Parallel straight lines do not meet one another in either direction. — Euclid

8 ACOUSTIC RENDERING

AURALIZATION describes the process of employing physical and mathematical models based on Euclidean geometry to render a virtual auditory scene audible. Thereby a binaural sound output that simulates the acoustic experience for a certain location and for a specific listener in this virtual environment is created. Auralization is an important factor for the research in this thesis, as sound and acoustics are both used as the main carrier to display abstract information. The majority of current applications that employ 3D sounds and room acoustics simulations use freely and commercially available APIs, such as OpenAL/EFX, FMOD or AM3D (Firelight Technologies Pty, Ltd, 2001-2008; AM3D A/S, 2008; Hiebert, 2006; Peacock et al., 2006). Although these APIs are easy to deploy and achieve quite good results for audio/visual presentations, they often fail in audio-only applications as too many approximations are applied regarding the human auditory perception and propagation of sound waves. This chapter takes a closer look into acoustics and 3D sound rendering with a focus of sonifying 3D virtual/augmented auditory environments. Here not only several techniques are discussed, but also advanced and transferred to a graphics-inspired sound rendering technique to achieve a more realistic and efficient simulation.

The chapter is divided into five sections, of which the first one takes a closer look on the requirements of sound rendering for virtual and augmented auditory environments and discusses the most important concepts and techniques. The following section continues the discussion in the direction of a graphics-based sound rendering and debates the utilization of graphics hardware for general (sound) signal processing. Based on these findings and developments, the following two sections present and discuss ideas for a graphics-inspired sound simulation through a ray- and a wave-based acoustic simulation. As each of these techniques has its respective advantages and drawbacks, the last section discusses concepts towards a possible unification.

8.1 AURALIZATION AND SOUND RENDERING

This first section is dedicated to the auralization of spatial auditory environments and discusses their requirements for sound rendering and auditory design. Auditory environments and augmented audio reality have been introduced and discussed in Chapter 5 and Chapter 6, although in these sections with a focus centered around their definition and application. Section 5.2.2 already motivated the needs for a *non-realistic auditory design* for 3D virtual auditory spaces, at which point this section continues this discussion from a more technical, auralization-centered, perspective.

Essential for an auralization of 3D auditory environments are three *types* of sound:

- Spatialized 3D sound sources,
- Non-spatialized sounds, and
- Room acoustic simulations.

A discussion of their application can be found throughout this research, but is highlighted especially in Section 3.4 and Section 5.2. 3D spatialized sounds are required for all objects with a defined position within a 3D virtual/augmented environment. The spatialization of sound allows later to determine the sounds origin and distance relative to the user, which are both encoded using directional and distance cues. For this task, techniques of HRTF/HRIR convolution (direction) and low-pass filtering (distance) are applied, refer also to Section 3.2.2. Non-spatialized sounds, eg. mono or stereo sounds with an in-head localization, are employed for additional descriptions that are not assigned a specific position within the 3D environment. Examples include the narrators voice from the off, as well as ambient music and feedback sounds from an non-spatial auditory menu system. Room and environmental acoustic simulations are applied to spatialized sounds only, to further integrate and represent them within their local surroundings. As an example, Figure 55 shows an overview of the sound rendering required and also identifies some of the various sound types used. Figure 55 displays 3D positional sound sources as red objects in the scene, as well as shows their direct (red) and secondary (blue) reflections on the room's walls. The two blue objects (window and door) show the position of additional environmental sound objects, which both describe the outer exteriors, eg. the acoustic space on the other side of the window/door.

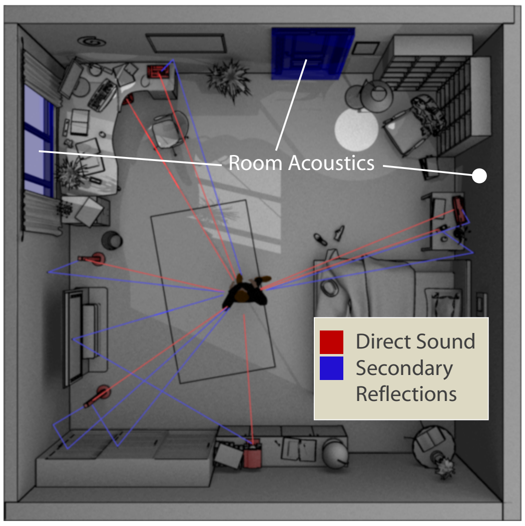


Figure 55: Room Auralization.

An accurate 3D sound rendering along a realistic simulation of a room acoustics is still a very difficult and computationally intensive task. Yet many applications, such as 3D computer games and virtual training scenarios, rely on a realistic acoustic presentation that is able to complement the visual depictions. The difference to audio-only applications and 3D virtual auditory environments is that in these cases the visual image delivers the key information, which is only complemented by the auditory display. That means that an accurate description of an auditory environment requires techniques with a higher quality for both, 3D sound spatialization and acoustic simulation. Almost every application that currently employs either of these techniques is based on 3D sound APIs, with their shortcomings and approximations described earlier (Boer, 2002b). Although the examples and prototypes in this research are also mainly based on OpenAL/EFX and AM3D, this chapter takes a closer look on techniques for a higher quality and more efficient sound rendering and simulation.

The propagations of sound and light seem on a first glance to not have very much in common, but they share, nevertheless, several similarities that can easily be exploited. The following sections examine the possibilities to employ computer graphics techniques and commodity graphics hardware for sound rendering and simulation. Some of the most promising techniques are thereby discussed in more detail and adapted towards a graphics-based sound rendering approach. The motivation for this research is twofold, as on one hand computer graphics and graphics hardware can be used to qualitatively and quantitatively improve existing sound simulations, but also provides a glimpse into a possible future of *programmable* sound hardware. Currently available PC sound hardware is rather fixed in its pipeline and functionality, although some steps were already made into this direction (Aureal, 2000; Creative Labs, 2005). Nevertheless, a fully customizable sound rendering pipeline, similar to the development of graphics hardware over the last decade, would benefit many applications and allow sound simulations and custom effects with a much higher realism by the use of personalized HRTFs, thus increasing the level of immersion in all applications (Röber et al., 2006c, 2007).

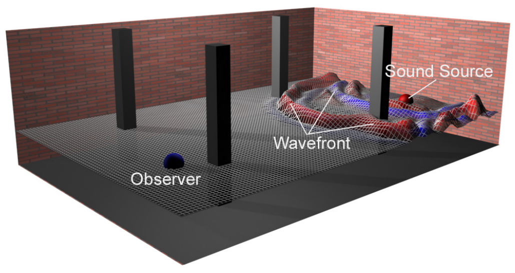
8.1.1 *3D Sound Rendering*

One of the key aspects that is required is an efficient and high-quality 3D spatialization of monaural sound sources. The perceptional and physical principles for this were outlined and discussed in Chapter 3. The focus of this section is to introduce several concepts and techniques to actually perform 3D sound spatialization, with the goal to identify possibilities that can be exploited to improve 3D sound synthesis in both, quality and efficiency.

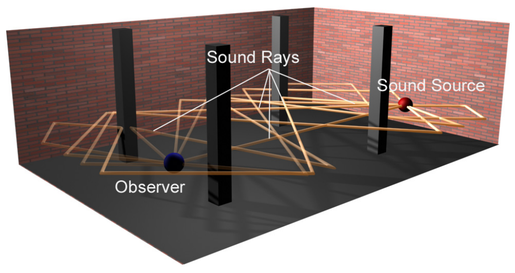
A very interesting approach is here described using so called *perceptual rendering* techniques, which acoustically display the virtual scene tailored to the auditory senses. This approach can, in conjunction with Section 5.2.2, be described as a non-realistic auditory scene design, as it only considers those parts of the environment that are clearly audible from the user's current position. Research in this area has been conducted by Funkhouser et al. and Tsingos and Drettakis, who both looked at certain techniques to increase the richness of 3D auditory scenes by grouping and classifying sound sources depending on their importance and perception (Funkhouser et al., 1999a; Tsingos and Drettakis, 2004; Tsingos, 2007). Their results can be applied to many areas and used to enhance the display and the perception of 3D virtual auditory environments. However, efficient techniques for the spatialization of monaural sounds are still required.

Most 3D sound systems employ here a convolution using HRIR/HRTF filters, which describe the transformation of a monaural sound into a binaural signal representing the source with its current position, orientation and distance. This function is based on sound reflections along/within the torso/shoulder/head system, and therefore exhibits a strong personalization effect. Current HRTFs that are employed in sound hardware and sound spatialization APIs are based on generalizations using so called *standard ears*. This, however, causes several perceptional artifacts and especially front/back and up/down conversions. A solution to this problem would be the measurement or simulation of personal HRIR filters to increase the accuracy and efficiency for 3D source localization. However, some researchers have also shown that listeners can – in certain cases – adopt to non-personalized HRTFs, which allows the development of *training* applications to accommodate the listener's hearing to adjust to given HRTFs (Hofman et al., 1998; Richardson and Kaiwi, 2002).

The personalization of HRIR filters is still an active and not yet fully resolved area of research. As the measurement is complex and requires dedicated equipment, the majority of approaches focus on a simulation using 3D geometrical models (Kahana et al., 1999; Richardson and Kaiwi, 2002). Two techniques exist, which are often applied for sound simulations are the wave-based and the ray-based approach, see also Section 8.1.2. Two often employed techniques are based on either boundary element methods (BEM), or finite element methods (FEM) (Ise and Otani, 2002). However, as the simulations are very complex and time consuming, an alternative solution has to be found. An idea that was already expressed at the beginning of this chapter is the use of computer graphics hardware and graphics-based techniques for the simulation. The remaining sections of this chapter further develop this idea and explore two possibilities to employ graphics technology to enhance the display of 3D virtual auditory environments, both qualitatively and quantitatively.



(a) Wave-based Approach.



(b) Ray-based Approach.

Figure 56: Acoustic Simulation Techniques.

A different approach for 3D sound rendering is the use of ambisonics, or B-Format, for the recording and playback of real/virtual sound fields. The technology was developed in the 1960s and describes a multi-channel recording/playback system that can be used for spatial recordings, but also for the presentation of virtual 3D sound fields (Cooper and Taylor, 1998; Pope et al., 1999). Ambisonics are based on a decomposition of a sound field using spherical harmonics and describes the sound pressure along several different gradients. The first-order B-Format is based on just four channels, but more can be added to increase the accuracy of the system. A great advantage for using ambisonics is that they can efficiently be simulated using computer graphics as well. Dempinski and Viale describe an implementation in 3D computer graphics to illuminate 3D objects using global illumination models (Dempinski and Viale, 2005). The exact same principles, with slight modifications of course, can also be applied for a rendering of 3D sound sources and virtual sound fields, which makes this technology very interesting to enhance any spatial auditory display system.

Besides the rendering and synthesis of 3D sound sources, also the simulation of environmental, or room acoustics is very important. It conveys information about the local surroundings, but it can also be used to detect objects and obstacles in close vicinity. The next section therefore summarizes two of the most widely applied techniques for the simulation of environmental acoustics.

8.1.2 *Acoustic Simulation Techniques*

The two most often used approaches for the simulation of room acoustics are 3D waveguide meshes and ray/beam tracing techniques, see also Figure 56 for a comparative overview. Figure 56a displays a visualization of the waveguide technique, a more physically correct sound propagation model that is based on differential equations. The acoustic energy (eg. pressure) is distributed along nodes using difference equations, which emphasize the applicability of this technique to the simulation of wave-based propagation effects, such as diffraction and interference. Due to its computational high complexity, it is usually only employed for the lower frequency end.

Figure 56b visualizes an alternative approach that is based on energy acoustics and uses ray tracing techniques. This method is based especially on techniques from computer graphics, and is, due to the approximation of sound waves to *sound rays* only applicable to the middle and higher frequency parts.

Over the past years, graphics hardware has inspired several researchers to also deploy it in a large variety of non-graphics applications, including sound rendering and sound simulation (Whalen, 2005; Aszódi and Czuczor, 2002; Jedrzejewski, 2004; Röber et al., 2006c, 2007). Jedrzejewski uses the GPU for simple 2D geometric room acoustics using ray tracing and regular specular reflections (Jedrzejewski, 2004), while Kapralos et al. and Deines et al. employ a particle-based system to adopt the photon mapping technique towards a *phonon tracing* approach (Kapralos et al., 2004; Bertram et al., 2005; Deines et al., 2006b). The aforementioned ray- and wave-based techniques for sound simulation posses a great potential for a hardware-accelerated graphics-based implementation as well. Section 8.3 and Section 8.4 refer to these techniques and discuss a GPU-based implementation that enhances both methods in terms of quality and simulation efficiency.

8.2 SOUND SIGNAL PROCESSING

With the availability of programmable graphics hardware and high-level shading languages, graphics programming moved into the focus of many research communities and improved their scientific computations (Owens et al., 2005). The GPU as the core of current graphics hardware can be characterized as a massively parallel streaming processor that has applications in many research areas. The advantages of a GPU-based implementation for sound rendering and simulation are obvious: Not only that the propagation of light and sound shares several similarities which can be easily exploited, but also because the GPU can be straightforwardly turned into a freely programmable DSP for general (sound) signal processing.

Digital signal processing is concerned with the alteration and modification of digital signals, and involves a frequency-dependent amplification or attenuation of certain parts. Digital filters can perform virtually any operation, but are limited by the filter's cost and execution speed. Several different filter types exist in time-domain sound signal processing, with the most common being linear, causal, time-invariant and FIR filters (Zölzer, 2002). A digital filter can be described by its impulse response or its transfer function in the Z-Domain:

$$y(n) = \sum_{i=0}^{N} h_i x(n-i) \tag{8.1}$$

$$H(z) = \sum_{n=0}^{N} h_n z^{-n} \tag{8.2}$$

Here, Equation 8.1 shows the familiar time-based convolution of an input signal with a finite impulse response (FIR) filter of size $N+1$, while Equation 8.2 displays the filter's transfer function within the Z-Domain. When plotted along the Z-plane, $H(z)$ visualizes the zeros and poles of this filter operation and thereby directly the areas of amplification and attenuation of the frequency response (Zölzer, 2002). This is very useful for the design of digital filters, in which poles and zeros can simply be placed along the unit circle to quickly evaluate the filter's frequency response.

Sound signal processing has many applications for the sonification of 3D virtual auditory environments. Impulse responses are used to convolve dry sound files to create a spatialized, binaural impression (HRIR) and to simulate a room's acoustics (RIR). Due to the large number of sound files, the convolution is thereby required to be performed in the most efficient way possible. Nevertheless, several approximations are available and can be used to reduce the computational cost with a minimal impairment of the signal's accuracy. Currently available sound APIs employ a large variety for *emulating* room acoustics and sound spatialization. OpenAL, for instance, uses low-pass filters and several delay lines to simulate obstruction and occlusion effects (Hiebert, 2006; Peacock et al., 2006). Hall and echo, again with low- and band-pass filters applied, are employed to create an illusion of room acoustics.

Using the earlier discussed advantages of commodity graphics hardware, the following section explores the possibilities for utilizing computer graphics techniques and high-level shading languages for a general (sound) signal processing.

8.2.1 *The GPU as Digital Signal Processor*

The GPU as a stream-based and freely programmable processor is highly suited for a general time-domain signal processing. Two publications that already discuss these possibilities have been presented by Gallo and Tsingos and Whalen (Gallo and Tsingos, 2004; Whalen, 2005). Gallo and Tsingos employ the GPU for 3D sound spatialization and measured a slight increase in performance compared to a regular CPU implementation (Gallo and Tsingos, 2004). Whalen concentrated his efforts on a classic DSP approach and implemented several convolution-based signal processing effects using fragment shaders (Whalen, 2005). A direct comparison with a CPU implementation revealed, however, that the CPU outperformed the GPU at several occasions. The work of both authors is, however, based on earlier GeForceFX GPUs, which also both describe as inefficient for audio processing due to limitations in texture-access modes, shader-length and -complexity, as well as a too slow AGP bus connection (Whalen, 2005; Gallo and Tsingos, 2004). Since then, several improvements have been made and current graphics technology with its unified shader architecture and high-speed PCIe bus connection performs very well for the manipulation and filtering of digital signals.

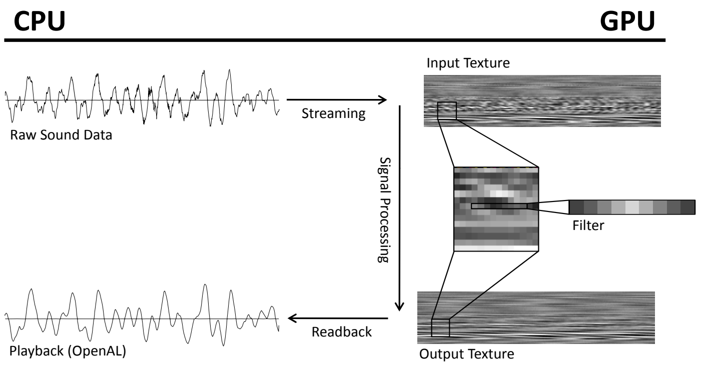


Figure 57: GPGPU-based Signal Processing.

Utilizing the GPU for time-domain DSP filtering operations is very similar to the general GPGPU[1] approach, as is illustrated in Figure 57. The input signal is transferred into graphics memory and loaded as a floating point texture with each texel representing one sample. A second texture holds the convolution kernel, while a third texture is used for the filter result. The convolution itself is implemented using fragment shaders and applied per texel of the input texture.

GPU-based Signal Processing.

A shader example that performs a box average can be seen in Listing 8.1. In this example, `texture` contains the 1D input signal and is stored in a 2D texture array with `length` being half the filter's size. The return value is the original texture, in which the red channel now holds the averaged filter signal. Additional examples for employing the GPU as general DSP can be found in Section A.1, which also discusses shader examples for the later discussed GPU-based sound simulations.

The enormous efficiency advantage of a GPU-based implementation is primarily the result of a parallel computation. This approach, however, does not permit a realization of all types of digital filters in graphics hardware. Some have to be implemented as several

1 GPGPU = General Purpose computations using a Graphics Processing Unit (GPU)

```
float4 convolution(float2 coords : TEX0, uniform sampler2D texture) : COLOR
{
  float4 s1 = tex2D(texture, coords);

  //----- median filtering with size 2*length+1 -------//
  float tmp = 0.0;
  for (int i=(-1*length) ; i<(length+1) ; i++)
    tmp += (0.5-tex2D(texture, sample(coords, i)));

  float tmp2 = 1.0 + (((length*2)+1) / 100.0);
  s1.r = saturate(0.5+(tmp2*(tmp / (1+(2*((length*2)+1))))));

  return s1;
}
```

Listing 8.1: GPU Signal Processing (Convolution).

rendering passes, or require a pass-in of additional parameters/computations performed on the CPU (Micea et al., 2001; Röber et al., submitted). Table 6 shows the efficiency of several examples, which have been implemented and tested on the CPU, as well as in graphics hardware using fragment shaders and the approach displayed in Figure 57. The efficiency is measured in *fps*, and refers to the number of sound samples (44.1kHz resolution, 1s length) processed per second. It shows the raw processing efficiency only, without data streaming or any other computations. Two GPUs were evaluated to additionally assess the differences in available graphics hardware.

The results in Table 6 show clearly that older graphics technology is not always able to compete with the CPU (Whalen, 2005), but also that newer graphics hardware with its unified shader architecture possesses enormous computational capacities.

Time-domain signal processing filters are easy to implement and can also be combined with sound simulation techniques to perform a binaural sound rendering. However, convolutions with large kernel sizes are still very time consuming. A possible solution is a filtering in the frequency domain, at which the following section discusses an alternative approach using an implementation of frequency band decompositions and a signal filtering with adjustable frequency weights.

Filter Type	CPU-based	GPU-6800GT	GPU-8800GTX
Volume	3,400 fps	11,200 fps	90,500 fps
Normalize	3,150 fps	10,600 fps	96,250 fps
Convolution (5)	1,600 fps	1,320 fps	43,500 fps
Convolution (25)	501 fps	300 fps	11,300 fps
Convolution (125)	90.5 fps	64.1 fps	2,560 fps
Pitch / Resampling	1,700 fps	1,880 fps	58,500 fps
6-Tap Delay	1,450 fps	1,600 fps	25,800 fps
Chorus / Flanger	2,650 fps	3,600 fps	70,000 fps
Compressor / Limiter	820 fps	1,120 fps	30,300 fps

Table 6: CPU vs. GPU - Signal Processing Efficiency (fps per 44.1kHz/1s).

f_j	f_{range_j}	f_{center_j}	λ_{center_j}
f_0	22 Hz — 44 Hz	31.5 Hz	10.88 m
f_1	44 Hz — 88 Hz	63 Hz	5.44 m
f_2	88 Hz — 177 Hz	125 Hz	2.74 m
f_3	177 Hz — 354 Hz	250 Hz	1.37 m
f_4	354 Hz — 707 Hz	500 Hz	0.68 m
f_5	707 Hz — 1,414 Hz	1,000 Hz	0.343 m
f_6	1,414 Hz — 2,828 Hz	2,000 Hz	0.172 m
f_7	2,828 Hz — 5,657 Hz	4,000 Hz	0.086 m
f_8	5,657 Hz — 11,314 Hz	8,000 Hz	0.043 m
f_9	11,314 Hz — 22,627 Hz	16,000 Hz	0.021 m

Table 7: Frequency Bands f_j.

8.2.2 *Frequency-based Filtering*

Two approaches are applicable for a frequency-dependent sound signal processing, either with or without the use of a Fourier transform. The straightforward approach uses complex textures and a CPU- or a GPU-based FFT for a frequency domain conversion of the signal data (Ritter et al., 1999; Govindaraju and Manocha, 2007). The phase and amplitude are stored within complex textures and can be processed and manipulated using fragment shaders, analog to Figure 57. After filtering and an inverse transformation, the signal data is streamed back to main memory for further processing or playback. An alternative approach is to decompose the signal in a pre-processing step using windowed sinc filters into several frequency bands (Table 7), and to process each of these bands individually using a frequency-dependent weighting function. This still allows an efficient processing and implementation, but in respect to a filters frequency behavior and without the necessity of time-consuming FFT conversions.

Frequencies of the audible spectrum are classified and described by frequency bands (octaves) according to human psychoacoustics (Vorländer, 2007; Goldstein, 2007). Table 7 provides an overview of the different frequency bands, along their index number, frequency range f_{range_j}, center frequency f_{center_j} and center wavelength λ_{center_j}. The audible spectrum is therefore defined as the sum of these 10 frequency bands:

$$A_{spectrum} = A_s = \sum_{j=0}^{9} f_j \tag{8.3}$$

HRIR convolutions are a good application to describe this filtering approach. HRIRs are a collection of FIR filters that are dependent on direction, distance and time, and are used for 3D sound spatialization. Prior to the binaural rendering of a scene, all HRIRs and footage sounds need to be decomposed into these 10 frequency bands. Later, the acoustic energy of each direction is filtered and delayed with its associated HRIR. In a second step, the energy of all bands are accumulated according to Equation 8.3. Compared with the last section, this approach is preferable if a frequency-dependent weighting is required. Besides the pre-processing step for the initial signal decomposition, this technique is still very efficient, easy to implement, and allows a frequency-dependent modeling of materials and sound/object interactions.

8.3 3D WAVEGUIDE TECHNIQUE

As outlined in Section 8.1, the waveguide technique is a commonly employed method in room acoustics to simulate the propagation of sound waves using numerical techniques and time-domain difference models. Waveguides have been originally developed for the simulation of string-based musical instruments (VanDuyne and Smith, 1993; Smith, 1992), and were later also applied to model the vibrations of air in room acoustic simulations (Savioja et al., 1995). Waveguides and 3D waveguide meshes are very well suited for the simulation of sound wave propagation, but require a huge amount of sampling points (nodes) in order to achieve realistic results. This makes this approach technically only applicable to the lower frequency end, as with higher frequencies the necessary sampling distance decreases and the number of nodes required rises cubically.

As motivated throughout the last chapters, the quality and efficiency of sound rendering is of the highest importance, especially for an acoustic display of 3D auditory environments. Here not only techniques for 3D sound spatialization are required, but also methods for a realistic simulation of room acoustics and an auditory presentation of the local environment. The waveguide technique offers here a promising approach that can be improved in terms of quality *and* efficiency by using computer graphics and commodity graphics hardware. Current graphics applications typically require the processing of huge amounts of data, for which graphics hardware has been optimized to support using a highly parallel design. This makes this hardware, along with its high-level programming techniques, very interesting for parallel computing problems, such as wave propagations using waveguide meshes. The idea is to develop a technique that allows a fast and – if possible – real-time implementation of 3D waveguide meshes for the lower and mid frequency ranges. The simulation results can then directly be applied for an environmental presentation of 3D auditory spaces. Using additional techniques for a non-realistic auditory design, this setup should provide the environmental information required to navigate and orient oneself through 3D virtual auditory environments.

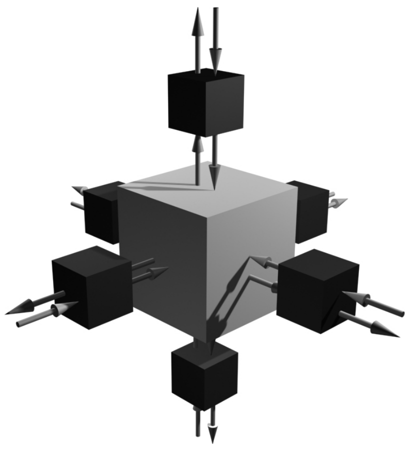


Figure 58: 3D Waveguide Node.

8.3.1 *Waveguide Meshes*

Waveguide meshes are an extension of the waveguide technique and constructed by bi-linear delay lines that are arranged in a mesh-like structure (VanDuyne and Smith, 1993). Each node in the mesh is defined as scattering junction and acts as spatial and temporal sampling point for the wave and energy propagation. Scattering junctions are thereby of equal impedance with two main constraints in effect:

- The sum of all inputs is equal to all outputs, and
- The pressures in each crossing waveguide are equal at the junction.

By assuming a lossless scattering, the *acoustic pressure* v_J is determined by adding all incoming wave components v_i^+ according to Equation 8.4 (VanDuyne and Smith, 1993).

The relationship between the incoming v_i^+ and outgoing v_i^- components is expressed by Equation 8.5.

$$v_J = \frac{2\sum_i R_i v_i^+}{\sum_i R_i} \tag{8.4}$$

$$v_i^- = v_J - v_i^+ \tag{8.5}$$

For a homogenous N-dimensional rectilinear mesh, in which each junction connects to 2N neighbors, Equation 8.4 is reduced to:

$$v_J = \frac{\sum_i v_i^+}{N} \tag{8.6}$$

By discretizing time and space one obtains the difference equations that govern the wave propagation within an N-dimensional rectangular mesh with:

$$v_{J,k} = \frac{\sum_l v_{J,l}(n-1)}{N} - v_{J,k}(n-2) \tag{8.7}$$

In this equation, k identifies the current node, n represents the discretized time steps and l is associated with neighboring nodes. In order to simulate boundary conditions, so called 1D termination nodes with only one neighbor are employed to simulate phase-reversing and -preserving reflections, but also non-reflective, anechoic walls (Savioja et al., 1995). Certain atmospheric absorption effects can be emulated using an additional factor, but are here ignored due to an application in interior acoustics only.

A major problem with digital waveguide meshes is a non-isotropic speed for energy propagation. This is also known as frequency dispersion and varies between different mesh topologies. The dispersion error for the rectilinear grid ranges from zero along the diagonals to its maximum extent along the coordinate axes, and is quantified as:

$$k_d(\underline{\beta}) = \frac{c'\underline{\beta}}{c} \tag{8.8}$$

with c being the speed of propagation in a dispersion free environment, and $c'\underline{\beta}$ the actual speed in the direction of $\underline{\beta}$. The analytical expressions for k_d can be derived using Von-Neumann analysis (Bilbao, 2004; Campos and Howard, 2005; Fontana and Rocchesso, 2001), and show that the dispersion error for the 3D rectilinear mesh ranges on a spherical surface between $0.927 < k_d < 1$, with $k_d = 1$ along the diagonal axes. This causes a distortion of the initially spherically bandlimited signal along the coordinate axes, which also impairs the simulation results.

8.3.2 *Optimal Sampling*

The rectilinear Cartesian lattice is in terms of sampling efficiency not the most optimal grid available. Under the assumption of an isotropic spherically bandlimited signal, hexagonal lattices provide a much better packing density. A denser packing of spectra in the frequency domain translates to an increase in sampling distance in the spatial domain. The Body-Centered-Cubic grid (BCC) represents such a hexagonal lattice, see also Figure 59b. A direct comparison with the Cartesian grid in Figure 59a reveals that a BCC node has 8 nearest neighbors and is constructed with an additional sampling point in the cell center. Due to the more optimal sampling, the internodal distance can be increased to $\sqrt{1.5}$, which in 3D results in roughly only 70 % of the sampling points required to represent the same information (Conway and Sloane, 1976). Hexagonal grids are also very common in nature, as bees, for example, build their honeycombs using hexagonal cells and as a result achieve a minimum expenditure of wax.

An advantage of the BCC grid is that it can be decomposed into two cubic grids that intertwine and are offset by half the sampling distance, see Figure 59b. More generally, the N dimensional hexagonal lattice can be constructed by two N-1 rectilinear grids that are shifted by $\sqrt{2}$ in all N dimensions. This characteristic allows an easy indexing of the data and is also of high importance for a later implementation in graphics hardware. The BCC lattice is already known and used in computer science for image processing and scientific visualization (Theußl et al., 2001b; Röber, 2002), but has its main roots in chemistry and crystallography (Jackson, 1991; Wells, 1984a).

The BCC lattice offers several advantages for the simulation of room acoustics using digital waveguide meshes. The inherent optimal sampling requires only about 70 % of the original sampling points, and thereby directly reduces the computational load and data storage required. Additionally, as the BCC grid has 4 delay units per node, a different and lower frequency dispersion error can be expected.

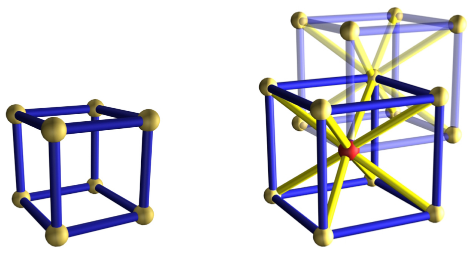

(a) Cartesian Lattice. (b) BCC Lattice.

Figure 59: Cartesian and Body Centered Cubic Lattice.

The equations that govern the propagation of sound waves using the BCC lattice are based on difference equations derived from the Helmholtz equation by discretizing time and space. With now 4 principal axes they are transformed to:

$$\begin{aligned} p(t+1,w,x,y,z) &= \\ &\tfrac{1}{4}[p(t,w+1,x,y,z)+p(t,w-1,x,y,z) \\ &+p(t,w,x+1,y,z)+p(t,w,x-1,y,z) \\ &+p(t,w,x,y+1,z)+p(t,w,x,y-1,z) \\ &+p(t,w,x,y,z+1)+p(t,w,x,y,z-1)] \\ &-p(t-1,w,x,y,z) \end{aligned} \tag{8.9}$$

in which p is the pressure at point (w,x,y,z) at time step t. Because of the increased sampling distance, the unit length in the BCC lattice is extended to $\sqrt{1.5}$, which also changes the update frequency f_{update} to:

$$f_{update} = \frac{c\sqrt{2}}{\Delta x} \approx \frac{480.8}{\Delta x}\ \text{Hz} \tag{8.10}$$

As a result, the BCC lattice propagates sound waves *faster* than the rectilinear Cartesian lattice, which has to be considered in virtual impulse response measurements. The

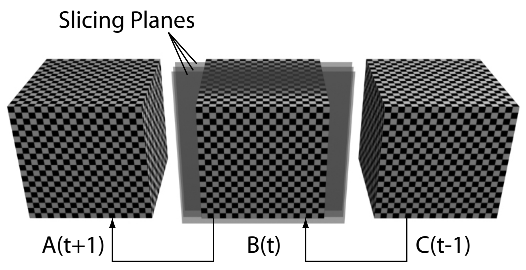


Figure 60: 3D Waveguide Mesh – Rendering Principle.

frequency dispersion factor k_d has for a spherical surface a range of $0.953 < k_d < 1$ (Campos and Howard, 2005). At its maximum extent, the error is only 4.7 %, compared to the 3D rectilinear grid with 7.3 % in the direction of $\beta_n = \pi/2$. The spatial bandwidth can be determined by decomposing the BCC lattice into two rectilinear grids of spacing d. The sampling efficiency of the BCC lattice compared to the 3D rectilinear grid therefore is:

$$\frac{\mu_{BCC}}{\mu_{CC}} = \frac{1}{\sqrt{2}} \approx 0.707 \tag{8.11}$$

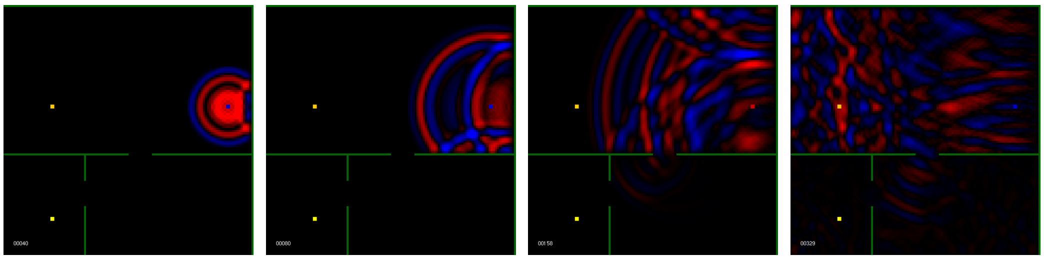
Although, in a direct comparison with the rectilinear Cartesian grid, the computational effort per node is slightly higher, overall it is still much more efficient and also exhibits a less pronounced dispersion error.

8.3.3 *GPU-based Implementation of 3D Waveguide Meshes*

) 3D waveguide meshes are easy to implement and realize in graphics hardware using a high-level shading language. The technique is mainly based on 3D 32-bit floating point textures, fragment shaders, as well as OpenGL's framebuffer objects (FBO). The BCC waveguide mesh can be decomposed into two rectilinear 3D textures, which are loaded and stored separately into texture memory. Both grids with time frames $t-1$ and t reside here in just one single RGBA texture. The base grid is loaded into the *Red* and *Green* channel, while the offset grid is placed in *Blue* and *Alpha*. This allows to compute two nodes, eg. one BCC cell, in one step. The channels are directly rendered into a framebuffer object in an alternating fashion, having one texture attached to it as the primary render target. Figure 60 visualizes the method's principle. The three textures A,

Mesh Size	2D-CC CPU	2D-CC GPU	3D-CC CPU	3D-CC GPU	3D-BCC GPU
$64 \times 64 \times 24$	32,000 fps	9,800 fps	238.0 fps	1,358.0 fps	1880.0 fps
$128 \times 128 \times 24$	7,500 fps	6,330.0 fps	16.1 fps	990.0 fps	1240.0 fps
$256 \times 256 \times 24$	2,000 fps	5,024 fps	4.1 fps	322.0 fps	430.0 fps
$512 \times 512 \times 24$	456.6 fps	2,670.0 fps	0.9 fps	88.3 fps	121.0 fps
$768 \times 768 \times 24$	213.7 fps	1.377 fps	0.28 fps	39.9 fps	55.2 fps
$1024 \times 1024 \times 24$	123.06 fps	830.0 fps	– fps	19.1 fps	31.6 fps
$2048 \times 2048 \times 24$	29.93 fps	221.0 fps	– fps	– fps	– fps
$4096 \times 4096 \times 24$	7.9 fps	56.5 fps	– fps	– fps	– fps

Table 8: Waveguide Mesh Efficiency – CPU vs. GPU (fps).



(a) Wavefront at $t = 40$. (b) Wavefront at $t = 80$. (c) Wavefront at $t = 160$. (d) Wavefront at $t = 320$.

Figure 61: 3D Waveguide Meshes (BCC Lattice).

B and C contain the waveguide node data at their respective time frames. Texture B is *sampled* using texture aligned slicing planes that have the same resolution as the texture itself. During this sampling, and for every time frame, the fragment shader solves the difference equations according to Equation 8.9 for each voxel in the waveguide mesh and stores the results into the next buffer (eg. A).

A second channel contains additional scene information, such as scene geometry and material specifications to model basic boundary conditions. An example of the fragment shader discussed can be found in Section A.2, which shows the implementation using both BCC nodes, an implementation of a sound source, as well as simple phase-reversing reflections.

8.3.4 *Results and Efficiency*

Several experiments have been performed to evaluate and assess the quality and efficiency of a graphics-based implementation of 3D waveguide meshes. The experiments have been performed on a regular PC equipped with a P4 3GHz processor, 1GB of main memory and an *n*Vidia GeForce 8800GTX graphics accelerator. The performance results can be seen in Table 8, although some results are missing due to insufficient texture memory and CPU performances. For comparison reasons, also the efficiency of a simple, non-optimized CPU implementation is shown. The results clearly show the advantages of a graphics-based implementation, as especially for larger meshes, the CPU is outperformed by a substantial factor. Table 8 also shows that the implementation of the BCC lattice is faster by a factor of approximately $\sqrt{2}$, the number of samples *saved* due to the more optimal sampling used. An earlier implementation of this approach was impaired by the non-availability of 3D)framebuffer objects, which massively decelerated the performance due to a forced computation in 2D and a heavy texture copying (Röber et al., 2006c). Although wave-based room acoustics can be computed much faster and with the same accuracy using this technique, one limitation that applies is the texture memory available. Dedicated GPGPU hardware provides today up to 1.5GB of RAM, and allows thereby with this technique for over $100,000,000$ waveguide nodes (Nvidia, 2007). This should be sufficient for the modeling of most scenarios and can even be pushed further using more efficient texture packing techniques.

3D Waveguides CC.

3D Waveguides BCC.

Figure 61 shows several time frames for a setting of three rooms with reflecting walls and ceiling. It was used to compare the quality, as well as the efficiency of both implementations. The sound source is marked by a blue dot, the two microphones by yellow dots, while walls are identified by green color. The data set in this example has a size of $128 \times 128 \times 24$ for the rectilinear Cartesian grid and a size of $92 \times 92 \times 34$ for the BCC lattice. The sound source, walls and microphone positions are adjusted accordingly to fit the BCC's dimensions. The rendering efficiency is for the rectilinear grid on average of 990 fps, and for the BCC lattice 1,240 fps. Both meshes were excited using a single sine wave, whose frequency was adjusted for the BCC lattice according to Equation 8.10.

After this detailed examination of wave-based acoustic simulations, the following section explores the applicability of a graphics-based implementation for ray-based acoustic simulations.

8.4 RAY/ENERGY ACOUSTIC SIMULATIONS

Geometrical acoustics is often referred to as ray/energy acoustics, and is in this respect very similar to optical models used in computer graphics. Ray acoustics thereby approximates sound waves as particles that are moving along directional rays and adopts existing ray tracing techniques for sound simulation. Due to this approximation, wave-based propagation effects, such as diffraction and interference, are usually not considered and therefore not part of the simulation. Ray acoustics is therefore only applicable to frequencies, whose wavelength are much shorter than the dimensions of the enclosure or any object within (Everest, 2001; Kuttruff, 2000).

Ray tracing is a long known area of research in computer graphics and has seen many improvements and enhancements. Lately, with the introduction of dedicated hardware, ray tracing shifted from an offline simulation towards an interactive and realtime rendering system (Purcell et al., 2002; Purcell, 2004; Moreno-Fortuny, 2004). Many advancements from the visual realm can also be mapped and beneficially applied to ray acoustics simulation as well. Similar to the last section, computer graphics and graphics hardware can aid ray-based sound simulations to perform faster and with a higher quality. However, due to several differences in light and sound wave propagation, spectral and temporal effects have to be integrated into the simulation model and accounted for by the ray tracing system.

Several articles about ray/energy acoustics have been published, of which some already discuss the realtime possibilities of a ray-based acoustic simulation system (Funkhouser et al., 2002c; Savioja et al., 2002; Tsingos and Drettakis, 2004). The majority, however, concentrates only on specular reflections using a ray/beam tracing-based approach and uses conventional 3D sound APIs for sound spatialization and rendering (Wand and Straßer, 2004; Neumann, 2004; Jedrzejewski, 2004).

To the author's knowledge, none of the existing ray-based sound simulations were so far examined regarding an application of global and local illumination models towards ray/energy acoustics in greater detail. The next sections therefore provide an in-depth analysis of computer graphics and ray tracing techniques, and concentrate especially on building a foundation for ray/energy acoustics by extending models used in computer graphics towards a time- and frequency dependent acoustic energy propagation.

8.4.1 *Acoustic Energy Propagation*

Sound is defined as mechanical energy and propagates through pressure variations within a participating media, and can be described by attributes such as frequency, wavelength and speed of propagation. Light on the other hand is an electromagnetic radiation that is characterized by similar, however, largely different quantities. In order to study and describe the propagation of sound waves using ray tracing techniques, an adequate propagation model that includes time- and frequency dependencies needs to be defined. Such a model can be developed in analogy to the physics of light transportation by using and extending the tools of radiometry to also include spectral and temporal propagation effects (Dutre et al., 2003; Beranek, 1986).

Acoustic energy is described as the amount of pressure variations per unit volume and time, or, more precisely, by the changes in velocity of air particles contained in a volume element per unit time. Acoustic energy is quantitatively measured in $Watt$ or $Joule/sec$ and described as radiant power Φ or flux (Dutre et al., 2003). The intensity is thereby defined as the amount of acoustic energy that flows from/to/through a surface element per unit time:

$$I(t) = \frac{d\Phi}{dA}dt \tag{8.12}$$

The energy density in the medium of propagation is defined as the sum of its kinetic and potential energy per unit volume dV and time: $E(t) = E_{kin}(t) + E_{pot}(t)$ (Beranek, 1986). The kinetic energy density, or sound wave pressure, is therefore described as:

$$E_{kin}(t) = \frac{1}{2}\frac{Mc^2}{V_0}dt = \frac{1}{2}\rho_0 c^2 dt \tag{8.13}$$

with c being the average velocity of air particles, ρ_0 the average media density and $\frac{M}{V_0}$ the mass per unit volume. The potential energy density can be derived from the gas law as:

$$E_{pot}(t) = \frac{\int pdp}{c^2\rho_0}dt = \frac{1}{2}\frac{p^2}{c^2\rho_0}dt \tag{8.14}$$

with p as the pressure of the sound wave and c the speed of sound in this medium. The total amount of acoustic energy density is therefore described as (Beranek, 1986):

$$E(t) = E_{kin}(t) + E_{pot}(t) = \frac{1}{2}(\rho_0 c^2 + \frac{p^2}{c^2\rho_0})dt \tag{8.15}$$

With Equation 8.15 being sound at any position and time within the virtual auditory environment, it serves as basis for defining an acoustic energy propagation model. In order to quantitatively measure flux per unit projected surface area and per unit angle, radiance is introduced with (Dutre et al., 2003):

$$L(x,\Theta) = \frac{d^2\Phi}{d\omega dA\cos\theta} \tag{8.16}$$

and varies with position x and the ray's direction Θ. By incorporating the wavelength λ_j of the frequency bands used (refer to Table 7), Equation 8.16 changes to:

$$L(x,\Theta,f_j) = \int_{A_s} L(x,\Theta,f_j)d\lambda \tag{8.17}$$

The acoustic energy that interacts with a surface element is further differentiated in incident E_i (incoming) and exitant E_e (outgoing) energy:

$$E_i = \frac{d\Phi}{dA}, E_e = kE_i \tag{8.18}$$

The scalar k (defined over $[0,1]$) hereby describes the reflectivity of a surface element with $E_{surface} = E_i - E_e$, and is affected by the surface material definitions. Using a lossless participating media, the exitant radiance at one point $L(x_1 \to \Theta)$ is exactly the same as the incident radiance at another point receiving this amount of energy $L(x_2 \leftarrow \Theta)$ (Dutre et al., 2003). This approach of reciprocity is also sound in real world acoustics and employed in highly efficient *inverse* HRIR measurements (Li et al., 2004). Using a density function and volume elements, $p(x)dV$ defines the physical number of *sound particles* that are carrying an *acoustic energy quant*. If moved in time dt across a differential surface area dA, and by using the direction ω and speed of propagation c:

$$N = p(x, \omega, f_j)\ c\ dtdA\ \cos\theta\ d\omega d\lambda \tag{8.19}$$

describes the number of particles flowing through this surface element. The radiance per unit volume is accordingly redefined to:

$$L(x, \Theta, f_j) = \int_{A_s} \int p(x, \omega, f_j) h \frac{c}{\lambda_j} d\lambda \tag{8.20}$$

In this model, acoustic energy radiates from a sound source (speaker) using a certain emittance pattern. This pattern can be homogenous in any direction (eg. spherically), or direction dependent (eg. cone shaped). Similar to light, acoustic energy also attenuates with distance using the familiar inverse square law and through atmospheric absorption effects. For small enclosures, this factor can safely be ignored, but becomes more prominent with increasing distances. To sample the acoustic energy present at a certain location, an observer, or listener is required. This listener does not interfere or participate in the energy propagation, but, if required, such as in a binaural listening simulation, additional geometry can be used to emulate head-shadowing effects.

8.4.2 *Local acoustic Energy Exchange*

The most interesting part in a ray-based acoustic simulation is the interaction and exchange of acoustic energy with objects and surface elements. Depending on the objects size and the acoustic material parameters specified, some of the incoming energy might get absorbed, reflected, refracted or transmitted, with the total amount of energy, according to Equation 8.18, being constant. Figure 62 shows a schematic of the local acoustic energy exchange. Every ray that is cast into the scene contains, depending on the sound source emittance, energy from all frequency bands. The contribution of each ray is evaluated at the point of intersection with the surface element using the ray's length, its frequency spectrum, as well as the surface material properties defined.

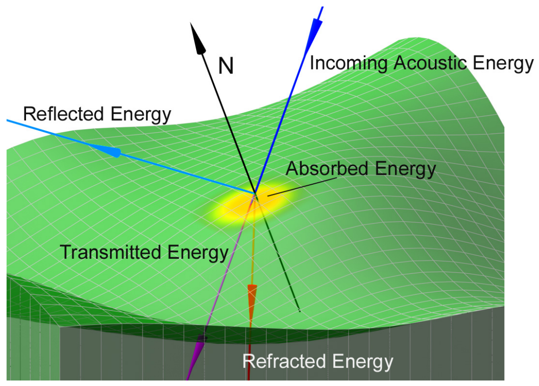
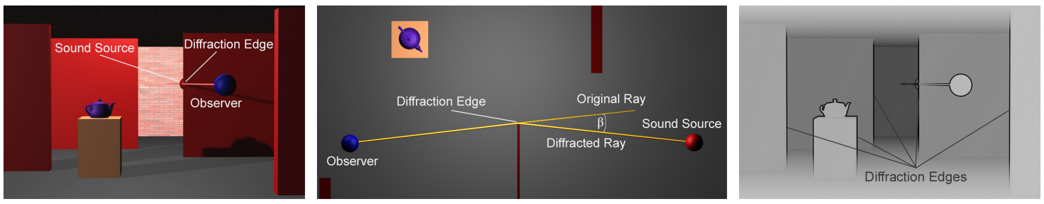


Figure 62: Acoustic Energy Exchange.

Parts of the incident energy are usually absorbed and *removed* from the system. The absorption is frequency dependent and characterized through a coefficient per frequency band α_{f_j}:

$$L_{e_{absorbed}}(x \leftarrow \Theta) = \sum_{j=0}^{9} E_{i_j} \alpha_{f_j} \tag{8.21}$$

The diffraction of sound waves is also frequency dependent. Sound waves are thereby simply bend around objects smaller than their wavelength, but continue unchanged otherwise. In this case, transmission is redefined to describe the amount of energy



(a) Normal Scene Rendering. (b) Top View – original and diffracted Ray. (c) Combined Depth/Edge Map.

Figure 63: Ray Acoustic Diffraction Simulation.

that passes *through* an object unaltered and without refraction. A frequency dependent modeling can be implemented similar to Equation 8.21, which describe the transmission of acoustic energy whose wavelength is equal or above a certain cutoff wavelength set by the objects bounding box:

$$L_{e_{transmitted}}(x \to (\pi + \Theta)) = \sum_{j=0}^{9} E_{i_j} \tau_{f_j}. \tag{8.22}$$

Here $L_{e_{transmitted}}(x \to (\pi + \Theta))$ describes the amount of exitant energy per ray for all bands, which simply passes along the direction opposite to the incoming ray, ie. the ray's original direction. The term τ_{f_j} is used for a finer modeling and a frequency-weighting of the transmission effects.

Reflection and diffuse scattering are probably the two most important qualities in an acoustic ray tracing simulation system, and can be very well described using bidirectional reflection distribution functions (BRDF) (Dutre et al., 2003). A BRDF is defined for a point x as the ratio of the differential radiance reflected in an exitant direction Θ_e and the differential irradiance incident through an incoming angle Θ_i:

$$brdf_{reflected}(x, \Theta_i \to \Theta_e) = \frac{dL(x \to \Theta_e)}{dE(x \leftarrow \Theta_i)} \tag{8.23}$$

The BRDF is thereby frequency dependent, but direction independent, eg. $f_r(x, \Theta_i \to \Theta_e) = f_r(x, \Theta_e \to \Theta_i)$ (Dutre et al., 2003; Li et al., 2004). Diffuse scattering hereby uniformly reflects the incoming acoustic energy in all directions. In acoustics, this behavior is largely influenced by the surface roughness μ, which can be used to derive a specular reflection coefficient that describes the ratio between specular and diffuse reflections. A frequency dependent BRDF for acoustic ray tracing includes all frequency bands, and can be described through:

$$L_{e_{reflected}}(x \leftarrow \Theta_i) = \sum_{j=0}^{9} E_{i_j} \upsilon_{f_j} clamp(\frac{\mu}{f_j}, 0, 1) \tag{8.24}$$

in which υ_{f_j} is an additional weighting factor per frequency band f_j. True refraction effects can be computed similarly to Equation 8.23, in which the outgoing angle Φ of the refracted ray can be determined using *Snell's Law*. A frequency band weighted refraction can be defined in a similar way to Equation 8.24.

8.4.3 *Acoustic Ray Tracing and Diffraction Modeling*

Figure 64 shows the auralization pipeline of the graphics-based ray acoustics system. The 3D scene geometry is converted into a uniform grid structure in a pre-

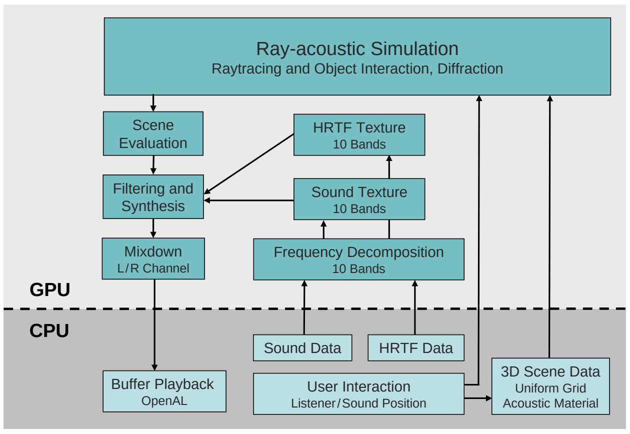


Figure 64: Auralization Pipeline.

processing step. It subdivides 3D space and groups neighboring triangles in an axis aligned uniform voxel-based topology (Purcell et al., 2002; Purcell, 2004; Moreno-Fortuny, 2004). This re-structuring is necessary for a more efficient ray/object intersection testing, as now only triangles from the same voxel element have to be evaluated. In a second step, footage sounds and HRIR filters are loaded as floating point textures into graphics memory and are decomposed using windowed sinc filters with their cutoff frequencies set to the bands respective border frequencies, refer also to Section 8.2.2 and Table 7. Furthermore, sound data is assigned to virtual speakers as well as a position and an emittance pattern.

The actual casting of rays and the energy accumulation is performed using cube maps that are centered around the listeners position (Kaminski, 2007), refer to Figure 65. These cubemaps sample the scene with one ray cast per cubemap texel. Each ray is thereby traced through the virtual scene and its acoustic energy is accumulated and stored per frequency band within the cubemap texel. At points of ray/object intersection, the local surface acoustic energy exchange is evaluated according to Section 8.4.2. Secondary rays, emanating from points of refraction, transmission and/or reflection, are further traced until their energy contribution falls below a certain threshold ϵ. The final sound mixdown is performed binaurally using the previously decomposed sound data and HRIR filters. According to the accumulated energy spectrum in the cubemap texture, the sound data is weighted and delayed per frequency band. The frequency bands of all contributing rays are accumulated and stored in a two channel texture (binaural sound buffer), refer to Figure 64. This sound data is streamed back to the CPU and fills a native OpenAL stereo buffer for playback. All convolutions, simulations, ray tracing, mixdown and sound rendering are performed using fragment shaders in graphics hardware, with a single shader for each task. Several of these shaders and additional code examples are discussed in more detail in Section A.3.

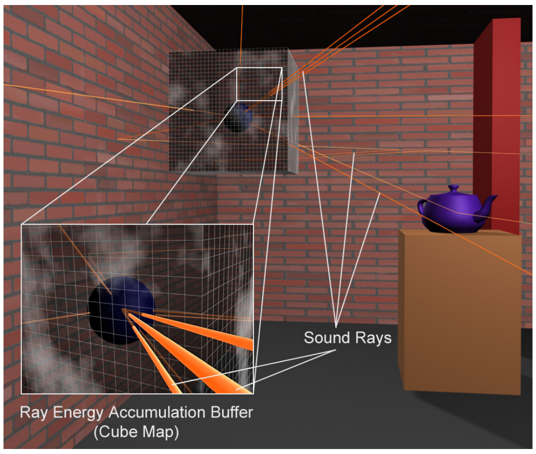


Figure 65: Cubemap ray tracing/Sampling.

Diffraction and interference are both two phenomena that can be well modeled using wave-based simulation techniques, but are difficult to capture using ray-based approaches. Frequency-dependent diffraction effects, however, can be approximated using a ray bending technique, in which the outgoing ray is bent according to the ray's associated frequency band f_j. Here, lower frequencies diffract stronger than higher frequency bands.

		Size of Cubemap				
		8×8	16×16	32×32	64×64	128×128
Possible Number of Directions		384	1,536	6,144	24,576	98,304
Simple Box (80 pol.)	ray tracing only	75.3 fps	72.5 fps	70.2 fps	68.4 fps	63.2 fps
	with auralization	23.1 fps	9.7 fps	3.4 fps	0.97 fps	0.27 fps
Church (800 pol.)	ray tracing only	55.1 fps	44.3 fps	37.8 fps	24.9 fps	18.9 fps
	with auralization	12.8 fps	6.2 fps	2.6 fps	0.77 fps	0.24 fps
Apartment (1,400 pol.)	ray tracing only	43.1 fps	42.1 fps	42.5 fps	34.2 fps	26.8 fps
	with auralization	15.8 fps	7.5 fps	3.1 fps	0.88 fps	0.25 fps
KEMAR (5,500 pol.)	ray tracing only	16.8 fps	16.4 fps	16.4 fps	16.4 fps	16.0 fps
	with auralization	11.2 fps	6.7 fps	3.0 fps	0.89 fps	0.25 fps
Large Hall (37,000 pol.)	ray tracing only	4.2 fps	4.2 fps	4.2 fps	4.2 fps	4.1 fps
	with auralization	3.7 fps	3.0 fps	1.9 fps	0.76 fps	0.23 fps

Table 9: Ray Acoustics Efficiency (fps per 44.1kHz).

The amount of diffracted energy is determined individually per frequency band and depends on the band's maximum diffraction angle. Figure 63 exemplifies the concept and shows a virtual scene from the listener's perspective (Figure 63a), the constructed diffraction/edge map (Figure 63c) and the by β diffracted ray from the listener to a sound source (Figure 63b). The edge map in Figure 63c is constructed by using the scene's depth buffer and an image based edge detection algorithm, in which for each edge additional rays are cast into the scene to perform the diffraction simulation.

Demo Reflection.

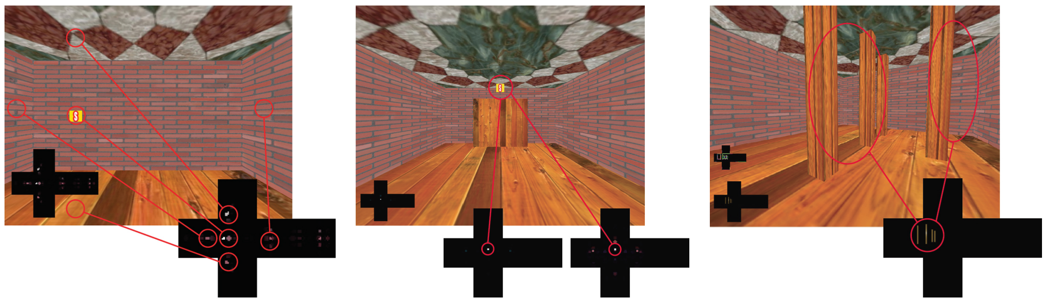
Demo Refraction.

Demo Diffraction.

Interference describes the superposition of two or more sound waves and the subsequent changes in amplitude. By employing a ray-based acoustic simulation, interference effects can only roughly be approximated using the ray's length and its center wavelength λ_{center_j} (Table 7). Although such a system only allows coarse approximations, the results can clearly enhance the simulation and provide a more realistic virtual auditory environment. More and finer subdivided frequency bands will, nevertheless, improve both techniques and also allow a finer modeling of frequency-dependent material definitions.

8.4.4 *Results and Experiments*

Several experiments and tests have been performed to evaluate the quality and the efficiency of this approach. A more detailed discussion of examples can be found in Section 9.7, which discusses and compares several room acoustics and virtual HRIR simulations. KEMAR is a dummy head model that is generally used in acoustics to measure head-related impulse responses. The results of this simulation can be found in Section 9.7.2. All evaluations have been performed on the same computer that was used for assessing the efficiency of the 3D waveguide mesh implementation. Table 9 shows the evaluation results with the number of frames per second (fps) for five different environments and five cubemap sizes, as well as with and without auralization applied.



(a) Reflections (Direct, 1st and 2nd). (b) Different Material Settings. (c) Diffraction around Pillars.

Figure 66: Modeling of Sound Wave Propagation Effects.

In acoustics, the term *realtime* is defined as an update rate of 10Hz or more (Funkhouser et al., 1999a), which the system currently achieves for meshes of up to 15,000 polygons.

A cubemap with a size of 32×32 thereby allows for more than 6,000 possible ray directions. With the simulation running for all 10 frequency bands, this results in over 60k *convolutions* per simulation step, refer to Section 8.2.2. If a direction's acoustic energy is below a certain threshold, it does not contribute to the final auralization, and therefore, the real number of convolutions is vastly reduced. The acoustic quality of smaller cubemaps is comparable with higher resolutions, even though, some details are only audible using larger cubemap sizes, especially diffraction effects. The decomposition of sound data and publicly available HRIR filters was performed using windowed sinc filters with a length of 512 samples (Gardner and Martin, 2000). As expected, the performance decreases with the size of the scenario, as more triangles have to be checked for intersection. An update frequency of at least 10Hz is reached easily for smaller scenarios, but fails for the large hall. A better acceleration structure with a non-uniform subdivision, such as kD trees, might solve this problem.

Figure 66 displays three visualizations of sound wave propagation effects from the ray acoustics system. Here Figure 66a and Figure 66b display direct, 1^{st} and 2^{nd} order reflections with different material settings applied, whereas Figure 66c shows diffraction effects around several wooden pillars. The simulation results are visualized in the unfolded cubemaps and display the frequency range of 22Hz - 320Hz (Figure 66c). The red/brown shifting in color in all cubemaps denotes a stronger transmission/diffraction in the lower frequency end.

Original Sound.

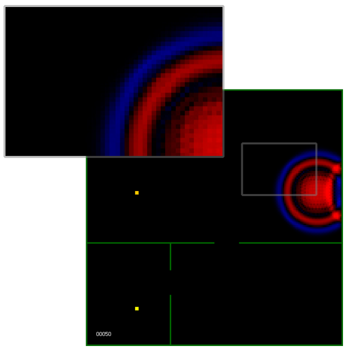
Lowpass Filtered.

CC Simulation.

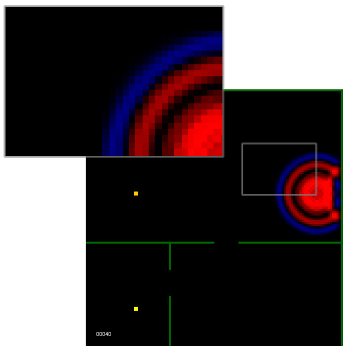
BCC Simulation.

8.5 ANALYSIS AND DISCUSSION OF THE RESULTS

The goal of this research was not to develop a new and more efficient library for 3D sound rendering and simulation, but to discuss these simulations in the context of the special requirements for an auralization of 3D virtual auditory environments. As these environments entirely rely on sound and acoustics to convey abstract data and information, the techniques employed have to fulfill certain requirements: This is on one hand a more accurate and efficient 3D sound rendering and simulation of room acoustics, but also a non-realistic design of the auditory environment and an exaggeration of certain sound propagation effects. As currently available sound rendering APIs only emulate real sound wave propagation to a certain degree, this chapter examined the possibilities for a more efficient and especially for a higher quality 3D sound rendering and room acoustics simulation. Candidates were found in the realm of 3D computer graphics through the exploitation of programmable graphics hardware. The results from Section 8.2, Section 8.3 and Section 8.4 clearly show the advantages of a graphics-based implementation. Besides an improved computational efficiency, also the quality of the examples – see here also Section 9.7 – are very promising and strongly encourage a further research in this direction. Using the results and example implementations in this chapter, it was shown that even the highest demands of 3D virtual auditory environments regarding sound rendering and simulation can be fulfilled.



(a) Rectilinear Grid Frame 50.



(b) BCC Grid Frame 40.

Figure 67: 3D Waveguide Meshes – Wavefront at $t = 50$.

The sound examples on the left allow to listen to several results of a simulation using the Cartesian and the BCC lattice. The examples provide the original sound and a low-pass filtered version, as well as the convolved results of both lattices. The implementation of the 3D waveguide technique in Section 8.3 was realized using two different mesh topologies. Figure 67 and Figure 68 visually compare the differences of both lattices using the example from Figure 61. The visualization of the BCC lattice is not aligned with an axis of propagation, but displayed as a rectilinear grid and sliced along the z axis. The visualizations clearly show a very similar wave propagation at the beginning, which, however, diverges as the animation continues. This is due to the differences in the mesh topologies, as well as the wave propagation itself. The BCC lattice has a smaller dispersion error and propagates the *pressure* along 4 instead of just 3 axes. This can be very well seen in Figure 67, which shows a much *smoother* wave front for the BCC lattice. The coarser resolution is due to the optimal sampling scheme, which requires only 70% of the sampling points to represent the same information, refer Section 8.3.2. The simulation runs approximately $\sqrt{2}$ faster than the implementation of the rectilinear grid.

8.5.1 *Combining Ray- and Wave-based Techniques*

The discussed ray- and wave-based approaches have their respective advantages and drawbacks and are both only applicable to certain parts of the audible frequency range. A combination of the two techniques would allow each method to perform at its peak efficiency and both simulations to complement each other. Depending on the rooms size and the objects therein, a certain threshold (overlapping frequency part) needs to be defined where both techniques are still applicable. The propagation of lower frequencies is simulated using 3D waveguide meshes, while the middle and higher frequencies are approximated using ray acoustics and the here described ray tracing techniques. Alternatively, both techniques can also be combined in a different way. The waveguide technique would have to be changed towards a boundary element method (BEM) and

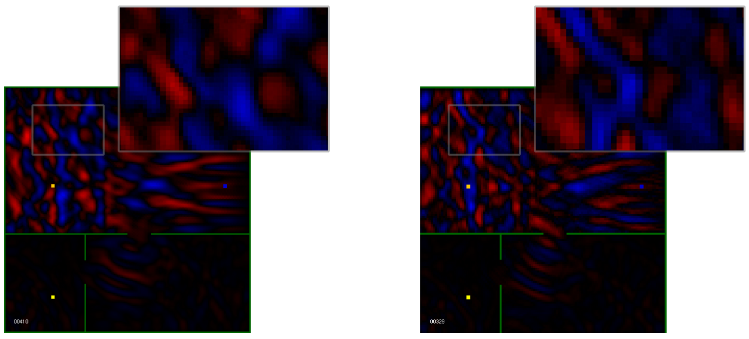

(a) Rectilinear Grid Frame 400. (b) BCC Grid Frame 325.

Figure 68: 3D Waveguide Meshes – Wavefront at $t = 400$.

would now only be applied in the direct vicinity of objects, sound sources and listeners. This would allow a disposal of all waveguide nodes positioned in free space and those which are far away from any object or sound source. At the same time, a decrease of the internodal sampling distance can be performed to achieve a higher sampling frequency for the remaining waveguide meshes. Ray tracing techniques can be used to connect the individual meshes and to transmit and propagate the acoustic energy over larger distances. Ideal for a realization would be an SLI[2]-based graphics system, in which each graphics board is assigned one of the simulation techniques and later both results are combined into a single impulse response. In order to be employed for a continuous auralization, the update frequencies of both techniques have to be aligned.

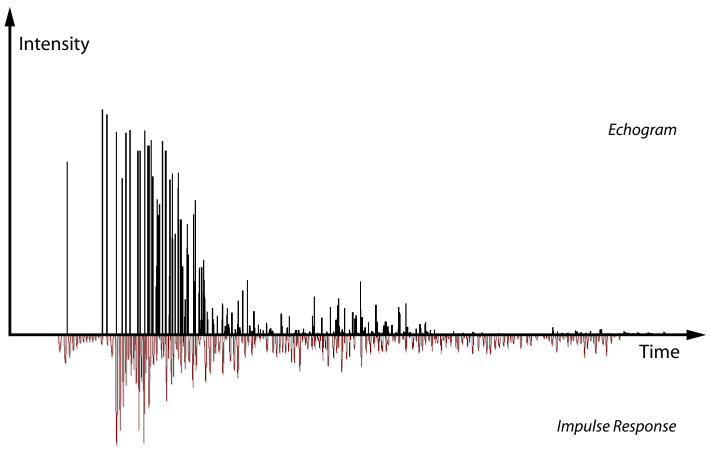


Figure 69: Comparison of Ray- and Wave-based Acoustics Simulations.

Figure 69 shows a direct comparison of both implementations. It displays an echogram (top) of the ray acoustics system and a measured room impulse response (bottom) using 3D waveguide meshes. The room depicted is the same as displayed in Figure 67, and was initialized in both simulations with the same parameters. Even though both techniques are designed for different frequency ranges, the major features of the frequency responses are clearly visible in both results. As this additionally proves the viability of both approaches, it also demonstrates the great potential for a combination of both techniques.

2 Scalable Link Interface refer http://developer.nvidia.com