that goal.

2 LITERATURE REVIEW

2.1 The First APU

The concept of an audio processing unit is not novel in the sense that is not completely original. NVIDIA Corporation, an American technology company that creates GPUs and systems-on-a-chip, is the first—and as far as I could find, only—company to create and designate an audio processing unit. NVIDIA's SoundStorm was initially funded by Microsoft for use in their original Xbox console.

At its core, [NVIDIA's] Audio Processing Unit is a multi-processor audio rendering engine. The APU is responsible for providing hardware audio acceleration for both output streams and input streams and renders completely to system memory. The APU is divided into four main sections:

- Setup Engine — This unit is responsible for performing all data and parameter setup for the other processors. All memory management, mapping and DMA resources are controlled in this unit.
- Voice Processor — This unit contains several fixed function digital signal processing (DSP) units responsible for processing voices and mixing the results in the mixer buffers.
- Global Processor — This unit is built around a programmable DSP. The DSP is responsible for adding varied effects to the data in the mixer buffers and producing the final output stream to the OS.
- Dolby Digital Interactive Content Encoder — This unit is built around a programmable DSP, which is responsible for encoding Dolby Digital (AC-3) data that'll be sent over the SPDIF to an external consumer decoder. This allows 5.1 speakers (left front, right front, center, right rear, left rear, sub-woofer) to be transmitted over a digital interface. [5]

Ultimately, NVIDIA decided that *SoundStorm* was too expensive to use on their nForce chips and discontinued the SIP block after the nForce 1 and 2 [6]. But the idea of a GPU for audio did not die with that product.

2.2 Use of the GPU for Audio

Lauri Savioja, Vesa Välimäki, and Julius O. Smith of Aalto University in Finland and Stanford University in California are one of many groups of researchers that have studied the use of graphics processing units for audio signal processing. In one 2011 study, they compared audio digital signal processing applications on a GPU by looking at cases of additive synthesis, Fourier transformations and convolutions in the frequency domain, and finite impulse response filtering in the time domain. They concluded that GPUs are well suited for audio applications that need heavy computation, can be parallelized, and can tolerate some latency; however, they noted that not all possible tasks are suitable for the parallelization that a GPU is designed for [7].

This study followed one they did in the previous year where they looked at real-time additive synthesis using a GPU. They created a simple additive synthesis algorithm in order to find out how many sinusoids they could produce and play in real-time on both a CPU and a GPU. The first case concentrated only on pure sinusoid computation at random frequencies. The second case defined starting phase and gain for each wave. Their results showed that a parallel implementation of additive synthesis on a GPU offers over 1300 times the performance of a serial table-lookup implementation on CPU, and over 4400 fold gain over the function call version on a CPU with modest buffer sizes [8]. Note that they conducted this experiment on what is now a 9-year-old GPU (the NVIDIA GeForce GTX480). Considering that they were able to produce almost 2 million concurrent sine waves with the chips they had, the speed improvements and core count increases on GPUs since then could yield much greater results today.

Savioja, Välimäki, and Smith were not alone in their research. The same year as their additive synthesis study, Nicolas Tsingos of the Sound Technology Research department at Dolby Laboratories in San Francisco was conducting his own study of 3D audio processing and sound scattering simulations on a GPU with Wenyu Jiang and Ian Williams. Their paper explains the differences in architecture and function of CPUs and GPUs, then analyzes the ways GPUs can be utilized for audio processing and au-

ralization. They consider bottlenecks in GPU performance (with regard to acoustical applications) and analyze methods of ray casting, scattering, and occlusion through geometry processing and rasterization [9].

This analysis was essentially a republication of a paper he wrote 2 years prior as an individual [10] while conducting a concurrent study on geometric acoustical modeling in game environments. In his paper, he discusses how pre-computing image-source gradients for early reflections and directional decay profiles allows location dependent reverberation effects to be generated without storing or accessing the actual geometry at run-time. The pipeline described enables fine-grain rendering of distance and surface proximity effects and modeling of both outdoor and coupled indoor spaces with arbitrary reverberation decay profiles [11]. This is one of many components to consider when crafting an APU.

Meanwhile, in the University of Applied Sciences and Arts of Southern Switzerland, Tiziano Leidi, Thierry Heeb, Marco Colla, and Jean-Philipp Thiran were completing their own analysis of real-time audio processing applications for GPGPUs. In their 2011 paper, they consider the issue of insufficient data pressure in the GPGPU pipeline (leading to inactive GPU cores) and propose an event-driven scheduling method that maximizes data pressure, thereby increasing performance [12].

2.3 Defining the Ideal APU

2.3.1 Architecture

All of my previous sources (as well as others) discuss, analyze, and conclude that the nature of parallel computing offered by graphics processing units (general purpose or not) offer performance improvements in audio processing and even synthesis. This strengthens my supposition that the concept of the GPU can be translated into a dedicated audio processing unit; however, in order to do so, I must consider why a dedicated unit is more desirable than continued research into tailoring and optimizing (GP)GPUs for audio modeling use.

I propose the primary reason for an APU is to eliminate what I see as a fundamental bottleneck of (current) GPU usage: the need to convert audio data into graphical data. Traditionally, "graphics hardware has a specific data flow computational model. Its architecture is originally targeted at manipulating 3D primitives such as points, lines or polygons, performing raster graphics operations and rendering the result to the screen. Processing audio through a graphics APIs[sic] involves converting the audio signal into textures and processing lines or polygons textured with the audio data" [10].

For geometric calculations such as sound occlusion or reflection, this structure is well suited [10]; however, it isn't necessarily favorable all aspects of audio processing. it was only with the introductions of languages like NVIDIA's CUDA and Khronos Group's openCL that opened the GPGPU door of allowing developers to access the GPU as a stream processor without going through a graphics-oriented rendering pipeline [2]. In light of this, the APU I propose would not necessarily need hardware that different from current (high-end) GPU chips. Its most crucial component would be in its processing of commands with regards to audio rendering.

According to Leidi, Heeb, Colla, and Thiran, "audio processing applications are often computational[sic] intensive with stringent requirements in terms of low processing latency. Audio signals are often processed on-the-fly and interactively, by streaming audio samples through networks of filters and other software components. Loading the complete signals into memory may prove inefficient, illogical or simply impossible, because audio streams may be produced by unpredictable sources or may be huge and should, ideally, be processed instantaneously. Therefore, realtime processing of audio streams is usually done by applying the required software components, directly as samples are acquired, loaded or synthesised[sic]. Additionally, the user may occasionally need to interact with the application at run-time and most user changes have to take effect immediately" [12].

Furthermore, "to speed-up audio processing applications, it is possible to profit from GPGPUs by processing audio signals in parallel on the available hardware resources. One of the effective solutions is to process more data in parallel by means of data decomposition approaches. The throughput consequently rises and leaves a buffer zone to improve the quality or quantity of processing. Fine-grain data decomposition and execution of filters on GPGPUs is very effective; however, enhancing an audio processing application for execution on a combination of CPU and GPGPU can prove difficult. Bottlenecks filters[sic] and serial dependencies between or within the filters' algorithms may compromise performance of the initially foreseen parallelism. Delay lines or IIR filter states are examples of the most typical obstacles. To avoid corrupting internal states, serial processing of audio signals may be required" [12].

All of this is to say that an APU can theoretically solve the problem of variability in computing needs for audio processing if built as an integrated chip that effectively has both CPU and GPU-like components to perform audio computation appropriate to each respective architecture. Coupled with an original, dedicated, high level language like CUDA or OpenCL, the chip would need to execute a form of audio deconstruction for parallel processing and serial processing (depending on what was needed for any given transformation), and return a complete result in real-time. Integrated chips already exist and in fact, AMD's aforementioned Advanced Processing Unit seems to be an accurate approximation of the hardware design of an Audio Processing Unit.

2.3.2 Acoustic Computation

In the following section, I will detail and analyze many different computational methods for acoustic modeling that researchers have studied in order to determine the desired capabilities of an APU (assuming an integrated CPU-GPGPU architecture). I will examine geometric meth-

ods (incorporating techniques like ray casting), as well as wave-based methods in order to come up with a comprehensive list of APU functions.

In the years since it released in 2007, NVIDIA's CUDA architecture has been expanded with various libraries that take advantage of its GPGPU function. One of these is *cuFFT*. The NVIDIA CUDA Fast Fourier Transform library provides GPU-accelerated FFT implementations that perform up to 10x faster than CPU-only alternatives [13]. Fast Fourier Transformations are a fundamental part of digital signal processing; and thus are an important part of GPGPU acoustical applications. The very year after CUDA's debut, Naga K. Govindaraju, Brandon Lloyd, Yuri Dotsenko, Burton Smith, and John Manferdelli of Microsoft Corporation presented several algorithms for efficiently performing FFTs of arbitrary length and dimension on GPUs using the CUDA interface. Their results indicated a significant performance improvement over optimized GPU-based and CPU-based FFT algorithms [14]. Work on FFTs expanded beyond NVIDIA's hardware by researchers such as Franz Franchetti, Markus Puschel, Yevgen Voronenko, Srinivas Chellappa, and Jose M. F. Moura, who wrote a 2009 article giving an overview on the techniques needed to implement discrete Fourier transform efficiently on (mostly Intel-based) GPUs [15].

In 2010, Jamie Angus and Andrew Caunce conducted research on improving the efficiency and accuracy of finite difference time domain acoustic simulation. The spectral methods they described and implemented into CUDA demonstrate how FFT is particularly suited to GPU architecture. Spectral methods for solving differential equations involve using the Fourier transform to perform the differentiation. This is achieved by taking the impulse response of a perfect differentiator and convolving it with the input data. Their FDTD calculation method resulted in an increase in accuracy as well as a reduction in computational expense (and without the benefit of time to optimize their code for efficiency) [16]. In general, the FDTD method is used for computational electrodynamics, rather than acoustical modeling; however, its nature of being used for impulse response calculations has clear potential for audio applications.

That same year, Alexander Southern, Damian Murphy, Guilherme Campos, and Paulo Dias conducted their own research into a wave-based method for finite difference room acoustic modeling on a GPGPU. They noted that "FDTD models are well established for the purpose of approximately simulating the wave propagation of sound throughout an enclosed space over a period of time". In their paper, they discussed their GPU CUDA implementation of a 2D FDTD acoustical model for generating RIR responses of arbitrary room geometries. They demonstrated significant accelerations for numerous GPUs when compared to a typical 2D FDTD CPU implementation. They claimed that their results further re-enforced the suitability of the wave-based techniques for generating RIR datasets for detailed walkthrough auralizations [17].

Their research was encouraging; however, there are challenges with the FDTD method that were identified and addressed in later research by Jukka Saarelma, Jonathan Califa, and Ravish Mehra. In 2018, they looked at challenges of distributed real-time finite-difference time-domain room acoustic simulation for auralization through three experiments. In their paper, they noted that "large-scale wave-based room acoustic simulations are commonly considered computationally expensive. Distributed computing can be used to accommodate such methods for wide-bandwidth simulation, but may pose several challenges for real-time execution, such as communication and synchronization latencies between the computing hardware". To solve such issues, they proposed a method of partitioning rooms onto multiple devices (compute nodes consisting of eight GPGPU devices), overlapping the parts, and communicating communicating the parts after each simulation step [18].

"From the results of [their] first experiment, it can be deduced that usage of multiple compute nodes is very beneficial for large domain sizes. For small domain sizes, the execution of the simulation is more efficient on a single compute node with the current implementation". "From the results of [their] second experiment, it can be deduced that for small domain sizes and multiple GPGPU devices the synchronization of devices becomes a prominent factor for the execution time. When the domain size is reduced, the actual simulation update using high performance GPGPUs becomes very small and the synchronization takes proportionally more time. Additionally, sub-optimal load balancing increases the processing time". This point was previously discussed by Leidi et al [12]. Lastly, "the third experiment [gave] a rough overview to the possibility of real-time FDTD simulation for auralization on VR/AR device using distributed computing. The solver used in the experiment can achieve moderate performance with a limited bandwidth simulation" [18].

Fortunately, I will note that for an APU that exists on a single die, latency and synchronization times are less of a concern with regard to operation (since electrical signals won't be traveling across wires in a mesh). That fact is also a boon for using the APU as a DSP accelerator GPGPU, as described by Nicholas Jillings and Yonghao Wang. Their 2014 paper investigated the use of idle graphics processors to accelerate audio DSP for real-time algorithms. Moreover, it discussed the importance of optimizing the code for GPU operation including the allocating shared resources, optimizing memory transfers and forced serialisation of feedback loops. Lastly, It introduced a new method for audio processing using GPUs as the default processor instead of an accelerator [19].

The results of these studies are important to consider in terms of how an APU would function. It's not as simple as just linking several GPUs on one die alongside the CPU component. If their distributed method showed to be the most effective, there would need to be a pre-determined way of assigning acoustical rooms to the CPU or GPU components based on their respective domain sizes. Additionally, the GPU component may need to function as one node rather than many when performing FDTD methods in order to keep synchronization times as low as pos-

sible. However, if this method could be refined in such a way that the CPU component of the APU could function as the sole computing node for sufficiently large domains while the properly load-balanced GPU component ran the smaller ones, synchronization times could be significantly reduced.

2.3.3 Acoustic Simulation

Room modeling is a popular aspect of acoustic simulation research and usage. The earliest source I examined in my research (which was by no means exhaustive) was a scene description model and rendering engine for interactive virtual acoustics proposed by Jean-Marc Jot and Jean-Michel Trivi in their titular 2006 paper. They reviewed two standard 3D positional audio representations (MPEG-4 AABIFS and OpenAL); then proposed expansions to the EAX environmental audio scene description model and rendering engine. Through techniques of environment morphing and incorporation of multiple reverberators, "the scene description model proposed in [their] paper allows a sound designer or composer to work as far as possible with parameters directly relevant to perceived audio effects, without relying on architectural or geometrical parameters when they are not necessary to produce a plausible environmental audio scene" [20]. This research could be combined with Nicolas Tsingos' work on precomputing geometry-based reverberation effects as part of the APU's rendering engine capabilities [11].

For a more recent look at the state of environmental acoustics, refer to Sönke Pelzera, Dirk Schrödera, Michael Vorländera, and Frank Wefers' overview published in the Journal of Building Performance Simulation in 2014. They discuss virtual reality (VR) acoustics and relevant components in terms of accuracy, implementation and computational effort. With regard to room modeling, they detailed the usage of room impulse responses (RIR) and geometric acoustics (GA). They compared GA with wave-based methods (which I will do in the next subsection), as well as examined real-time auralization and convolution while accounting for data management [21].

Imran Muhammad and Jin Yong Jeon have also researched immersive audio rendering for virtual environments. In 2016, they published a study that "investigated methods for sound propagation in virtual complex architectural environments for spatialized audio rendering to use in immersive [VR] scenarios". Their computational framework adopted a hybrid method of room acoustics simulations of complex architectural environments based on the Image Source Method (ISM) and a ray tracing (RT) algorithm. Their experiment successfully generated RIRs considering the reflection and sound scattering phenomenon in multi-coupled rooms. The sound propagation paths were computed between the listener and the speaker, including scenarios where they were located in different rooms. They did note, however, that "wave base solutions produce[sic] more realistic result as compared to ray tracing," and that further research in the area would be conducted using such methods [22].

2.3.4 Ray-Based vs. Wave-Based

At this point, a discussion on the strengths and weaknesses of ray-based methods and wave-based methods is warranted. In geometric acoustics, deterministic simulation methods use physical models of image sources (ISs). "In general, ISs are good approximations for perfectly reflecting or low-absorbing surfaces in large rooms with large distances between the sound source, wall and receiver" [21]. As Carl Schissler and Dinesh Manocha put it, "the most accurate geometric approach is the image source method where sound sources are recursively reflected over every triangle in a scene to form images of the source positions. Reflection paths from the listener to the source are then validated in reverse via occlusion queries" [23].

They further explained that "sound propagation is usually modeled as a combination of several different phenomena. Any sound received by a listener can be split into 3 components: direct sound, early reflections and late reverberation". Ray tracing comes into play when estimating the acoustic qualities of rooms [23]. "In physics, ray tracing is a method for calculating the path of waves or particles through a system with regions of varying propagation velocity, absorption characteristics, and reflecting surfaces. Under these circumstances, wavefronts may bend, change direction, or reflect off surfaces, complicating analysis. Ray tracing solves the problem by repeatedly advancing idealized narrow beams called rays through the medium by discrete amounts" [24].

Graphical ray tracing is already a well-defined feature and modus operandi of GPUs that began being researched soon after they hit the market. Timothy Purcell, Ian Buck, William Mark, and Pat Hanrahan are just one example of researchers who published studies on mapping ray tracing onto a GPU [25]. They did so in 2002, only a few years after the release of NVIDIA's first GPU. One can see how acoustic ray tracing is particularly suited to the GPU architecture because of its parallels to graphical ray tracing. Unfortunately, there are disadvantages to this method. "Round robin tests of room acoustics simulation programmes revealed the drawback of the IS-method, which is the incapability of simulating the important wave phenomena of surface and obstacle scattering" [21]. Furthermore, "while accurate, the image source method is extremely slow: the running time increases exponentially with the reflection depth" [23].

The ideal APU is designed for ultimate accuracy; hence, we consider wave field synthesis. "Wave Field Synthesis (WFS) is a spatial audio reproduction system that provides an accurate spatial sound field in a wide area. This sound field is rendered through a high number of loudspeakers to emulate virtual sound sources". "A synthesis operator for each loudspeaker can be derived. The general 3D solution can be transformed into the 2-D solution, which is sufficient for reconstructing the original sound field in the plane of listening. For that purpose a linear array of loudspeakers is employed to generate the sound field of virtual sources" [26]. Essentially, WFS seeks to accurately recreate the actual sound propagation of a given environment by

simulating sound source locations within and around said environment and calculating the changes of their resulting waves over time (accounting for room effects) in order to produce localized representations that can be traversed in three-dimensional space.

For comparison, I defer to Schissler and Manocha: "ray tracing for sound propagation has several advantages over other numeric and geometric simulation techniques: it can easily handle dynamic scenes, requires less preprocessing, and is much more amenable to realtime simulation. Due to the simplicity of ray tracing, it can also be implemented on the GPU, improving simulation times" [23]. This is a strong reason to design an APU around the ray tracing method since its GPU component would already be optimized for performing such calculations. Further, "numeric methods are accurate and produce true-to-life results but are generally too slow to be used in realtime systems. FDTD methods [including WFS] are very accurate but require enormous amounts of time and memory to simulate" [23].

Indeed, Jose A. Belloch, Miguel Ferrer, Alberto Gonzalez, Jorge Lorente, and Antonio M. Vidal noted in their 2013 paper that "WFS systems require high computational capacity since they involve multiple [virtual] loudspeakers and multiple virtual sources. Furthermore improvements of the[sic] spatial audio perception imply even higher processing capacity, mainly to avoid artifacts when the virtual sources move, and compensate the room effects at certain control points within the listening area" [26]. Despite that, it is important to remember that Schissler and Manocha wrote their words 8 years ago, as of this writing. That is to say that Belloch, Ferrer, Gonzalez, Lorente, and Vidal's paper proposed solutions to some of the problems of WFS in a real-world environment that can be applied to the virtual environments present in gaming. They produced a "rendering system that uses a Room Compensation block that is able to render up to 300 virtual sound sources in real time. The main feature of this system is that all its audio processing was carried out by a GPU, while the CPU could be being used for other tasks at the same time" [26].

This supports one of my initial reasons for proposing this hardware unit: freeing up CPU cores and cycles for use in other things in a game console. It is also indicative of why WFS is the method I propose for an APU. Because virtual environments do not have the same limitations on loudspeaker counts that real-world WFS implementations do, it is conceivable to construct an entirely new virtual WFS system that surrounds the player object at all times, and moves with them in the case of first and third-person perspective games. This would be an invisible spherical mesh surrounding the player object where each "loudspeaker" is a sound playback object. The scene environment in the game engine would be populated with sound source objects and the APU would be responsible for calculating the wave field sounds of those sources based on the position of the player object, then distributing them across the WFS mesh around the player.

2.3.5 Acoustical Complexities

"Time domain wave-based methods sidestep the simplifying hypotheses underlying geometric (ray-based) methods and offer, in theory, a complete solution to the problem of room acoustics simulation. Many issues remain at the level of algorithm design, particularly under realistic room configurations. The main design criteria are: (a) arbitrary room geometry, (b) general passive frequency-dependent and spatially-varying wall conditions, and (c) adequate modeling of viscothermal and relaxation effects". Stefan Bilbao and Brian Hamilton noted this at the beginning of their paper presenting a solution to that third criterion: the effect of losses due to viscothermal and relaxation effects on sound wave propagation (leading to frequency-dependent decay in free space). They also proposed a framework for the construction of time-domain wave-based schemes allowing for stability under the previously mentioned attributes (a) and (b) [27].

They are not alone in trying to address the complexities of accurate sound simulation in a virtual environment. In 2009, Changxi Zheng and Doug L. James proposed "a practical method for automatic procedural synthesis of synchronized harmonic bubble-based sounds from 3D fluid animations" [28]. In 2011, Brent Cowan and Bill Kapralos published a paper in which they described how they built upon an existing GPU-based acoustical occlusion/diffraction modeling method in order to better approximate acoustical occlusion/diffraction effects for complex, multi-room environments. They described their method as computationally efficient, allowing it to be incorporated into real-time, dynamic, and interactive virtual environments and video games where the scene is arbitrarily complex [29].

As recently as 2018, Kenji Kojima and Takahiro Kitagawa published a paper proposing low-cost methods for determining sound propagation paths using layered quadtree grids and enabling sound designers to design more realistic sounds by using audio buses and sound effects. By focusing on space and openings, they showed how to dynamically reduce processing for unnecessary sources and determine propagation paths. Their experiments showed that their method processes each source more than twenty times faster than the conventional method [30]. As virtual acoustic simulation is a nascent, yet burgeoning, field, there are many other studies that have been conducted or are currently in progress beyond the scope of this paper. All of this research will be necessary to make an APU a reality.

2.3.6 Sound Synthesis

Thus far, the methods and complexities of processing and playback of existing sound files, essentially defining and utilizing the APU as an accelerator, have been examined. A further consideration for APU functionality would be the capability to synthesize new sounds at runtime. Several researchers have looked into real-time sound synthesis for virtual environments. In 2011, Leonard J. Paul published a paper that investigated then-current re-

search in sound granulation and demonstrated some effective methods of utilizing granulation for games. He noted that, ”granulation can be used to dynamically change the playback speed or pitch of [a] speech sample independently of another in real-time. This can make the retriggering of certain speech lines (“Hey! Over here!”) sound a little less repetitive.” ”Crowd backgrounds and chants can be made much more dynamic by variation of the voices using granulation. One the[sic] easiest uses is to augment existing non-specific backgrounds such as room tones to extend their length without noticeable loop points” [31]. Such capabilities could help with sonic immersion, as well as the practicalities of playing video games when it comes to repeated sounds, dialog or otherwise.

One year later, Verron Charles and George Drettakis presented a sound synthesizer dedicated to particle-based environmental effects, for use in interactive virtual environments. Their synthesis engine was based on five physically-inspired basic elements that they called sound atoms, which could be parameterized and stochastically distributed in time and space. They illustrated their approach with three models that are commonly used in video games: fire, wind, and rain. Their proposed synthesizer, as they put it, was ”able to generate realistic sounds evoking a wide variety of phenomena existing in our natural environment” [32].

In 2015, Sadjad Siddiq published a paper that ”introduced an algorithm to morph between two or more sounds, which can be used to synthesize new sounds in real-time whose features lie between the tone color, amplitude envelope, pitch, and length of the source sounds. It is used to increase variation of commonly used impact sounds in video games, but the algorithm can also be applied to other sound types like instrument sounds” [33]. While Siddiq notes that there is further research to be done on this method, particularly with regard to optimization, I posit that it is a viable consideration for acoustic rendering of sonic impulses that better communicate them as natural phenomena in a virtual environment.

3 METHODOLOGY

My methodology for the proposed ideal APU is based on historical research detailed in my literature review. The sources I examined implemented both empirical and theoretical research (not to mention historical research of their own) in order to draw conclusions from studies conducted regarding their hypotheses.

3.1 Historical

Through comprehensive research of the Audio Engineering Society electronic library, as well as several university and corporate publications, I was able to find data concerning several aspects of acoustical simulation in virtual environments. Developing an ideal APU will require further historical research to determine any oversights as to the functionality of the APU.

3.2 Empirical

Several of the sources I discuss in this paper incorporate acoustic measurements that the researchers conducted or cited in order to conduct their experiments. Developing an ideal APU will require empirical research into the speed, efficiency, and effectiveness of their various acoustic simulation algorithms.

3.3 Theoretical

Several of the sources I discuss in this paper present computational models for executing acoustic simulations. Developing an ideal APU will require theoretical research in order to solve the problems in their algorithms pointed out by said sources.

4 METHOD

For this paper, I conducted qualitative research on the research conducted by my sources. To construct an ideal APU, my high-level method is as follows:

1. Historical research is conducted to ascertain other features and considerations for acoustic simulation on the APU.
2. Theoretical research is conducted in order to determine the amount of processing power needed to perform each of the functions outlined in the forthcoming proposal (based on the work produced by the studies in the literature review). This would determine the maximum amount of concurrent calculations that need to be executable, and thus the number of (GP)GPU cores in the APU.
3. A high-end CPU is selected and implemented for the purposes of empirical research running acoustic simulations that are distributed across the CPU and GPU nodes.
4. The methods/functions created for the acoustic simulations are assembled into a foundational API for the APU.
5. A software graphic user interface and coding environment is constructed to allow users to execute real-time acoustic renderings.

5 PROPOSAL

5.1 Crafting the APU

Based on my research, an ideal audio processing unit should have 3 broad capabilities: sound propagation calculation, sound processing, and sound synthesis. These 3 areas can be broken down into the following:

1. Sound propagation:
   a. room modeling
   b. reverberation
   c. refraction/scattering/diffusion
   d. reflection
   e. impulse response
2. Sound processing:

   a. filtering
   b. occlusion
   c. DSP/acceleration
   d. load balancing
   e. HRTFs* *(algorithm dependent, binaural considerations)*
3. Sound synthesis:
   a. granulation
   b. sound particle generation
   c. morphing

5.1.1 API and Interface

In my literature review, I covered many different approaches to solutions for these various challenges; however, I did not discuss some of the required external changes that I propose for the successful creation of an APU. While the GPGPU pipeline removes the need for converting pertinent data into graphical data for the purposes of performing calculations, the high-level user interface API should still be explicitly constructed around parameters and metadata that is specific to sound. For example, it should contain a library of equations and methods that it uses to calculate acoustic phenomena such as the acoustic wave equation, acoustic pressure, pulse integral intensity, etc., all of the different equations used in the literature I reviewed at a minimum. The API should be a living system that is kept updated with new equations and methods as they are discovered.

Beyond this, game (development) engines would need to incorporate new metadata parameters for their object models. For example, the polygons of a table should be able to support a parameter that defines their material composition (e.g., wood) so that the APU API could call the appropriate methods and equations. The environment should have a parameter for medium (air, water, etc.) and temperature that determines the equations used for sound propagation calculations. Essentially, the encapsulation provided by the APU API would need to be as comprehensive as possible to make its integration into game engines as seamless and intuitive as possible.

One way of doing this would be to integrate the API into a middleware application in the vein of Audiokinetic Inc.'s Wave Works Interactive Sound Engine (Wwise) or Firelight Technologies' FMOD® that would operate alongside the game engines. Given that those two platforms operate using banks of sounds and allow audio implementers to define metadata and parameters for how and when the sounds are triggered at run-time, they seem to be suitable examples of interfaces for the APU API.

The pipeline could be constructed such that the game engines would contain the metadata for the polygonal objects and environments in each scene while the middleware would organize, trigger, and process the sounds based on those parameters. The game engine would either execute the playback of 2D sounds itself (nondiegetic sounds that are not actually a part of the virtual environment such as music or narration), or utilize existing middleware solutions to control their playback. Meanwhile, three-dimensional sounds (diegetic sounds that are actually occurring in the virtual environment) would be sent to the APU middleware.

5.1.2 Configuration

Based on my research, particularly the work of Saarelma, Califa, and Mehra [18], I believe the ideal APU would best be designed as an integrated single chip consisting of a CPU module and several GPU modules operating in a GPGPU configuration. Consider their experimental configuration in Figure 1.

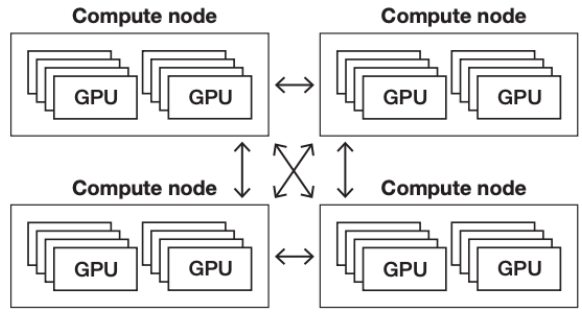

Fig. 1. Node configuration of Saarelma, Califa, and Mehra's GPGPU cluster [18]

I propose substituting a unitary CPU for one of their designated compute nodes and utilize a similar GPU distribution for the rest; however, I note that the precise number of GPU cores required cannot be determined at this time due to the need for further research and development of the sonic algorithms required for truly accurate acoustic modeling (not to mention optimization to reduce the amount of computation needed). As such, that detail is beyond the scope of this paper, in addition to the determination of power consumption. Yet, there remains the option of the APU actually being structured as a chipset on a board, especially considering the fact that it would contain its own CPU. Though such a design would reinstate the synchronization concerns discussed by Saarelma, Califa, and Mehra, [18] it would allow for easier alteration of GPU module counts.

The CPU-GPGPU configuration seems best suited to be able to perform the calculations of all of the references I cited in my literature review, especially given that today's top-of-the-line GPU, the NVIDIA Titan RTX, can perform 130 teraflops (1.3 trillion floating point operations per second) with 672 GB/s of total memory bandwidth. The computational capability of a chipset consisting of a CPU with 8 Titan GPUs (to mimic the configuration of the previously mentioned study) is staggering; however, this introduces a new consideration: cost.

5.2 Discussion

A Titan RTX GPU is priced at 2,499 US Dollars, as of this writing. Eight of them would then cost just under 20,000 USD. Even if all of the other requirements for an ideal APU were suddenly met to perfection, that is

hardly practical for game console implementation today. The video game console market, in contrast to the personal computer gaming market, has traditionally been a battle of affordability. Consider that the most expensive console ever released (ignoring special editions made with valuable metals) was the Phillips CDi, released at 700 USD in 1991 [34] (1,314.37 USD adjusted for inflation as of 2019). An implementation of the ideal APU as best as it could exist today would consist of offering as much processing power as physically possible without regard to what is fiscally possible. Therefore, the audio processing unit of today might be best designed for one of the many functions discussed in the literature review, rather than all of them, especially if one factors in the cost of research and development into all of those capabilities.

With that stated, the ideal APU is not proposed as a reality of today; but rather of tomorrow. Given the computing capabilities of the best CPUs and GPUs currently available; it is certainly feasible to run quite large and complex acoustic simulations with astounding accuracy in real-time, given enough computing units. Accordingly, the ideal APU is a realization of the following:

- optimization of the numerous algorithms that power acoustic simulations while making them execute in an optimally load-balanced fashion with respect to the GPGPU pipeline
- creation of new algorithms that render other extant acoustic phenomena not already accounted for
- creation of an intuitive control interface that intelligently operates all of the algorithms
- consolidation of the experimental CPU and GPGPU modules into the design and creation of an integrated CPU-GPGPU chip that requires as few GPU units as possible in as small a form factor as possible

With the right capital, which can be collected through investments from multiple parties for whom this research has strong potential for profitability (even beyond applications in gaming), the ideal APU can be realized a lot sooner that it might otherwise be.

6 CONCLUSION

This paper examined a variety of research into acoustic computation and modeling for virtual environments for the purpose of proposing an ideal audio processing unit that would offer accurate real-time acoustic simulation in video games for home consoles. The proposal covers architecture, configuration, operation, processing, and functionality based on the myriad sources outlined in the literature review that were analyzed and discussed.

This paper was crafted by an audio engineer in the video game and music industries who began his collegiate studies as a computer science student. With his unique multidisciplinary background, he is best able to understand and communicate the disparate aspects of such a comprehensive product, while operating effectively in the team of researchers, engineers, and designers necessary to realize the product. As a long-time consumer of console games, as well as someone who has contributed to their development, he has a vested interest in seeing them achieve their ultimate potential of immersion.

7 ACKNOWLEDGMENT

This research was conducted in spring 2019 when Etaba Assigana was a graduate student at American University. Thank you to professor Braxton Boren, Ph.D., for his input and feedback in the creation of this paper.

8 REFERENCES

[1] W. Contributors, “Graphics Processing Unit,” (2019).

[2] W. Contributors, “General-purpose computing on graphics processing units,” (2019).

[3] W. Contributors, “AMD Accelerated Processing Unit,” (2018).

[4] P. W. Rubin, “What to Expect from Sony’s Next-Gen PlayStation,” (2019 apr).

[5] “Audio Processing Unit (APU)—NVIDIA,” (2001).

[6] W. Contributors, “SoundStorm,” (2018).

[7] L. Savioja, V. Välimäki, J. O. Smith, “Audio Signal Processing Using Graphics Processing Units,” *Journal of the Audio Engineering Society*, vol. 59, no. 1/2, pp. 3–19 (2011 mar).

[8] L. Savioja, V. Välimäki, J. O. Smith III, “Real-Time Additive Synthesis with One Million Sinusoids Using a GPU,” presented at the *AES 128th Convention* (2010 may).

[9] N. Tsingos, W. Jiang, I. Williams, “Using Programmable Graphics Hardware for Acoustics and Audio Rendering,” *Journal of the Audio Engineering Society*, vol. 59, no. 9, pp. 628–646 (2011).

[10] N. Tsingos, “Using Programmable Graphics Hardware for Acoustics and Audio Rendering,” presented at the *AES 127th Convention* (2009 oct).

[11] N. Tsingos, “Pre-computing Geometry-Based Reverberation Effects for Games,” presented at the *AES 35th International Conference* (2009 feb).

[12] T. Leidi, T. Heeb, M. Colla, J.-P. Thiran, “Event-Driven Real-Time Audio Processing with GPGPUs,” presented at the *AES 130th Convention* (2011 may).

[13] N. Corporation, “cuFFT,” (2019), URL `https://developer.nvidia.com/cufft`.

[14] N. K. Govindaraju, B. Lloyd, Y. Dotsenko, B. Smith, J. Manferdelli, “High Performance Discrete Fourier Transforms on Graphics Processors,” (2008).

[15] S. Chellappa, F. Franchetti, J. M. Moura, M. Puschel, Y. Voronenko, “Discrete Fourier Transform on Multicore,” *IEEE SIGNAL PROCESSING MAGAZINE*, pp. 91–102 (2009 nov), [Online]. Available: 10.1109/MSP.2009.934155.

[16] J. A. S. Angus, A. Caunce, “A GPGPU Approach to Improved Acoustic Finite Difference Time Domain Calculations,” presented at the *AES 128th Convention* (2010 may).

[17] A. Southern, D. Murphy, G. Campos, P. Dias, "Finite Difference Room Acoustic Modelling on a General Purpose Graphics Processing Unit," presented at the *AES 128th Convention* (2010 may).

[18] J. Saarelma, J. Califa, R. Mehra, "Challenges of Distributed Real-Time Finite-Difference Time-Domain Room Acoustic Simulation for Auralization," presented at the *Conference on Spatial Reproduction* (2018 aug).

[19] N. Jillings, Y. Wang, "CUDA Accelerated Audio Digital Signal Processing for Real-Time Algorithms," presented at the *AES 137th Convention* (2014 oct).

[20] J.-M. Jot, J.-M. Trivi, "Scene Description Model and Rendering Engine for Interactive Virtual Acoustics," presented at the *AES 120th Convention* (2006 may).

[21] S. Pelzera, D. Schrödera, M. Vorländera, F. Wefers, "Virtual reality for architectural acoustics," *Journal of Building Performance Simulation*, vol. 8, no. 1, pp. 15–25 (2014), [Online]. Available: 10.1080/19401493.2014.888594.

[22] I. Muhammad, J. Y. Jeon, "Immersive Audio Rendering for Interactive Complex Virtual Architectural Environments," presented at the *Conference on Audio for Virtual and Augmented Reality* (2016 sep).

[23] C. Schissler, D. Manocha, "GSound: Interactive Sound Propagation for Games," presented at the *41st International Conference: Audio for Games*, pp. P2–6 (2011).

[24] W. contributors, "Ray tracing (physics) — Wikipedia, The Free Encyclopedia," (2019).

[25] T. J. Purcell, I. Buck, W. R. Mark, P. Hanrahan, "Ray Tracing on Programmable Graphics Hardware," *ACM Transactions on Graphics*, vol. 21, no. 3, pp. 703–712 (2002), [Online]. Available: 10.1145/566654.566640.

[26] J. A. Belloch, M. Ferrer, A. Gonzalez, J. Lorente, A. M. Vidal, "GPU-Based WFS Systems with Mobile Virtual Sound Sources and Room Compensation," presented at the *AES 52nd International Conference* (2013 sep).

[27] S. Bilbao, B. Hamilton, "Wave-Based Room Acoustics Simulation: Explicit/Implicit Finite Volume Modeling of Viscothermal Losses and Frequency-Dependent Boundaries," *Journal of the Audio Engineering Society*, vol. 65, no. 1/2, pp. 78–89 (2017 feb), [Online]. Available: 10.17743/jaes.2016.0057.

[28] C. Zheng, D. L. James, "Harmonic Fluids," (2009).

[29] B. Cowan, B. Kapralos, "GPU-Based Acoustical Diffraction Modeling for Complex Virtual Reality and Gaming Environments," (2011 feb).

[30] K. Kojima, T. Kitagawa, "A Low Cost Method for Applying Acoustic Features to Each Sound in Virtual 3D Space Using Layered Quadtree Grids," presented at the *Conference on Spatial Reproduction* (2018 jul).

[31] L. J. Paul, "Granulation of Sound in Video Games," presented at the *AES 41st International Conference* (2011 feb).

[32] C. Verron, G. Drettakis, "Procedural Audio Modeling for Particle-Based Environmental Effects," presented at the *AES 133rd Convention* (2012 oct).

[33] S. Siddiq, "Real-Time Morphing of Impact Sounds," presented at the *AES 139th Convention* (2015 oct).

[34] T. Top Staff, "Top 10 Most Expensive Video Game Consoles Ever," (2017), URL `https://bit.ly/2E2gfHV`.

APPENDIX

A.1 Annotated Bibliography

See attached file at the end of this paper.

NOMENCLATURE

API = a set of functions and procedures allowing the creation of applications that access the features or data of an operating system, application, or other service

APU = audio processing unit, theoretical hardware unit dedicated to real-time acoustic modeling calculations

CUDA = NVIDIA's implementation of GPGPU, originally based around the C programming language but has since moved to C++, originally an acronym for Complete Unified Digital Architecture

cuFFT = a foundational CUDA library based on the well-known Cooley-Tukey and Bluestein algorithms. It is used for building commercial and academic applications across disciplines such as computational physics, molecular dynamics, quantum chemistry, seismic and medical imaging. With support for batched transforms and optimized precision arithmetic, it is widely used for building deep-learning based computer vision applications.

DFT = the most important discrete transform, used to perform Fourier analysis in many practical applications, the discrete Fourier transform converts a finite sequence of equally-spaced samples of a function into a same-length sequence of equally-spaced samples of the discrete-time Fourier transform (DTFT)

DSP = digital signal processing, the numerical manipulation of audio signals

DTFT = discrete-time Fourier transform is a form of Fourier analysis that is applicable to a sequence of values, often used to analyze samples of a continuous function

EAX = a number of digital signal processing presets for audio, present in Creative Technology Sound Blaster sound cards starting with the Sound Blaster Live and the Creative NOMAD / Creative ZEN product lines.

FFT = a fast Fourier transform algorithm computes the discrete Fourier transform (DFT) of a sequence, or its inverse. Fourier analysis converts a signal from its original domain (often time or space) to a representation in the frequency domain and vice versa.

GPU = graphics processing unit, dedicated chips that perform image calculations in real-time using a parallel architecture

GPGPU = general purpose graphics processing unit, utilizes a framework that transforms non-visual data into image format in order to perform parallel calculations on the data and return it
HRTF = head related transfer function, a response that characterizes how an ear receives a sound from a point in space
ISM = image source model/method, a well-known technique that can be used in order to generate a synthetic room impulse response (RIR) in a given environment
RIR = room impulse response, a transfer function between a sound source and an acoustic sensor
SIP = semiconductor intellectual property core, reusable chip design that is the intellectual property of one party
WFS = a spatial audio reproduction system that provides an accurate spatial sound field in a wide area

THE AUTHOR

Etaba Assigana

Etaba Assigana is a graduate student at American University in Washington, D.C. He is pursuing a master of arts degree in audio technology, having already received a master of entertainment technology degree from Carnegie Mellon University and a bachelor of science degree in computational media from the Georgia Institute of Technology. His career pursuits are game audio, sound design, and music.

Apppendix A1:
Annotated Bibliography

Acoustical Computation on GPUs Overview

Acoustic computation on graphics processing units

Angus, J. A. S., & Caunce, A. (2010). A GPGPU Approach to Improved Acoustic Finite Difference Time Domain Calculations. In *AES 128th Convention*. London: Audio Engineering Society. Retrieved from http://www.aes.org.proxyau.wrlc.org/e-lib/browse.cfm?elib=15260

This paper presents a proof of concept that Finite Difference Time Domain calculations can be a carried out with ease on a Graphics Processing Unit thanks to the advent of CUDA.

Belloch, J. A., Ferrer, M., Gonzalez, A., Lorente, J., & Vidal, A. M. (2013). GPU-Based WFS Systems with Mobile Virtual Sound Sources and Room Compensation. In *AES 52nd International Conference*. Guildford: Audio Engineering Society. Retrieved from http://www.aes.org.proxyau.wrlc.org/e-lib/browse.cfm?elib=16896

This paper proposes a GPU implementation that yields maximum parallelism by adapting the required computations to the different GPU architectures (Tesla, Fermi and Kepler).

Chellappa, S., Franchetti, F., Moura, J. M. F., Püschel, M., & Voronenko, Y. (2009, November). Discrete Fourier Transform on Multicore. *IEEE SIGNAL PROCESSING MAGAZINE*, 91–102. http://doi.org/10.1109/MSP.2009.934155

This paper examines the challenges of parallelization, vectorization, and memory hierarchy optimization in performing DFT on multicore CPU and GPU architectures.

Contributors, W. (2019). General-purpose computing on graphics processing units. In *Wikipedia, The Free Encyclopedia*.

This article provides information about the creation and usage of the GPGPU pipeline.

Contributors, W. (2019). Graphics Processing Unit. In *Wikipedia, The Free Encyclopedia*. Retrieved from https://en.wikipedia.org/wiki/Graphics_processing_unit

This article provides information about GPUs, their history, architecture, etc.

Cowan, B., & Kapralos, B. (2011). GPU-Based Acoustical Diffraction Modeling for Complex Virtual Reality and Gaming Environments. London: Audio Engineering Society. Retrieved from http://www.aes.org.proxyau.wrlc.org/e-lib/browse.cfm?elib=15754

This paper examines a method of audio diffraction modeling that builds upon a previous GPU-based acoustical occlusion method that was fairly limited with respect to approximating such multi-room environments. The method can operate in real-time allowing it to be incorporated into interactive, and dynamic gaming and virtual reality applications.

Govindaraju, N. K., Lloyd, B., Dotsenko, Y., Smith, B., & Manferdelli, J. (2008). High Performance Discrete Fourier Transforms on Graphics Processors. Microsoft Corporation. Retrieved from https://www.microsoft.com/en-us/research/wp-content/uploads/2008/01/FftGpuSC08.pdf

This paper examines several algorithms for performing FFTs efficiently on a GPU. Their hierarchical FFT minimizes the number of memory accesses by combining transpose operations with the FFT computation. It also addresses numerical accuracy issues. The results indicate a significant performance improvement over optimized GPU-based and CPU-based FFT algorithms.

Jillings, N., & Wang, Y. (2014). CUDA Accelerated Audio Digital Signal Processing for Real-Time Algorithms. In *AES 137th Convention*. Los Angeles: Audio Engineering Society. Retrieved from http://www.aes.org.proxyau.wrlc.org/e-lib/browse.cfm?elib=17443

This paper investigates the use of idle graphics processors to accelerate audio DSP for real-time algorithms and discusses the importance of optimizing the code for GPU operation including the allocating shared resources, optimizing memory transfers and forced serialization of feedback loops. It also introduces a new method for audio processing using GPUs as the default processor instead of an accelerator.

Leidi, T., Heeb, T., Colla, M., & Thiran, J.-P. (2011). Event-Driven Real-Time Audio Processing with GPGPUs. In *AES 130th Convention*. London: Audio Engineering Society. Retrieved from http://www.aes.org.proxyau.wrlc.org/e-lib/browse.cfm?elib=15884

This paper analyzes the limits of audio processing on GPGPUs and present an approach based on event-driven scheduling, that maximizes data pressure to favor performance improvements when combining multi-core CPUs and GPGPUs..

Savioja, L., Välimäki, V., & Smith, J. O. (2011). Audio Signal Processing Using Graphics Processing Units. *Journal of the Audio Engineering Society*, *59*(1/2), 3–19. Retrieved from http://www.aes.org.proxyau.wrlc.org/e-lib/browse.cfm?elib=15772

This paper concludes that GPUs are well suited for audio applications that need heavy computation, can be parallelized, and can tolerate some latency. It notes the essentiality of getting enough concurrent threads running so that the GPU is fully utilized at an acceptable cost of latency (a few milliseconds).

Savioja, L., Välimäki, V., & Smith III, J. O. (2010). Real-Time Additive Synthesis with One Million Sinusoids Using a GPU. In *AES 128th Convention*. London: Audio Engineering Society. Retrieved from http://www.aes.org.proxyau.wrlc.org/e-lib/browse.cfm?elib=15259

This paper describes tests analyzing the performance of a GPU implementation of sinusoidal modeling versus a CPU implementation. The results show that a parallel implementation of additive synthesis on a GPU offers over 1300 times the performance of a serial table-lookup implementation on CPU.

Southern, A., Murphy, D., Campos, G., & Dias, P. (2010). Finite Difference Room Acoustic Modelling on a General Purpose Graphics Processing Unit. In *AES 128th Convention*. London: Audio Engineering Society. Retrieved from http://www.aes.org.proxyau.wrlc.org/e-lib/browse.cfm?elib=15325

This paper discusses an accelerated GPU CUDA implementation of a 2D finite-domain time-difference acoustical model for generating room impulse response for arbitrary room geometries. Results say that wave-based modeling is suitable to generating RIR datasets.

Tsingos, N. (2011). Using Programmable Graphics Hardware for Acoustics and Audio Rendering. *Journal of the Audio Engineering Society*, *59*(9), 628–646. Retrieved from http://www.aes.org.proxyau.wrlc.org/e-lib/browse.cfm?elib=15979

This paper considers the challenges of auralization systems including audio signal processing, numerical calculations and linear algebra, and geometry processing. It examines the method of converting audio data into visual data, which current GPUs are already equipped to handle.

Individual Research

Acoustic simulation on graphics processing units

cuFFT. (2019). Retrieved May 3, 2019, from https://developer.nvidia.com/cufft

This page on NVIDIA's developer site offers details and links to further information about the cuFFT library for CUDA.

Audio Processing Unit (APU)|NVIDIA. (2001). NVIDIA Corporation. Retrieved from https://www.nvidia.com/object/apu.html

This technical brief outlines the meaning of an audio processing unit, as NVIDIA defined and constructed it. Their APU was designed to offload complex audio algorithms and effects processing from the CPU while rendering completely to system memory.

Contributors, W. (2018). AMD Accelerated Processing Unit. In *Wikipedia, The Free Encyclopedia*.

This article provides information about AMD's "APU" designated microprocessors.

Contributors, W. (2019). Ray tracing (physics). In *Wikipedia, The Free Encyclopedia*. Retrieved from https://en.wikipedia.org/w/index.php?title=Ray_tracing_(physics)&oldid=888867063

This article provides information on ray tracing with regard to its applications in physics.

Jot, J.-M., & Trivi, J.-M. (2006). Scene Description Model and Rendering Engine for Interactive Virtual Acoustics. In *AES 120th Convention*. Paris: Audio Engineering Society. Retrieved from http://www.aes.org.proxyau.wrlc.org/e-lib/browse.cfm?elib=13464

This paper introduces a statistical reverberation model to account for per-source distance and directivity effects. An efficient spatial reverberation and mixing architecture is described for the spatialization of multiple sound sources around a virtual listener navigating across multiple connected virtual rooms including acoustic obstacles.

Kojima, K., & Kitagawa, T. (2018). A Low Cost Method for Applying Acoustic Features to Each Sound in Virtual 3D Space Using Layered Quadtree Grids. In *Conference on Spatial Reproduction*. Tokyo: Audio Engineering Society. Retrieved from http://www.aes.org.proxyau.wrlc.org/e-lib/browse.cfm?elib=19616

This paper proposes low cost methods for determining the sound propagation paths using layered quadtree grids and enabling sound designers to design more realistic sounds by using audio buses and sound effects. By focusing on space and openings, it shows how to dynamically reduce processing for unnecessary sources and determine propagation paths. Experiments show this method processes each source more than twenty times faster than the conventional method.

Muhammad, I., & Jeon, J. Y. (2016). Immersive Audio Rendering for Interactive Complex Virtual Architectural Environments. In *Conference on Audio for Virtual and Augmented Reality*. Los Angeles: Audio Engineering Society. Retrieved from http://www.aes.org.proxyau.wrlc.org/e-lib/browse.cfm?elib=18508

This paper described and investigated the techniques and methodology for sound propagation in virtual architectural environments. They computed Room Impulse Responses using the Image Source Method and ray tracing, then applied binaural reproduction.

Paul, L. J. (2011). Granulation of Sound in Video Games. AES 41st International Conference. Retrieved from http://www.aes.org.proxyau.wrlc.org/e-lib/browse.cfm?elib=15760

This paper considers the advantages and disadvantages of granulation for use in video games in conjunction with sample-based methods in order to reduce the amount of sample data required for specific audio events.

Pelzera, S., Schrödera, D., Vorländera, M., & Wefers, F. (2014). Virtual reality for architectural acoustics. *Journal of Building Performance Simulation*, *8*(1), 15–25. http://doi.org/10.1080/19401493.2014.888594

This article reviews simulation tools for room acoustics and sound insulation, and discusses their application in real-time VR-systems for architectural acoustics.

Purcell, T. J., Buck, I., Mark, W. R., & Hanrahan, P. (2002). Ray Tracing on Programmable Graphics Hardware. *ACM Transactions on Graphics*, *21*(3), 703–712. https://doi.org/10.1145/566654.566640

This paper evaluates trends in programmability of the graphics pipeline and explains how ray tracing can be mapped to graphics hardware.

Saarelma, J., Califa, J., & Mehra, R. (2018). Challenges of Distributed Real-Time Finite-Difference Time-Domain Room Acoustic Simulation for Auralization. In *Conference on Spatial Reproduction*. Tokyo: Audio Engineering Society. Retrieved from http://www.aes.org.proxyau.wrlc.org/e-lib/browse.cfm?elib=19609

This paper proposes a system for real-time auralization of distributed finite-difference time-domain room acoustic simulation and analyzes the results of several experiments using the system, noting that the fastest performance times occurred with the use of a single computing node and multiple GPGPUs.

Schissler, C., & Manocha, D. (2011). GSound: Interactive Sound Propagation for Games. *41st International Conference: Audio for Games*, P2-6. London: Audio Engineering Society.

This paper presents a sound propagation and rendering system for generating realistic environmental acoustic effects in real time for game-like scenes.

Siddiq, S. (2015). Real-Time Morphing of Impact Sounds. *AES 139th Convention*. Retrieved from http://www.aes.org.proxyau.wrlc.org/e-lib/browse.cfm?elib=17964

This paper introduces an algorithm to morph between two or more sounds, which can be used to synthesize new sounds in real-time whose features lie between the tone color, amplitude envelope, pitch and length of the source sounds. It is used to increase variation of commonly used impact sounds in video games such as footsteps, collisions of weapons, shots, punches, rain, etc.

Tsingos, N. (2009, February 1). Precomputing Geometry-Based Reverberation Effects for Games. San Francisco: Audio Engineering Society. Retrieved from http://www.aes.org.proxyau.wrlc.org/e-lib/browse.cfm?elib=15164

This paper presents solutions to integrate off-line geometrical acoustic modeling in game environments. By pre-computing image-source gradients for early reflections and directional decay profiles, location dependent reverberation effects can be generated without storing or accessing the actual geometry at run-time. The pipeline enables fine-grain rendering of distance and surface proximity effects and modeling of both outdoor and coupled indoor spaces with arbitrary reverberation decay profiles.

Tsingos, N., Jiang, W., & Williams, I. (2011). Using Programmable Graphics Hardware for Acoustics and Audio Rendering. *Journal of the Audio Engineering Society*, *59*(9), 628–646. Retrieved from http://www.aes.org/e-lib/browse.cfm?elib=15979

This paper looks at a comparison between CPU and GPU implementations for acoustic modeling and looks at results for 3D audio processing and sound scattering simulations.

Verron, C., & Drettakis, G. (2012). Procedural Audio Modeling for Particle-Based Environmental Effects. In *AES 133rd Convention*. San Francisco: Audio Engineering Society. Retrieved from http://www.aes.org.proxyau.wrlc.org/e-lib/browse.cfm?elib=16506

This paper presents a synthesis engine based on five physically-inspired basic elements, referred to as sound atoms, that can be parameterized and stochastically distributed in time and space. The approach is illustrated with three models that are commonly used in video games: fire, wind, and rain.

Zheng, C., & James, D. L. (2009). Harmonic Fluids. New Orleans: Association for Computing Machinery. Retrieved from http://www.cs.cornell.edu/projects/HarmonicFluids/harmonicfluids.pdf

This paper proposes the first practical physically based method for synthesizing synchronized harmonic fluid sounds for computer animation. They model the creation of bubbles by air entrainment at the fluid surface; the advection of these bubbles with the fluid flow; the surface vibrations induced by the bubbles' vibrations; and the radiation of these vibrations into the air, producing sound.

Top 10 Most Expensive Video Game Consoles Ever. (2017). Retrieved May 5, 2019, from TENz.com website: https://tenz.top/top-10-expensive-video-game-consoles-ever/

This article compiles a list of the ten most expensive video game consoles ever released, based on their launch prices in US Dollars.