Waveguide-based Room Acoustics through Graphics Hardware

Niklas Röber*, Martin Spindler and Maic Masuch

Department of Simulation and Graphics,
School of Computing Science,
Otto-von-Guericke-University Magdeburg, Germany

Abstract

The modelling of room acoustics remains a difficult and computationally expensive task. Two main techniques exist, one focusing on the real physical - wave-oriented - sound propagation, while the other approximates sound waves using rays and ray-tracing techniques. One of the most popular wave-based technique are waveguide meshes, which simulate the propagation of sound using time domain difference models.

In this paper we discuss a realtime implementation of the waveguide technique using fragment shaders and computer graphics hardware. Graphics hardware is well suited for parallel implementations and thus able to speed-up the computation. Additionally, we discuss optimal sampling on hexagonal grids to further increase the computational efficiency. We compare both implementations and discuss the results achieved.

1 Introduction

The precise and efficient simulation of room acoustics is a difficult and complex task. Yet, it has applications in many areas, including architectural design, sound and music production and even audio-based computer games (Röber and Masuch 2005). Two main approaches exist: the wave-based and the ray-oriented technique. Raytracing, or geometrical acoustics, has received the most attention in realtime acoustic modelling. It omits the wavelength and approximates all sound waves as rays, and is therefore only applicable to frequencies whose wavelength are much shorter than the dimensions of the enclosure, refer also to (Everest 2001) and (Kuttruff 2000). Adding to this, raytracing has difficulties in modelling various wave phenomena, like diffraction and interference.

Wave-based room acoustics explicitly focuses on these characteristics and simulates the propagation of sound using physical models based on the wave equation (VanDuyne and Smith 1993), (Kahana et al. 1999). One popular method are waveguide meshes that are based on time domain difference models. They have been applied to simulate musical instruments (VanDuyne and Smith 1993), as well as to model the vibrations of air in room acoustics (Savioja et al. 1995). Depending on the mesh's resolution and the internodal sampling distance, the computation can be rather expensive. However, due to advances in computing power, realtime wave-based room acoustics is for smaller meshes within reach.

Driven by the games industry, the efficiency of graphics hardware has evolved by several magnitudes and nowadays even outperforms the CPU in computational capacity. Therefore it is not only used in games, but has also been applied to a variety of non-graphics calculations, including image processing, numerical analysis and various simulations (GPGPU Community 2005). More recently, the GPU was also employed as DSP for sound signal processing (Whalen 2005), (Gallo and Tsingos 2004) and to solve basic geometric room acoustics (Jedrzejewski 2004). The GPU is well suited for the computation of parallel problems, and should perform favorable in solving waveguide meshes.

In this work we present a GPU-based implementation of waveguide meshes, in which the entire computation is carried out within a single fragment shader. This research was motivated by the enormous processing power the GPU has to offer, as well as the possibility to parallelize the implementation. Compared to a standard CPU-based implementation, the performance increases by a factor of *1.5-12*. Additionally, we investigated on optimal sampling and adopted our technique for the BCC lattice. As such hexagonal grids exhibit a more ideal sampling, less sampling points and therefore less computations are required. This further increases the efficiency, and the performance advantage is now *2-15* times, compared to a regular CPU implementation.

The paper is organized as follows: After this introduc-

* {niklas|spindler|masuch}@isg.cs.uni-magdeburg.de

tion, in Section 2 we briefly review the waveguide technique and focus thereafter on our GPU-based implementation for rectilinear meshes. Here we provide details as well as sample code of the fragment shader. In the following Section 3 we extend this implementation to the BCC lattice and discuss the anticipated advantages. In Section 4 we present and discuss our results, while Section 5 summarizes the work and states possible directions for future improvements.

2 Waveguide Room Acoustics

Digital waveguide meshes represent a group of time-domain finite difference models. The 1-dimensional waveguide technique is a numerical solution to the wave equation and was first applied to simulate string-based music instruments (Smith 1992). The digital waveguide mesh is an extension of the 1D technique and constructed by bi-linear delay lines, that are arranged in a mesh-like structure (VanDuyne and Smith 1993), see also Figure 1. Higher dimensions are built by scattering junctions that are connected to the delay lines and act as spatial and temporal sampling points. These scattering junctions are of equal impedance and two main constraints apply:

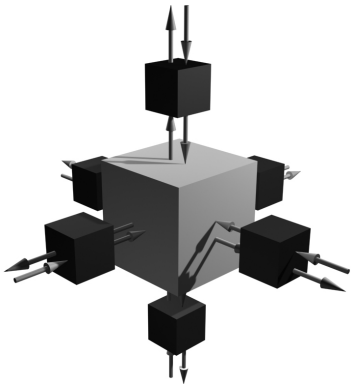


Figure 1: 3D Rectilinear Waveguide Node.

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

For a lossless scattering, the *acoustic pressure* v_J can be computed by summing the incoming wave components v_i^+ (VanDuyne and Smith 1993):

$$v_J = \frac{2\sum_i R_i v_i^+}{\sum_i R_i} \tag{1}$$

$$v_i^- = v_J - v_i^+ \tag{2}$$

Equation 2 expresses the relation between the incoming v_i^+ and the outgoing v_i^- wave components. In an homogenous N-dimensional rectilinear mesh, with each junction having $2N$ neighbors, Equation 1 reduces to the form:

$$v_J = \frac{\sum_i v_i^+}{N} \tag{3}$$

With further optimization and by discretizing time and space, one obtains the difference equations for the nodes of an N-dimensional rectangular mesh, with n being the time steps:

$$v_{J,k} = \frac{\sum_l v_{J,l}(n-1)}{N} - v_{J,k}(n-2) \tag{4}$$

Here k represents the current node and l is associated with the neighboring nodes. To model a boundary behavior, so called 1D termination nodes with only 1 neighbor are employed. The literature differentiates between three main cases: phase-reversing reflections (Equation 5), phase-preserving reflections (Equation 6) and anechoic walls (Equation 7) that absorb all acoustic energy (Savioja et al. 1995):

$$v_J = 0.0 \tag{5}$$

$$v_{J,k}(n) = v_{J,l}(n) \tag{6}$$

$$v_{J,k}(n) = v_{J,l}(n-1) \tag{7}$$

But this approach only provides a rough approximation of the real boundary conditions and more research should be conducted in this area.

The main problems with digital waveguide meshes are a direction dependent dispersion error and the finite mesh resolution to model boundary behavior (VanDuyne and Smith 1993). Several approaches have been discussed to overcome these limitations, and include higher tesselated meshes, different mesh topologies and frequency warping techniques (VanDuyne and Smith 1996), (Savioja and Välimäki 2000), (Beeson and Murphy 2004), (Fontana and Rocchesso 2001).

2.1 Rectilinear Waveguide Meshes

The equations that govern the 3D rectilinear waveguide mesh are based on difference equations derived from the Helmholtz equation by discretizing time and space (Savioja and Lokki 2001):

$$\begin{aligned} p(t+1,x,y,z) \quad = \quad & \\ & \tfrac{1}{3}[p(t,x+1,y,z) + p(t,x-1,y,z) \\ & +p(t,x,y+1,z) + p(t,x,y-1,z) \\ & +p(t,x,y,z+1) + p(t,x,y,z-1)] \\ & -p(t-1,x,y,z), \end{aligned} \tag{8}$$

in which p represents the sound pressure at sampling point (x, y, z) and time step t. The update frequency with unit length 1.0 is:

$$f_{update} = \frac{c\sqrt{N}}{\Delta x} \approx \frac{588.9}{\Delta x}\, Hz, \tag{9}$$

with $N = 3$ and $c = 340\ m/s$ for sound propagation under normal room conditions. This update frequency is also the sampling frequency of the room impulse response, with previous research showing that a typical waveguide mesh gives a valid bandwidth only as far as $f_{update}/4$ (VanDuyne and Smith 1993). This is due to the fact that in a square mesh the frequency response repeats itself after that limit (VanDuyne and Smith 1996), (Savioja and Välimäki 2000). Higher sampling rates require finer tesselated meshes, which further results in an increase in computing complexity.

The speed of wave propagation within the digital waveguide mesh is non-isotropic. This error is known as frequency dispersion and varies between different mesh topologies. The dispersion error for the rectilinear grid ranges from zero along the diagonal directions to its maximum along the coordinate axes. The factor that is used to quantify dispersion is defined as:

$$k_d(\underline{\beta}) = \frac{c'\underline{\beta}}{c}, \tag{10}$$

with c being the speed of the 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 are derived using Von Neumann analysis and can also be found in the literature (Bilbao 2004), (Campos and Howard 2005), (Fontana and Rocchesso 2001). For the 3D rectilinear mesh the dispersion factor ranges on a spherical surface between $0.927 < k_d < 1$ with $k_d = 1$ along the diagonal directions. The signal bandwidth is limited by the underlying lattice that is used to sample the pressure function at predefined positions (Dudgeon and Mersereau 1984). The sampling of a signal in the time domain corresponds to replicating their spectra in the frequency domain. Important is that the spectra do not overlap, as otherwise aliasing artifacts will be introduced. A 3D signal is spherically bandlimited with bandwidth B if its spatial spectrum resides completely within a sphere of radius B. One problem with waveguide meshes is that this spectrum is deformed due to the direction dependent dispersion factor k_d. The spatial bandwidth for the 3D rectilinear mesh is:

$$B_{rect} = \frac{1}{2d}, \tag{11}$$

with $d = 1$ (Campos and Howard 2005). The sampling and the computational efficiency for this lattice are both 1, but more efficient sampling schemes are available.

2.2 Graphic-based Waveguide Modelling

Today's graphics hardware with the GPU as its main processor can be seen as powerful parallel computing machines that efficiently executes small programs - so called shaders. The graphics pipeline paradigm describes two types of shaders: vertex shaders and fragment shaders that are both executed at different stages in the pipeline. Vertex shader allow arbitrary manipulations on a per vertex basis in 3D space, while fragment shader are used to compute the final color for each pixel. Both shaders are freely programmable using high level shading languages such as GLSL and Cg (Rost 2004). As graphics applications typically require the processing of huge amounts of data, graphics hardware has been optimized to support this with a highly parallel design. This makes this hardware also very interesting for parallel computing problems, such as wave propagation using waveguides meshes.

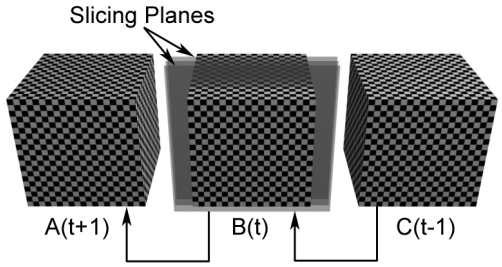


Figure 2: Waveguide Rendering Principle.

Our technique is mainly based on 3D textures, fragment shaders, and the new OpenGL framebuffer objects (fbo) extension. For the simulation we use two channels of two 3-component 32-bit floating point textures, which hold the mesh data of the time steps $t-1$ and t. In an alternating fashion the channels are directly rendered into an fbo-buffer, having one texture attached to it as the primary render target (ping-pong buffer). An additional channel holds information about the scene geometry and the material specifications to model boundary conditions. In the implementation, these channels are combined into one RGB texture with the waveguide data stored in the red and blue components and the geometry and boundary coefficients in the green channel. Each time frame, the fragment shader computes the difference equations for each node in the waveguide mesh and stores the results into the next buffer. Figure 2 visualizes the principle.

The three textures A, B and C contain the waveguide node data, and texture B is *sampled* by texture aligned slicing planes. These planes have the same resolution as the texture, and for each voxel, the fragment shader in Listing 1 is evaluated. The result is saved in the current

time texture (A), which is then used in the next time step. A problem with the relatively new framebuffer objects is that 3D textures are not supported yet, although an extension is already defined. In our current implementation, the computation is done in 2D by slicing the texture along its shortest extent. This results in copying large amounts of texture data, which slows down the current implementation, see also the discussion in Section 4.

```
 1 uniform  float  layer;
 2 uniform  vec3  stepX, stepY, stepZ;
 3 uniform  sampler3D  tex;
 4 vec3  pos = vec3(gl_TexCoord[0].xy, layer);
 5
 6 vec4  center = texture3D(tex, pos);
 7 vec4  left   = texture3D(tex, pos - stepX);
 8 vec4  right  = texture3D(tex, pos + stepX);
 9 vec4  up     = texture3D(tex, pos + stepY);
10 vec4  down   = texture3D(tex, pos - stepY);
11 vec4  front  = texture3D(tex, pos + stepZ);
12 vec4  back   = texture3D(tex, pos - stepZ);
13
14 float  ampl  = left.r + right.r + up.r + down.r;
15        ampl += front.r + back.r;
16        ampl  = ampl * 0.3333 - center.b;
17 gl_FragColor = vec4(ampl, center.g, center.r, 1.0);
```

Lst. 1. Waveguide Fragment Shader (Cartesian Lattice).

Listing 1 shows the fragment shader for the Cartesian lattice without boundary conditions. Lines 1 to 4 provide the sampling width, using which the waveguide nodes are accessed between lines 6 and 12. The difference equations and the final amplification is computed in lines 14 till 16 and returned as the fragments color in line 17.

Although, wave-based room acoustics can be computed much faster using this technique, one limitation is the available graphics memory. Constantly increasing, current graphics hardware features either 256 or 512 MB of RAM. This provides enough memory for $15,000,000$ respectively $30,000,000$ waveguide nodes.

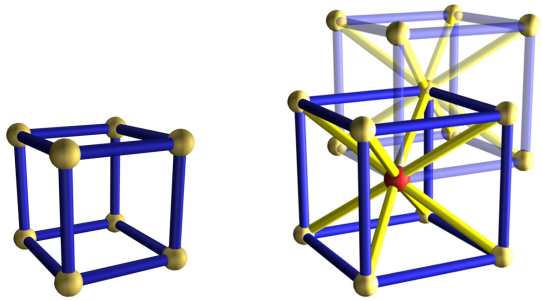
2.3 Auralization

Auralization is defined as technique that uses computer-based mathematical models of an acoustic environment as well as 3D sound processing methods to make (anechoic) sound audible as of a source in the modelled space. Here often the rooms responses to a short impulse is measured and later used to derive room modes and other acoustic properties. These responses are also used to convolve anechoic sound data, to appear as if played in this room.

In our auralization experiments, we have employed the waveguide technique to measure room impulse responses as well as to record the rooms influences directly by exciting the mesh using low-pass filtered sounds. A comparison of the results and implementation of the two lattices are discussed and shown in Section 4.

3 Optimal Sampling

The Cubic Cartesian lattice is employed in many applications and used as sampling scheme for the storage and measurement of data. But in terms of sampling efficiency, the rectilinear grid is not the most optimal. A denser packing of the spectra in the frequency domain, results in an increase of the sampling distance in the spatial domain. Under the assumption of an isotropic spherically bandlimited signal, hexagonal lattices provides a much better packing density. In mathematics, this problem refers to the closest packing of spheres (Sloane 1998). The Body Centered Cubic grid (BCC) represents such a hexagonal lattice, Figure 3(b). The use of hexagonal grids is common in nature, as for example bees build their honeycombs using hexagonal cells, and as a result achieve a minimum expenditure of wax. Also many crystals exhibit a hexagonal spacing to achieve a minimal energy state. A direct comparison with the Cartesian grid in Figure 3(a), shows that the BCC has 8 nearest neighbors, and is constructed as a cubic grid with and additional sampling point in the center. The sampling distance is $\sqrt{1.5}$, which in 3D results in roughly only 70 % of the sampling points needed to sample the same amount of data (Conway and Sloane 1976).



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

Figure 3: Rectilinear and Body Centered Cubic Lattice.

One advantage of the BCC grid is that it can be decomposed into two cubic grids, compare with Figure 3(b). Generally, the N dimensional hexagonal lattice can be constructed by two $N-1$ rectilinear grids which are offset by $\sqrt{2}$ in all N dimensions. This allows an easy indexing of the data, which is also important a the later implementation. The BCC is already known and used in computer

science for image processing and scientific visualization (Theußl et al. 2001), (Röber 2002), but has its main roots in chemistry and crystallography (Jackson 1991), (Wells 1984).

3.1 BCC Waveguide Mesh

The BCC lattice exhibits several advantages for the simulation of room acoustics through digital waveguides meshes. Due to a more optimal sampling, this grid only requires $70\ \%$ of the sampling points compared to a rectilinear mesh, which reduces the computational load by $\sqrt{2}$. But the most noticeable difference is that the BCC grid has 4 principal axes with 8 neighbors and therefore 4 delay units per node. The mesh itself is cubic, has a uniform spatial orientation and can be indexed easily, see also Section 3.3. The internodal sampling distance is increased by $\sqrt{1.5}$, which also influences the speed of the wave propagation.

The difference equations for the BCC lattice can be derived from the finite difference model discussed in Equation 4. With now 4 principal axes, they are modified to:

$$\begin{aligned} p(t+1,w,x,y,z) \quad = \\ \tfrac{1}{4}[&p(t,w+1,x,y,z)+p(t,w-1,x,y,z) \\ &+p(t,w,x+1,y,z)+p(t,w,x-1,y,z) \\ &+p(t,w,x,y+1,z)+p(t,w,x,y-1,z) \\ &+p(t,w,x,y,z+1)+p(t,w,x,y,z-1)] \\ &-p(t-1,w,x,y,z), \end{aligned} \tag{12}$$

in which p is again 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 $\sqrt{1.5}$, which changes the update frequency f_{update} to:

$$f_{update} = \frac{c\sqrt{2}}{\Delta x} \approx \frac{480.8}{\Delta x}\ Hz. \tag{13}$$

Due to these differences in sampling distance, the BCC lattice propagates the waves *faster*, which has to be considered for impulse response measurements. In order to directly compare the signals of both lattices, the frequency f_{BCC} which is used to excite the BCC mesh needs to be adjusted by: $f_{BCC} = \frac{f_{CC}}{\sqrt{1.5}}$, and later also the measured response by the inverse factor: $f_{CC} = f_{BCC}\sqrt{1.5}$.

3.2 Mathematical Analysis

The BCC unit cell, compare with Figure 3(b), contains two nodes with 8 delay units. With an internodal sampling distance of $d = \sqrt{1.5}$ the mesh's edge length Δ_{BCC} and density μ_{BCC} are defined as:

$$\Delta_{BCC} = \frac{2d}{\sqrt{3}} = \sqrt{2}, \tag{14}$$

$$\mu_{BCC} = \frac{3\sqrt{3}}{4d^3} = \frac{1}{\sqrt{2}}. \tag{15}$$

The frequency dispersion factor k_d, as can be found in the literature, has a range of $0.953 < k_d < 1$ for a spherical surface. At its maximum the error is only $4.7\ \%$, compared to the 3D rectilinear grid with $7.3\ \%$ for $\beta_n = \pi/2$ (Campos and Howard 2005). The spatial bandwidth can be determined by decomposing the BCC lattice into two rectilinear grids of spacing d. The grid bandwidth of the BCC lattice is equal to the FCC (face centered cubic) and with $d = \sqrt{1.5}$ defined as:

$$B_{BCC} = \sqrt{\frac{3}{2}}\frac{1}{2d}. \tag{16}$$

The sampling efficiency of the BCC lattice compared to the 3D rectilinear grid is:

$$\frac{\mu_{BCC}}{\mu_{CC}} = \frac{1}{\sqrt{2}} \approx 0.707. \tag{17}$$

Although the computational efficiency of the BCC lattice is for each node slightly higher when compared to the rectilinear Cartesian grid, it is still much more efficient due to the more optimal sampling scheme.

3.3 GPU Implementation

The possibility to decompose the BCC lattice into two interlaced cubic grids, refer Figure 3(b), makes the implementation straightforward and very similar to the rectilinear grid. The changes are one additional 3D texture and an adapted fragment shader. The BCC waveguide mesh is decomposed into two rectilinear 3D textures, which are both loaded into the texture memory. As for efficiency, we were able to place both grids with 2 time frames into just one single texture. The base grid resides in the channels R and G, while the offset grid is placed at B and A. This allows us to compute two nodes in one step. Although the complexity of the shader is a bit more expensive, but as less sampling points and a lower update frequency are required, the overall computation is vastly reduced.

4 Results and Discussion

We have carried out a set of experiments using both meshes to verify the proposed techniques and to demonstrate their advantages. The experiments have been performed using an AMD64 4000+ dual-core processor with

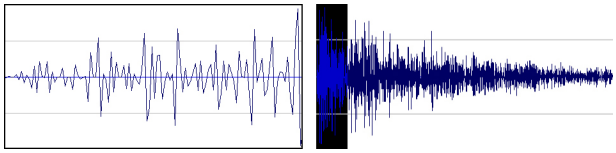
Mesh Size	2D-CC CPU	2D-CC GPU
128×128	1297.0 fps	$5,917.0$ fps
256×256	180.3 fps	$2,303.0$ fps
512×512	13.6 fps	850.1 fps
1024×1024	3.4 fps	234.3 fps

Table 1: Benchmark Results 2D - CPU vs. GPU.

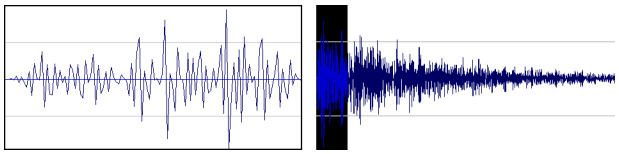
1 GB of main memory. The graphics accelerator was an *n*vidia GeForce7900GT with a PCIe interface. Table 1 displays the rendering speed for the 2D rectilinear implementation only.

As can be clearly seen from these results, the GPU implementation becomes even more efficient for larger meshes, as the speed advantage increases from a factor of 4.5 to 69. The main reason is within the chip's design that allows a highly parallel implementation, as well as a much higher computing capacity. Additionally, the memory access is much faster on graphics hardware than on the CPU. Here, we have to note for fairness that the CPU implementation is not optimized and simply implements the same code as the GPU version. Although modern CPUs feature several megabytes of fast accessible cache, this becomes more ineffective with an increasing size of data sets. Table 2 shows the benchmark results for the 3D and the 3D BCC lattice in comparison to a standard (not optimized) CPU implementation. The reason why the results for 3D and 3D BCC are not as impressive as for the 2D GPU implementation is because of a missing 3D framebuffer extension that allows a direct access of 3D textures within frame buffer objects. This extension is defined, but not implemented yet by the graphics hardware vendors. The current implementation uses the more expensive $glCopyTexSubImage3D$ function, which massively decelerates the computation due to a poor implementation in the driver. The last two columns of Table 2 show an anticipation of the speed with just these functions disabled. Clearly can be seen from these results that the proposed technique is able to improve the efficiency by a vast number. Furthermore, the advantages of the BCC lattice in computational efficiency are clearly visible here.

Figure 4 shows two room impulse responses that where measured using both techniques, in which a short sine pulse was used to excite the meshes. The mesh size is $82 \times 82 \times 12$ for the rectilinear, and $58 \times 58 \times 17$ for the BCC lattice. On the left side of Figure 4 is a magnification of the first parts of the impulse response, showing the early reflections. Although, both responses share many similarities, especially in the first samples, it can also clearly be seen that the signals deviate with increasing time. This is mostly due to the fact that the sound energy is distributed along 4 instead of 3 axes. The next two sections discuss two experiments in more detail. More results, additional videos and sound samples can be found online on our website[1].



(a) 3D Rectilinear Grid.



(b) 3D BCC Grid.

Figure 4: Room Impulse Response Measurement.

4.1 Example 1: Three Rooms

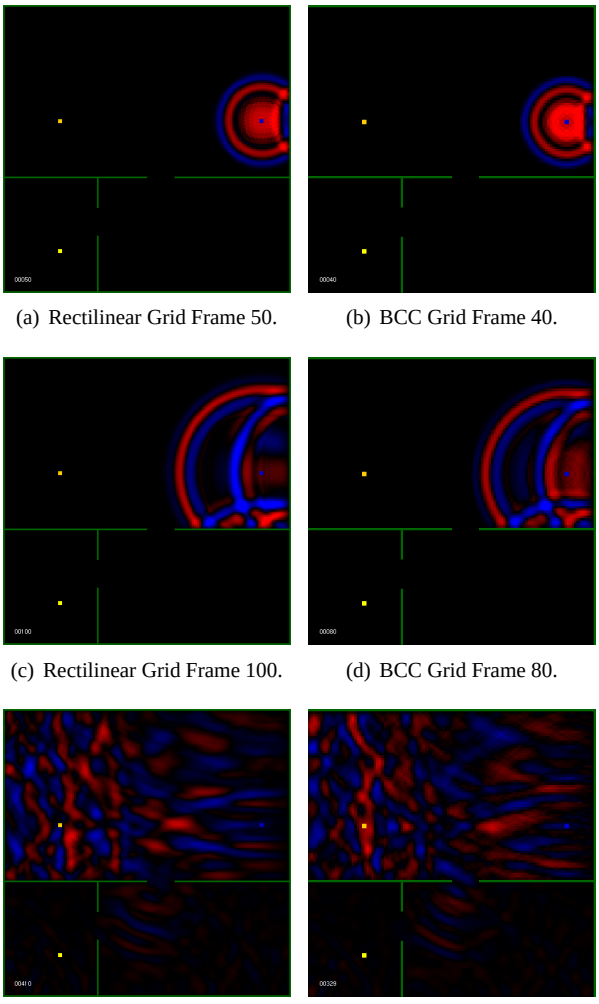
The first experiment shows a setting of three rooms with complete reflecting walls and ceiling. It was used to compare the quality, as well as the efficiency of both implementations. The layout of the room's design can be seen in Figure 5, which displays one sound source (blue), two microphones (yellow) and the walls (green). The data set has a size of $82 \times 82 \times 24$ for the rectilinear Cartesian and a size of $58 \times 58 \times 34$ for the BCC lattice. The sound source, the walls and the microphone positions are adjusted accordingly to fit the BCC's dimensions. The rendering speed is for the rectilinear grid on average $13.4\ fps$ and for the BCC $53.8\ fps$. Both meshes were excited using a single sine wave, whose frequency was adjusted for the BCC lattice according to Section 3.1.

Figures 5(a) to 5(f) display frames of an animation and show the visual results for the rectilinear, respectively the BCC grid. The visualization of the BCC grid is not sliced along the axes of propagation, instead it is displayed as the Cartesian, along the z axis. Blue denotes a negative pressure, while red visualizes regions with positive sound pressure. It can be clearly seen from the visualizations that the wave propagation in both meshes is very similar. Especially the first frames are very much alike. The divergence towards the end is due to the differences in the topology of the meshes. The dispersion error is different and less for the BCC grid, and additionally, the signal is propagated along 4 axes instead of just 3. As we make use

[1] http://games.cs.uni-magdeburg.de/acoustics.html

Mesh Size	3D-CC CPU	3D-CC GPU (cp)	3D-BCC GPU (cp)	3D-CC GPU (ncp)	3D-BCC GPU (ncp)
$41 \times 41 \times 24$	769.2 fps	42.2 fps	153.1 fps	1,164.0 fps	1,500.0 fps
$82 \times 82 \times 24$	144.9 fps	13.4 fps	53.8 fps	560.2 fps	690.5 fps
$164 \times 164 \times 24$	28.7 fps	3.7 fps	17.1 fps	238.4 fps	292.9 fps
$328 \times 328 \times 24$	5.3 fps	1.1 fps	4.5 fps	65.2 fps	79.6 fps

Table 2: Benchmark Results (cp = with glCopyTexSubImage3D, ncp = no glCopyTexSubImage3D).



(a) Rectilinear Grid Frame 50. (b) BCC Grid Frame 40.

(c) Rectilinear Grid Frame 100. (d) BCC Grid Frame 80.

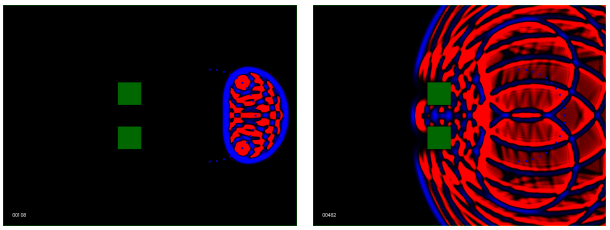
(e) Rectilinear Grid Frame 400. (f) BCC Grid Frame 325.

Figure 5: Three Rooms - Cartesian vs. BCC.

of the superior sampling efficiency of the BCC grid, the internodal distance, and hence the mesh size, is also different according to Section 3. We have chosen the values in a way that only small rounding errors are introduced, but nevertheless, they will sum up. The characteristic features of the wave propagation are present and the computation is faster by a factor of 4.

4.2 Example 2: Wavefield Synthesis

Wavefield synthesis aims at the generation of large wavefronts by combining several small ones (Boone 2001). If arranged in the form of a circle, they can be used to synthesize large parallel wavefronts to study wave-based phenomena, such as interference, occlusion and diffraction effects. Figure 6(a) shows a large wavefront as a result of 21 sound sources that excite the mesh time-controlled using a sine wave with a frequency depending on their separation. Figure 6(b) displays a later time frame and shows several wave-phenomena after the larger waves were broken and diffracted by and around two columns. The experiments were conducted in 2D with a mesh size of 640×480. The outer walls were designed to be anechoic, while the inner obstacles reflect the sound waves in a phase-reversing manner.



(a) Frame 108. (b) Frame 482.

Figure 6: Wavefield Synthesis using 21 Sound Sources.

5 Conclusions and Future Work

We have presented a novel and fast technique that computes wave propagation using waveguide meshes on computer graphics hardware. Additionally, we have explored the applicability of hexagonal lattices and demonstrated that this further decreases the computational complexity. The results are very promising, but also evoke a discussion on which lattice - in the acoustic sense - performs better. The answer is not clear and more research should be done in this area.

Although, this research has clearly shown that the BCC lattice is able to cut down the rendering time by a large

factor, still the increasing complexity for shorter wavelength makes the waveguide technique only applicable for lower frequencies. But an authentic auralization needs all parts of the audible spectrum. A combination of this waveguide method with other graphics-related techniques, such as raytracing, would make sense and offer a possibility for realtime auralization of the entire spectrum. Additionally, we would like to explore the potential of other applications to this technique, such as an auditory computer game that relies on acoustic perceptions based on echolocation.

6 Acknowledgement

The authors would like to thank Torsten Möller for helpful discussions on optimal sampling and hexagonal lattices, as well as Mathias Otto and Hanno Hugenberg for running the benchmarks.

References

Beeson, M. J. and D. T. Murphy (2004). Roomweaver: A digital Waveguide Mesh based Room Acoustics Research Tool. In *7th Int. Conference on Digital Audio Effects (DAFX)*, Naples, Italy, pp. 268–273.

Bilbao, S. (2004). *Wave and Scattering Methods for Numerical Simulation* . John Wiley & Sons.

Boone, M. M. (2001). Acoustic Rendering with Wavefield Synthesis. In *ACM SIGGRAPH Campfire: Acoustic Rendering for Virtual Environments*.

Campos, G. R. and D. M. Howard (2005). On the Computational Efficiency of Different Waveguide Mesh Topologies for Room Acoustic Simulation. *IEEE Transactions on Speech and Audio Processing 13*(5), 1063–1072.

Conway, J. H. and N. J. A. Sloane (1976). *Sphere Packings, Lattices and Groups* (2nd ed.). Springer.

Dudgeon, D. E. and R. M. Mersereau (1984). *Multidimensional Digital Signal Processing*. Prentice-Hall Signal Processing Series. Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 07632: Prentice-Hall.

Everest, F. A. (2001). *The Master Handbook of Acoustics*, Volume 4th edition. McGraw-Hill/TAB Electronics.

Fontana, F. and D. Rocchesso (2001, Februar). Signal-Theoretic Characterization of Waveguide Mesh Geometries for Models of Two-Dimensional Wave Propagation in Elastic Media. *IEEE Transactions on Speech and Audio Processing 9*(2), 152–161.

Gallo, E. and N. Tsingos (2004). Efficient 3D Audio Processing with the GPU. In *GP2, ACM Workshop on General Purpose Computing on Graphics Processors*.

GPGPU Community (2005). General-Purpose Computation Using Graphics Hardware. `http://www.gpgpu.org`.

Jackson, A. G. (1991). *Handbook of Crystallography*. Springer.

Jedrzejewski, M. (2004). Computation of Room Acoustics Using Programmable Video Hardware. In *International Conference on Computer Vision and Graphics*.

Kahana, Y., P. Nelson, M. Petyt, and S. Choi (1999). Numerical Modeling of the Transfer Functions of a Dummy-Head and of the external Ear. In *Audio Engineering Society, 16th International Conference*.

Kuttruff, H. (2000). *Room Acoustics* (4th ed.). Spon Press, London.

Röber, N. (2002). Visualization of Fuel Cell Simulations. Master's thesis, University of Magdeburg, Germany.

Röber, N. and M. Masuch (2005). Leaving the Screen: New Perspectives in Audio-only Gaming. In *Proceedings of 11th ICAD*. Limerick, Ireland.

Rost, R. J. (2004). *OpenGL Shading Language*. Addison-Wesley.

Savioja, L., J. Backman, A. Jrvinen, and T. Takala (1995). Waveguide Mesh Method for Low-Frequency Simulation of Room Acoustics. In *15th Int. Congress on Acoustics (ICA'95)*, Trondheim, Norway, pp. 637–640.

Savioja, L. and T. Lokki (2001). Digital Waveguide Mesh for Room Acoustic Modelling. In *ACM SIGGRAPH Campfire: Acoustic Rendering for Virtual Environments*, Utah, USA.

Savioja, L. and V. Välimäki (2000). Reducing the dispersion Error in the digital Waveguide Mesh using Interpolation and Frequency Warping Techniques. *IEEE Transactions on Speech and Audio Processing 8*, 184–194.

Sloane, N. J. A. (1998). The sphere packing problem. In *Proceedings of the International Congress of Mathematicians*, Berlin, pp. 387–396. Doc.Math.J.DMV Extra Volume ICM III.

Smith, J. O. (1992). Physical modelling using digital Waveguides. *Computer Music Journal 16*(4), 75–87.

Theußl, T., T. Möller, and E. Gröller (2001, October). Optimal Regular Volume Sampling. In *Proceedings IEEE Visualization*, pp. 91–98.

VanDuyne, S. and J. Smith (1993). Physical Modelling of the 2-D digital Waveguide Mesh. In *Int. Computer Music Conference*, Tokyo, Japan, pp. 40–47.

VanDuyne, S. and J. Smith (1996). The 3D Tetrahedral Digital Waveguide Mesh with Musical Applications. In *Int. Computer Music Conference*, Hong Kong, pp. 9–16.

Wells, A. F. (1984). *Structural Inorganic Chemistry* (5th ed.). Oxford University Press.

Whalen, S. (2005). Audio and the Graphics Processing Unit. Technical report. `http://www.node99.org/projects/gpuaudio/`.