Monte Carlo Path Tracing for Atmospheric Sound Propagation

Submitted in partial fulfillment of the requirements for the degree of

Master of Science (M.Sc.)

by

Fabian Krause
4331777

First Examiner
Prof. Dr.-Ing. Matthias Teschner

Second Examiner
Prof. Dr. Thomas Brox

Chair of Computer Graphics
Department of Computer Science
Faculty of Engineering
University of Freiburg

Fabian Krause: *Monte Carlo Path Tracing for Atmospheric Sound Propagation*.

ABSTRACT

Outdoor sound propagation is affected by atmospheric conditions, including temperature, humidity, pressure, and wind. Accurately predicting atmospheric sound propagation has important applications in both civilian and military contexts, such as minimizing noise pollution or enhancing mission planning. We implement a Monte Carlo path tracer for predicting ground noise levels in large outdoor scenes with arbitrary atmospheric conditions by approximating atmospheric sound propagation as a series of linear ray segments through a layered medium. We investigate the effectiveness of several variance reduction methods and validate our method by analyzing the results of various simulations for their physical accuracy, as well as comparing them to those obtained using established methods. Combining GPU-acceleration and variance reduction methods, we achieve near real-time capabilities, enabling the future integration of the method into interactive tools. However, we also demonstrate that performance benefits due to the linear ray approximation come at the cost of reduced accuracy.

ACKNOWLEDGMENTS

I would like to thank Prof. Dr.-Ing. Matthias Teschner for agreeing to examine this thesis, and for providing valuable feedback for the overall structure and direction of this thesis. I would also like to thank Prof. Dr. Thomas Brox for kindly agreeing to be the second examiner.

I would also like to thank Airbus Defence & Space for providing me with the opportunity and resources to write this thesis. I especially want to thank Maria Gruber for her continual support in all corporate matters, and Fabian Miehe for proposing the thesis topic of accelerating atmospheric sound propagation by use of graphics processing units.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

1 INTRODUCTION

PROBLEM STATEMENT Outdoor sound propagation is strongly influenced by atmospheric conditions such as temperature, humidity, pressure and wind. A method to accurately predict atmospheric sound propagation has numerous applications in both civilian and military contexts.

For example, it may be used for the planning of airport sites, making sure that noise pollution in residential areas is minimized in all weather conditions. Furthermore, it may assist flight planning, using routes that minimize noise impact on civilians and wildlife, based on current weather conditions.

The way sound refracts in the atmosphere can influence the outcome of a battle. There are numerous examples of this in the US Civil War [1]. Although these instances are unlikely today due to modern communication methods, the ability to accurately predict noise may prove useful in mission planning for aircraft. Routes may be chosen that reduce the likelihood of detection for stealth operations.

In the military context, time may be of essence. It is therefore important to not only find an accurate, but also quick method for predicting noise levels due to atmospheric conditions.

GOAL The goal of this thesis is to develop an efficient method for the prediction of ground noise levels generated by a sound source in large outdoor scenes, taking into account atmospheric conditions. The method should be efficient enough to be used interactively. To this end, a Monte Carlo path tracer is implemented, and both variance reduction strategies and hardware acceleration are used to facilitate interactive use. Since available hardware is optimized for linear rays, atmospheric propagation is modeled using linear ray segments.

RELATED WORK Several approaches to the simulation of atmospheric sound propagation exist. These can generally be separated into two categories: Wave-based numerical solutions and geometrical solutions.

Wave-based numerical approaches divide the environment into finite elements and solve the wave-equation using the Finite Element Method (FEM) [2], the Boundary Element Method (BEM) [3] or the Finite Difference Time Domain method (FDTD) [4]. These methods generally provide the most accurate results, but also take considerable time, making them infeasible for interactive use.

In geometrical acoustics, wavefronts are modeled by rays. Although stochastic rendering techniques are widespread in room acoustics [5] [6] [7] [8] [9], for atmospheric sound propagation in outdoor scenes with non-homogeneous moving media, the established methods deterministically calculate rays between a source and microphone positions (also known as Eigenrays), which requires solving differential

equations [10] [11] [12], making these infeasible for interactive use. In [13], an analytical solution is derived and solved using a root-finding algorithm to find Eigenrays. Whereas [10] and [11] do not model any reflections, [12] and [13] only model specular reflections, and only one specular reflection per microphone using the image-source method. Also, the terrain is assumed to be completely flat, making these methods inapplicable for most scenarios.

In [6], photon mapping is applied to room acoustics. This is called sonel mapping, or phonon mapping [14]. Our approach is similar, with the key difference that there is no second step where rays are traced from a listener position. In our case, we are interested in the global noise levels.

In [9], a bidirectional path tracing algorithm is developed for interactive sound propagation. The presence of an atmosphere makes connecting two paths non-trivial, hence, we implement a forwards path tracing algorithm.

This thesis is greatly inspired by a Master's thesis dealing with refraction due to differing sound speeds in underwater sound propagation [15], where sound ray refraction has also been modeled using linear ray segments, albeit in a two-dimensional setting. The author is not aware of any published works where Monte Carlo path tracing is applied to atmospheric sound propagation with non-trivial terrain.

OUTLINE In Chapter 2, we establish the theoretical framework of our method. We start with the basic theory of physically based rendering in Section 2.1 before moving onto the Monte Carlo method in Section 2.2. We finish the foundations with a section on acoustics in Section 2.3.

In Chapter 3, we discuss how the theoretical framework is implemented. Starting with the rendering engine, NoiseTracer, in Section 3.1, then covering performance improvements in Section 3.2 and finally discussing testing and validation in Section 3.3.

The quality of the method is discussed in Chapter 4, where we begin with establishing a set of three test scenes of various terrain types in Section 4.1, then continue with an analysis of several simulations with respect to their physical accuracy in Section 4.2. We proceed with discussing the effectiveness of the variance reduction strategies in the different scenes in Section 4.3, before considering the performance of NoiseTracer in Section 4.4.

Finally, in Chapter 5, we give a short summary of the results and discuss future work.

2 BACKGROUND

We start this chapter with a discussion of the basic theory of physically based rendering (PBR) in Section 2.1, which forms the basis of the method introduced here.

We then move on to Monte Carlo integration in Section 2.2 to provide a solution for the problem of light transport, as well as to discuss various methods for improving the Monte Carlo estimation.

Finally, in Section 2.3, we discuss the fundamentals of sound and the geometrical acoustics model. We then introduce the principles of atmospheric sound propagation, as well as their integration into the PBR framework.

2.1 PHYSICALLY BASED RENDERING

2.1.1 Measurement Equation

The goal of rendering is to get measurements for sensors, usually light measurements for pixels on a view plane, in a three-dimensional scene filled with objects and sources. These measurements can be calculated using the measurement equation [16],

$$I = \int_{M \times S^2} W_e(\mathbf{x}, \omega) L_i(\mathbf{x}, \omega) |\cos\theta| \, \mathrm{d}A(\mathbf{x}) \, \mathrm{d}\omega \,. \tag{2.1}$$

Let us unpack this equation. W_e is the responsivity function, which describes how much of ingoing light is measured, based on a point $\mathbf{x}$ and the ingoing direction ω. The factor L_i is the incoming radiance. We will explain radiance in Section 2.1.5, for now it is enough to know that this describes how much light arrives. This is again dependent on the point $\mathbf{x}$ on the sensor and on the direction ω. The $\cos\theta$ factor is due to Lambert's cosine law, where θ is the angle between surface normal and ω. Light may reach any surface from any direction, so we integrate over all surfaces M using the area measure A, and over the entire sphere S^2. For a specific sensor defined by some surface m in a scene, we define the responsivity function to be 1 if $\mathbf{x}$ is on m, and 0 otherwise.

The interesting part of the above equation is L_i: How do we know how much light reaches point $\mathbf{x}$ from a direction ω? Due to energy conservation, we know that the incident light L_i is equal to the outgoing light L_o at some other point $\mathbf{x}'$ that emits or scatters light towards $\mathbf{x}$. To find point $\mathbf{x}'$, we cast a ray starting from $\mathbf{x}$ in the direction ω. Denoting $-\omega$ as the opposite direction to ω, we have

$$L_i(\mathbf{x}, \omega) = L_o(\mathbf{x}', -\omega).$$

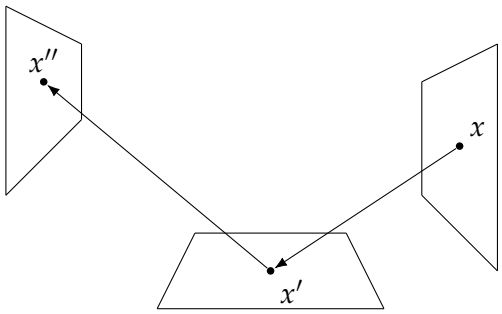


Figure 2.1: Three-point form of the light transport equation.

Any point may emit light, scatter light, or both. We denote the emitted light as L_e and the scattered light as L_s. The emitted light of a surface may be provided in the scene configuration, whereas the scattered light must be determined using the scattering equation [16],

$$L_s(\mathbf{x}, \omega_o) = \int_{S^2} L_i(\mathbf{x}, \omega_i) f(\mathbf{x}, \omega_i \to \omega_o) |\cos \theta_i| \, \mathrm{d}\omega_i \,. \tag{2.2}$$

Here, f is the bidirectional scattering function (BSDF), which we will introduce in Section 2.1.3. For now, it is enough to know that it describes how much of the arriving light from ω_i is scattered towards ω_o.

Combining L_e and L_s yields the light transport equation, also called rendering equation [17], given by

$$\begin{aligned} L_o(\mathbf{x}, \omega_o) &= L_e(\mathbf{x}, \omega_o) + \int_{S^2} L_i(\mathbf{x}, \omega_i) f(\mathbf{x}, \omega_i \to \omega_o) |\cos \theta_i| \, \mathrm{d}\omega_i \\ &= L_e(\mathbf{x}, \omega_o) + \int_{S^2} L_o(\mathbf{x}', -\omega_i) f(\mathbf{x}, \omega_i \to \omega_o) |\cos \theta_i| \, \mathrm{d}\omega_i \,, \end{aligned} \tag{2.3}$$

where $\mathbf{x}'$ is obtained by casting a ray from $\mathbf{x}$ towards ω_i. Here, the recursive nature of the rendering equation becomes apparent: We have L_o on both sides of the equation.

To get rid of directional variables ω_i and ω_o, we can rewrite the equation to be solely in the area domain. For this, we define $L(\mathbf{x}' \to \mathbf{x})$ as the radiance leaving $\mathbf{x}'$ towards $\mathbf{x}$, i.e.

$$L(\mathbf{x}' \to \mathbf{x}) = L_i(\mathbf{x}, \widehat{\mathbf{x}' - \mathbf{x}}) = L_o(\mathbf{x}', \widehat{\mathbf{x} - \mathbf{x}'}),$$

where $\widehat{\mathbf{x} - \mathbf{x}'}$ is the direction pointing from $\mathbf{x}'$ to $\mathbf{x}$. The BSDF is rewritten as $f(\mathbf{x} \to \mathbf{x}' \to \mathbf{x}'')$, with

$$f(\mathbf{x} \to \mathbf{x}' \to \mathbf{x}'') = f(\mathbf{x}', \omega_i \to \omega_o),$$

where $\omega_i = \widehat{\mathbf{x} - \mathbf{x}'}$ and $\omega_o = \widehat{\mathbf{x}'' - \mathbf{x}'}$. Combining these leads to the three-point form of the light transport equation [16], see also Figure 2.1,

$$\begin{aligned} L(\mathbf{x}' \to \mathbf{x}'') = L_e(\mathbf{x}' \to \mathbf{x}'') \\ + \int_M L(\mathbf{x} \to \mathbf{x}') f(\mathbf{x} \to \mathbf{x}' \to \mathbf{x}'') G(\mathbf{x} \leftrightarrow \mathbf{x}') \, \mathrm{d}A(\mathbf{x}) \,. \end{aligned} \tag{2.4}$$

The geometry factor G combines the change of variables from $\mathrm{d}\omega$ to $\mathrm{d}A$ as well as the original $|\cos\theta_i|$ factor. The change of variables is given by the relation

$$\mathrm{d}\omega = \frac{\mathrm{d}A\,|\cos\theta_o|}{r^2}, \tag{2.5}$$

which will become more clear in Section 2.1.2. The geometry factor is thus defined as

$$G(\mathbf{x} \leftrightarrow \mathbf{x}') = V(\mathbf{x} \leftrightarrow \mathbf{x}')\frac{|\cos\theta_o||\cos\theta_i|}{\|\mathbf{x}-\mathbf{x}'\|^2}, \tag{2.6}$$

where θ_o is the outgoing angle at $\mathbf{x}$, θ_i the ingoing angle at $\mathbf{x}'$ and V is the visibility function with $V(\mathbf{x} \leftrightarrow \mathbf{x}') = 1$ if the line between $\mathbf{x}$ and $\mathbf{x}'$ is unobfuscated, i.e. $\mathbf{x}$ and $\mathbf{x}'$ are visible to each other.

From the three-point form, we can derive the path-integral formulation of the measurement equation [16]. A path is a sequence of points $\overline{\mathbf{p}} = (\mathbf{p_0}\ldots\mathbf{p_i}) \in \Omega$, where $\mathbf{p_i}$ are points on surfaces of the scene and Ω is the path space. The path-integral measurement equation is then defined as follows,

$$I = \int_\Omega f(\overline{\mathbf{p}})\,\mathrm{d}\mu(\overline{\mathbf{p}})\,, \tag{2.7}$$

$$\begin{aligned} f(\overline{p}) =& L_e(\mathbf{p_0} \to \mathbf{p_1}) \\ & G(\mathbf{p_0} \leftrightarrow \mathbf{p_1}) f(\mathbf{p_0} \to \mathbf{p_1} \to \mathbf{p_2}) \\ & G(\mathbf{p_1} \leftrightarrow \mathbf{p_2}) f(\mathbf{p_1} \to \mathbf{p_2} \to \mathbf{p_3}) \\ & \ldots \\ & G(\mathbf{p_{k-1}} \leftrightarrow \mathbf{p_k}) W_e(\mathbf{p_{k-1}} \to \mathbf{p_k}), \end{aligned} \tag{2.8}$$

$$\mathrm{d}\mu(\overline{\mathbf{p}}) = \mathrm{d}A(\mathbf{p_0})\,\mathrm{d}A(\mathbf{p_1})\ldots\mathrm{d}A(\mathbf{p_k})\,. \tag{2.9}$$

In the above example of a path with length k, the path starts at a light source at $\mathbf{p_0}$ which emits some quantity of light $L_e(\mathbf{p_0} \to \mathbf{p_1})$ towards $\mathbf{p_1}$. This is then reflected towards $\mathbf{p_2}$, where the amount of reflected light depends on the BSDF of the surface of $\mathbf{p_1}$, described by $f(\mathbf{p_0} \to \mathbf{p_1} \to \mathbf{p_2})$. After a few reflections, the light reaches a sensor and contributes to its measurement weighted by the sensor contribution function W_e.

Every point on a path may be on any surface in a scene. Also, paths may have arbitrary length. The amount of possible paths is therefore infinite. Hence, it is not feasible to solve this integral analytically in all but the most basic cases. Therefore, we have to solve it numerically. This will be discussed further in Section 2.2.2 where we will introduce Monte Carlo path tracing.

2.1.2 Geometry

We have previously mentioned objects and scenes. Here, we discuss how arbitrary three-dimensional objects are modeled using triangles to create scenes, and how ray casting is implemented using ray-triangle intersection tests. We also introduce the concepts of the solid angle as well as spherical coordinates that allow us to describe directions on the sphere S^2. Finally, we

introduce Rodrigues' rotation formula which we will need to rotate vectors from local to word space.

COORDINATE SYSTEM The choice of coordinate system is arbitrary. We follow the convention of some rendering APIs and use a left-handed coordinate system with y pointing towards the sky.

TRIANGLES Triangle meshes can be used to model arbitrarily detailed objects. A triangle $\triangle\mathbf{abc}$ is defined by its three vertices $\mathbf{a}, \mathbf{b}$ and $\mathbf{c}$. The normal of a triangle is calculated as follows,

$$\mathbf{n} = (\mathbf{b} - \mathbf{a}) \times (\mathbf{c} - \mathbf{a}), \tag{2.10}$$

where $\times$ denotes the cross product, which gives a vector perpendicular to its two operands. The area of a triangle is given by

$$A(\triangle\mathbf{abc}) = \frac{1}{2} \left(\|(\mathbf{b} - \mathbf{a}) \times (\mathbf{c} - \mathbf{a})\| \right). \tag{2.11}$$

The barycentric coordinates (u, v, w) of point $\mathbf{p}$ with respect to triangle $\triangle\mathbf{abc}$ are calculated as follows,

$$u = \frac{A(\triangle\mathbf{pbc})}{A(\triangle\mathbf{abc})}, \tag{2.12}$$

$$v = \frac{A(\triangle\mathbf{pca})}{A(\triangle\mathbf{abc})}, \tag{2.13}$$

$$w = \frac{A(\triangle\mathbf{pab})}{A(\triangle\mathbf{abc})}. \tag{2.14}$$

If $u + v + w = 1$ and $0 \leq u, v, w \leq 1$, the point lies inside the triangle, and is given by

$$\mathbf{p} = u\mathbf{a} + v\mathbf{b} + w\mathbf{c}. \tag{2.15}$$

Note that if $u = 1$, we have $\mathbf{p} = \mathbf{a}$. There is also an alternative notation with $\mathbf{p} = w\mathbf{a} + u\mathbf{b} + v\mathbf{c}$, which follows from setting $w = 1 - u - v$ after rearranging of $\mathbf{p} = \mathbf{a} + u(\mathbf{b} - \mathbf{a}) + v(\mathbf{c} - \mathbf{a})$.

RAY CASTING In the previous section, we have mentioned ray casting as an operator to find the closest surface given an origin and direction. Given triangle meshes, the ray casting operator is implemented using ray-triangle intersection tests.

A ray is given by the equation

$$\mathbf{r}(t) = \mathbf{o} + t\mathbf{d}, \tag{2.16}$$

where $\mathbf{o}$ is the origin of a ray, $\mathbf{d}$ is its direction and $t \in \mathbb{R}^+$ is the ray parameter.

For efficient calculation of ray-triangle intersections, the Möller-Trumbore algorithm [18] is used. The algorithm works in two steps: First, we must determine whether the ray is parallel to the plane of the triangle. If this is

the case, then it cannot intersect. If not, we must determine where on the plane the ray intersects, and check if this point lies inside the triangle.

Given a ray $\mathbf{r}(t) = \mathbf{o} + t\mathbf{d}$ and a triangle $\triangle\mathbf{abc}$, we first compute two edges of the triangle as follows,

$$\mathbf{e_1} = \mathbf{b} - \mathbf{a},$$
$$\mathbf{e_2} = \mathbf{c} - \mathbf{a}.$$

To see whether the ray is parallel, we compute the cross product $\mathbf{q} = \mathbf{d} \times \mathbf{e_2}$. This gives us a vector orthogonal to $\mathbf{d}$ and $\mathbf{e_2}$. We calculate the determinant as $d = \mathbf{e_1} \cdot \mathbf{q}$. If $d = 0$, the ray is parallel to the plane, so it cannot intersect. Otherwise, the ray intersects with the plane of the triangle.

Next, we calculate the barycentric coordinates. We could use Equation 2.12, but there is a more efficient way. As we have seen above, any point on the triangle can be expressed as a linear combination of the two edges of the triangle as follows,

$$\mathbf{p}(u, v) = \mathbf{a} + u\mathbf{e_1} + v\mathbf{e_2}.$$

For the ray to intersect with the triangle, we must have

$$\mathbf{o} + t\mathbf{d} = \mathbf{a} + u\mathbf{e_1} + v\mathbf{e_2}.$$

We can rearrange this equation to

$$\mathbf{o} - \mathbf{a} = -t\mathbf{d} + u\mathbf{e_1} + v\mathbf{e_2}.$$

To find t, u and v, we need to solve the system of linear equations,

$$\begin{pmatrix} -\mathbf{d}_x & \mathbf{e_1}_x & \mathbf{e_2}_x \\ -\mathbf{d}_y & \mathbf{e_1}_y & \mathbf{e_2}_y \\ -\mathbf{d}_z & \mathbf{e_1}_z & \mathbf{e_2}_z \end{pmatrix} \begin{pmatrix} t \\ u \\ v \end{pmatrix} = \mathbf{o} - \mathbf{a}.$$

We can use Cramer's rule to solve this system. Cramer's rule states that a system of n linear equations for n unknowns, represented as $\mathbf{Ax} = \mathbf{b}$, has a unique solution with

$$x_i = \frac{\det \mathbf{A_i}}{\det \mathbf{A}},$$

where $\mathbf{A_i}$ is the matrix with the i-th column of $\mathbf{A}$ replaced by column vector $\mathbf{b}$. Note that $\mathbf{A}$ must be invertible, i.e. its determinant must be nonzero.

The determinant of a 3×3 matrix may be calculated using

$$\det \begin{pmatrix} a_1 & b_1 & c_1 \\ a_2 & b_2 & c_2 \\ a_3 & b_3 & c_3 \end{pmatrix} = (\mathbf{a} \times \mathbf{b}) \cdot \mathbf{c} = -(\mathbf{a} \times \mathbf{c}) \cdot \mathbf{b} = -(\mathbf{c} \times \mathbf{b}) \cdot \mathbf{a}.$$

In our case, we have $\det \mathbf{A} = \mathbf{e_1} \cdot (\mathbf{d} \times \mathbf{e_2}) = d$ as defined earlier. To get u, v, and t, we finally calculate

$$t = \frac{\mathbf{e_2} \cdot (\mathbf{b} \times \mathbf{e_1})}{d}, \tag{2.17}$$

$$u = \frac{\mathbf{b} \cdot (\mathbf{d} \times \mathbf{e_2})}{d}, \tag{2.18}$$

$$v = \frac{\mathbf{d} \cdot (\mathbf{b} \times \mathbf{e_1})}{d}, \tag{2.19}$$

where $\mathbf{b} = \mathbf{o} - \mathbf{a}$. We usually calculate u and v first, checking that $0 \leq u, v \leq 1$ and $u + v \leq 0$, i.e. the ray intersects the triangle, and if this is the case, calculate t.

SPHERICAL COORDINATES Two ways of parameterizing the sphere are considered here: Cartesian coordinates and spherical coordinates. Cartesian coordinates are simply the (x, y, z) coordinates on the surface of a sphere. Spherical coordinates are given by the polar angle θ, azimuth angle ϕ and radius r. The conversion between the two is given by

$$x = r \sin\theta \cos\phi, \tag{2.20}$$

$$y = r \cos\theta, \tag{2.21}$$

$$z = r \sin\theta \sin\phi. \tag{2.22}$$

The infinitesimal area element $\mathrm{d}A$ on the sphere is given by

$$\mathrm{d}A = r^2 \sin\theta \, \mathrm{d}\theta \, \mathrm{d}\phi \,. \tag{2.23}$$

SOLID ANGLES Solid angles describe how much space an object takes in the field of view of some observer. Or, equivalently, it describes how much of the surface of a sphere around an observer is subtended by an object, see Figure 2.2. Solid angles can also be understood as a generalization of the two-dimensional angle, which can be interpreted as describing how much of the surface of a unit circle is covered by some object. The unit for solid angles is the dimensionless steradian (sr).

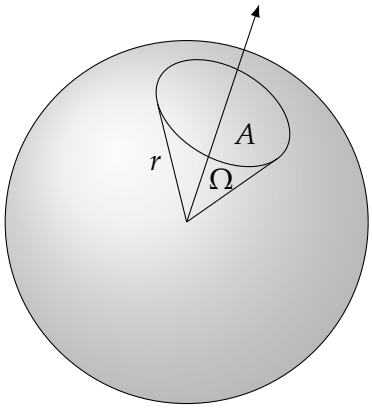


Figure 2.2: Definition of solid angle Ω. A is the area of the sphere subtended by some object.

Given the subtended area of the sphere A and its radius r, the steradian Ω is given by

$$\Omega = \frac{A}{r^2}. \tag{2.24}$$

The differential in spherical coordinates is thus given by

$$\mathrm{d}\Omega = \frac{\mathrm{d}A}{r^2} = \sin\theta \, \mathrm{d}\theta \, \mathrm{d}\phi \,. \tag{2.25}$$

In Section 2.1.1, we have used the following relationship for deriving the path-integral form of the measurement equation,

$$\mathrm{d}\omega = \frac{\mathrm{d}A \, |\cos\theta_o|}{r^2}.$$

The additional $|\cos\theta_o|$ factor comes from the fact that we are integrating over surfaces in a scene that are projected onto the unit sphere. The amount of light exiting from a point $\mathbf{p}$ on a surface towards the surface of the unit sphere depends on the angle θ_o between direction and surface normal at point $\mathbf{p}$, following Lambert's law.

ROTATION When a ray reflects off a material, we calculate the new direction ω with respect to local space, where the y-axis is equal to the normal of the surface and the origin is at the intersection point. The direction generated this way has to be rotated such that it is correct in world space.

To do so, we apply Rodrigues' rotation formula, which is given by

$$\mathbf{v}' = \mathbf{v}\cos\theta + (1 - \cos\theta)(\mathbf{k} \cdot \mathbf{v})\mathbf{k} + \mathbf{k} \times \mathbf{v}\sin\theta, \tag{2.26}$$

where $\mathbf{v}$ is the vector to be rotated, $\mathbf{k}$ is the (normalized) axis of rotation and θ is the angle of rotation.

Given the world space normal $\mathbf{n_1} = (0, 1, 0)$ and the unit normal of the surface $\mathbf{n_2}$, we rotate a direction towards $\mathbf{n_2}$ on the plane $\mathbf{n_1} \times \mathbf{n_2}$ by setting

$$\mathbf{k} = \frac{\mathbf{n_1} \times \mathbf{n_2}}{|\mathbf{n_1} \times \mathbf{n_2}|},$$
$$\theta = \operatorname{acos}(\mathbf{n_1} \cdot \mathbf{n_2}).$$

2.1.3 Materials

The material of an object is described by a bidirectional scattering distribution function (BSDF). A BSDF is a three-dimensional function $f(\mathbf{p}, \omega_o, \omega_i)$ mapping a point on the surface $\mathbf{p}$, an ingoing direction ω_i and an outgoing direction ω_o to the fraction of incident radiance that is radiated along the outgoing direction ω_o. Because of energy conservation, the overall exitant radiance must be less than or equal to the overall incident radiance at point $\mathbf{p}$, so for any ω_i,

$$\int_{S^2} f(\mathbf{p}, \omega_o, \omega_i) \cos\theta_o \, \mathrm{d}\omega_o \leq 1. \tag{2.27}$$

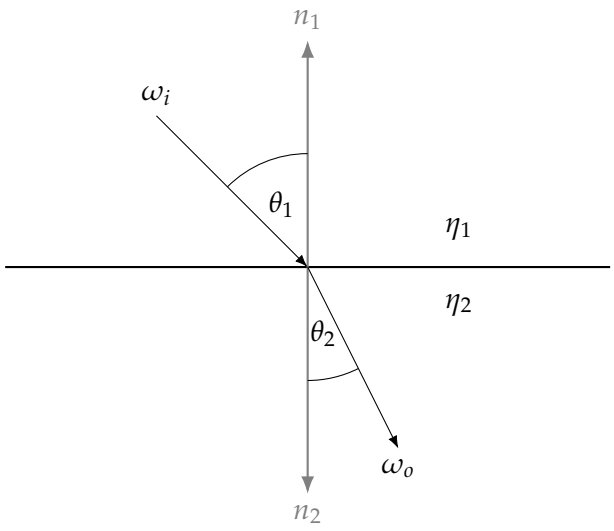


Figure 2.3: Refraction of incoming ray ω_i towards ω_o due to Snell's law.

Note that the integral can be less than one because of possible absorption by the material.

Two type of materials are used in this thesis. A diffuse and specular material for describing the ground, and a transmissive material for describing atmospheric layers.

For diffuse-specular materials, light is either reflected specularly, e.g. a mirror, or diffusely, e.g. a concrete wall. Some amount of light may be absorbed by the surface.

For transmissive materials, light passes through the material and may be refracted, e.g. water or a lens. The refraction of rays follows Snell's law. Snell's law relates the angle of incidence θ_1, the angle of refraction θ_2 and the refractive indices η_1 and η_2 as follows,

$$\frac{\sin\theta_1}{\sin\theta_2} = \frac{\eta_2}{\eta_1}. \tag{2.28}$$

See also Figure 2.3. We are usually interested in the angle θ_2, which we can directly calculate from Snell's law,

$$\theta_2 = \arcsin\left(\frac{\eta_1}{\eta_2}\sin\theta_1\right). \tag{2.29}$$

Note that this is undefined for $|\sin\theta_1 \cdot \eta_1/\eta_2| > 1$. In that case, we have a total internal reflection: The ray is reflected into the medium with refractive index η_1. As we will see in Section 2.3.3, this can happen for specific wind and temperature profiles in atmospheric sound propagation.

2.1.4 Acceleration Structures

Each time a ray is cast, intersection with each object in a scene has to be tested. Depending on the number of objects present in a scene, this can quickly become computationally expensive.

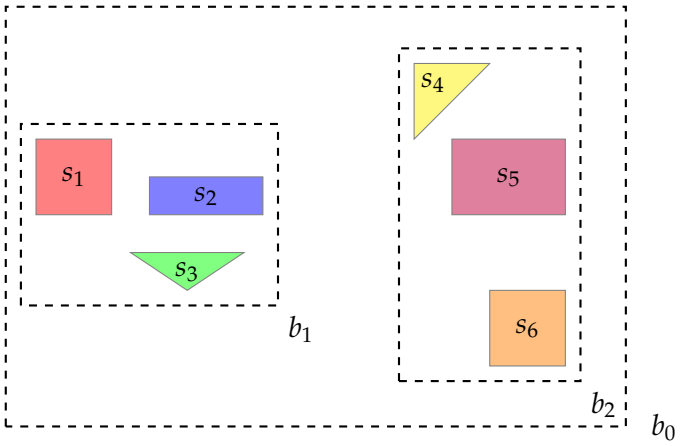


Figure 2.4: Bounding Volume Hierarchy: Instead of testing intersection with each shape separately, we can test for intersection with bounding boxes b_1 and b_2. On intersection, we can then test the shapes contained in the corresponding bounding box.

There are smarter approaches, exploiting the spacial separation and clustering of objects. Instead of testing whether a ray intersects with each object in a scene, we can test whether it intersects with a region in space containing several objects, and on intersection, test each object in that region for intersection. This approach is called Bounding Volume Hierarchy (BVH), shown in Figure 2.4. Starting from this basic idea, we can recursively divide regions into smaller and smaller regions, creating a tree structure. The BVH-tree for the space division given in Figure 2.4 is shown in Figure 2.5.

AXIS-ALIGNED BOUNDING BOXES The area that surrounds a set of objects is called a bounding box. In this work, only axis-aligned bounding boxes (AABB) are used. As the name suggests, these boxes are defined by intervals $I_x = [x_{\min}, x_{\max}]$, one per axis, defining rectangular prisms that completely surround a set of objects. It is easy to test for ray-AABB intersections: For the x-axis, we solve $\mathbf{o}_x + t\mathbf{d}_x = x_{\min}$ and $\mathbf{o}_x + t\mathbf{d}_x = x_{\max}$ for t. This gives us two values $t_{\min}$ and $t_{\max}$ that form an interval $T = [t_{\min}, t_{\max}]$. Note that $t_{\min}$ does not have to be the solution to the first equation, consider for example a bounding box with $x_{\min} = -4$ and $x_{\max} = -2$ with a ray originating at $\mathbf{o}_x = 0$ with $\mathbf{d}_x = -1$. We repeat this for the other axes, and take the

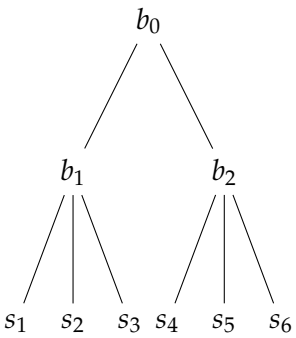


Figure 2.5: BVH tree for spacial division shown in Figure 2.4.

intersection with T each time. If, at the end, $T \neq \emptyset$, the ray intersects the box.

BVH TREE CREATION There are several methods for the creation of BVH trees, with differing degrees of efficiency in creation and traversal. Starting from the whole scene, the simplest approach is to pick an axis at random, and either divide the space in the center of all objects with respect to the chosen axis, or to sort the objects on the chosen axis and divide space by the median object, which works better in cases where objects are concentrated on certain parts of the scene. Both strategies are applied recursively, yielding a deep binary tree where each primitive is one leaf.

More involved, but also more efficient, is the Surface Area Heuristic (SAH) [19]. In the SAH method, we heuristically compare the traversal time needed when splitting a certain subspace with the traversal time needed when not splitting it, and greedily make the optimal decision based on this heuristic.

We will now describe SAH in detail. Given a subspace b with primitives $s_1, \ldots, s_n$, we know that for any ray intersecting b, we must check if it intersects with any of the n primitives. Assume that this takes some time $t_{\text{intersect}}$. If we choose not to further divide b, the computational cost for checking for intersection will be

$$c = n \cdot t_{\text{intersect}}. \tag{2.30}$$

Since we only have triangles in our scene, we know that $t_{\text{intersect}}$ must be constant.

We could alternatively split the region b into two smaller regions b_1 and b_2. The expected cost then depends on the probability that a ray intersects either region. These probabilities are denoted by $p(b_1)$ and $p(b_2)$, respectively. We also need to take into account the cost of determining which of the two regions the ray intersects. We denote this as t_{traverse}. The final expected cost when splitting the region into regions b_1 and b_2 is

$$c(b_1, b_2) = t_{\text{traverse}} + p(b_1) \sum_{s_i \in b_1} t_{\text{intersect}}(s_i) + p(b_2) \sum_{s_i \in b_2} t_{\text{intersect}}(s_i). \tag{2.31}$$

The probabilities $p(b_1)$ and $p(b_2)$ are easily computed. It follows from geometric probability that for any convex volume V_1 contained in another convex volume V_2, the probability of a ray passing through V_2 also passing through V_1 is the ratio of their surface areas $p(A \mid B) = A(V_1)/A(V_2)$, see [19]. This is where the name of this method comes from.

To decide whether to divide a region of space into smaller regions, we simply compare c with $c(b_1, b_2)$. But how do we choose b_1 and b_2 such that $c(b_1, b_2)$ is minimal?

First, we need to pick an axis that we want to divide space on. We could do this at random, but it is better to pick the axis where primitives are spread on the most, i.e. the longest axis of the current region, with respect to the centroids of primitives. Then, we divide this space into a predefined number of buckets. We use $n = 12$ buckets as suggested in [19].

For each bucket, we keep the number of primitives and the surface area of a minimal bounding box containing the respective primitives. Using this information, we can perform a combination of a prefix sum and suffix sum to obtain the bucket boundary division which yields minimal cost. For the prefix sum, we iterate over the buckets and store for each bucket the aggregated approximate cost of intersecting all objects that came before. This gives us $p(b_{-i}) \sum_{s_i \in b_{-i}} t_{\text{intersect}}(s_i)$ for each bucket i. The suffix sum is computed analogously, starting at the end of the bucket list, giving us $p(b_{+i}) \sum_{s_i \in b_{+i}} t_{\text{intersect}}(s_i)$ for each bucket i. Combining these two results, we have the cost of intersection for the regions b_{-i} and b_{+i} that are obtained from dividing b at the boundaries of bucket i. Since t_{traverse} is constant, we just pick i with minimal cost:

$$\arg\min_i \left(p(b_{-i}) \sum_{s_i \in b_{-i}} t_{\text{intersect}}(s_i) + p(b_{+i}) \sum_{s_i \in b_{+i}} t_{\text{intersect}}(s_i) \right). \tag{2.32}$$

We set t_{traverse} such that it is smaller than $t_{\text{intersect}}$, as it is generally faster to check a bounding box for intersection than it is to check a triangle for intersection. Then, we can finally compare the costs of splitting with the cost of not splitting a region, and act accordingly. We also split if not doing so would yield in a region with too many primitives. The limit is set to 255, as suggested in [19].

An alternative is the Linear Bounding Volume Hierarchy (LBVH) method, which, although less efficient than SAH for intersection tests, takes less computational time for tree construction [19].

Another method to speed up intersection tests is to make use of spatial separations predefined in a scene. In our case, this will be the separation of atmospheric layers and geometry, as we will see in Section 3.2.

2.1.5 Radiometry

So far, we have not considered the quantities carried by rays and measured by sensors. Here, we will quickly introduce the radiometric quantities used in rendering.

POWER Power ϕ measures the total amount of energy Q passing through a region of space per unit time,

$$\phi = \frac{\mathrm{d}Q}{\mathrm{d}t}. \tag{2.33}$$

The unit of power is $\mathrm{J/s} = \mathrm{W}$.

IRRADIANCE Irradiance measures how much power arrives at an area per unit time. Given an area A, the irradiance is

$$E = \frac{\phi}{A}. \tag{2.34}$$

The unit of irradiance is $\mathrm{W/m^2}$.

INTENSITY Intensity measures the angular density of emitted power. For a sphere, this is

$$I = \frac{\phi}{4\pi}. \tag{2.35}$$

The unit of intensity is W/sr.

RADIANCE Radiance measures irradiance with respect to solid angles, thereby taking direction into account. It is the basic unit considered in all light transport, and is defined as

$$L(\mathbf{p}, \omega) = \frac{\mathrm{d}E_\omega(\mathbf{p})}{\mathrm{d}\omega}, \tag{2.36}$$

where $E_\omega(\mathbf{p})$ is the irradiance at point $\mathbf{p}$ perpendicular to direction ω.

2.2 MONTE CARLO

2.2.1 Integration

There are accurate numerical integration techniques such as adaptive and Gaussian quadrature, but they suffer from poor rate of convergence for high dimensions or discontinuous domains [19]. A method that works well in those cases is the Monte Carlo estimator, a Monte Carlo algorithm to solve arbitrary integrals [19].

Given independent uniform random variables $X_i \in [a, b]$, the integral $\int_a^b f(x)$, where $f(x)$ is a function on the domain of X_i, can be approximated by

$$F_n = \frac{b-a}{n} \sum_{i=1}^{n} f(X_i). \tag{2.37}$$

More generally, if the random variables are sampled from any probability density function $p(x)$, the integral can be approximated by

$$F_n = \frac{1}{n} \sum_{i=1}^{n} \frac{f(X_i)}{p(X_i)}. \tag{2.38}$$

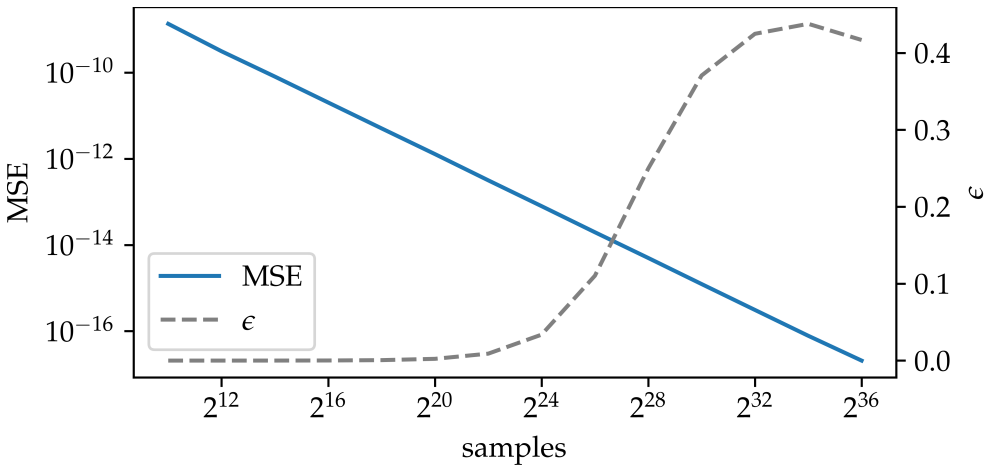


Figure 2.6: Mean squared error (MSE) versus efficiency for the `alpshigh` scene introduced in Section 4.1. A reference render with 2^{40} samples was used for calculation of the MSE. As the number of samples increases, the MSE decreases as expected, but so does the computational time required. The best efficiency is achieved with 2^{34} samples, afterwards, the additional decrease in error comes with a significant increase in computational time.

It is easy to see that the expected value of this estimator is the integral in question:

$$\begin{aligned}\mathbb{E}\left[F_n\right] &= \mathbb{E}\left[\frac{1}{n}\sum_{i=1}^{n}\frac{f\left(X_i\right)}{p\left(X_i\right)}\right] \\ &= \frac{1}{n}\sum_{i=1}^{n}\int_a^b \frac{f(x)}{p(x)}p(x)\mathrm{d}x \\ &= \frac{1}{n}\sum_{i=1}^{n}\int_a^b f(x)\mathrm{d}x \\ &= \int_a^b f(x)\mathrm{d}x.\end{aligned} \tag{2.39}$$

An important characteristic of an estimator is variance, which describes the deviation from its expected value. The variance of an estimator F is defined as

$$\mathbb{V}[F] = \mathbb{E}\left[(F - \mathbb{E}[F])^2\right]. \tag{2.40}$$

The bias of an estimator is given by

$$\beta = \mathbb{E}[F] - \int f(x)dx. \tag{2.41}$$

Another important measure is the mean squared error (MSE) of an estimator, which is defined as follows,

$$\mathrm{MSE}[F] = \mathbb{E}\left[\left(F - \int f(x)dx\right)^2\right]. \tag{2.42}$$

Importantly, for an unbiased estimator, $\mathrm{MSE}[F] = \mathbb{V}[F]$. Since in rendering we usually do not have the analytical result of the integral, we can estimate the MSE by using an accurate estimate of the integral $\tilde{F} \approx \int f(x)dx$ obtained from a simulation with a large number of samples.

The efficiency ϵ of an estimator F combines running time and mean squared error as follows

$$\epsilon[F] = \frac{1}{\mathrm{MSE}[F]\tau[F]}, \tag{2.43}$$

where $\tau[F]$ is the running time of the estimator F [19]. For an example, see Figure 2.6.

2.2.2 Path Tracing

As mentioned in Section 2.1.1, for all but the most trivial cases, it is not possible to analytically solve the measurement equation. If, however, we can sample paths $\overline{\mathbf{p}}_i$, we can estimate Equation 2.7 using Equation 2.38 as follows,

$$I \approx \frac{1}{n}\sum_{i=1}^{n}\frac{f(\overline{\mathbf{p}}_i)}{q_A(\overline{\mathbf{p}}_i)}. \tag{2.44}$$

Here, $q_A(\overline{\mathbf{p}})$ denotes the probability density for $\overline{\mathbf{p}}$. How do we sample paths $\overline{\mathbf{p}}_i$? We can always start from a light source, as any light reaching a sensor must have originated there. We sample $\mathbf{p_0}$ from some distribution q_{light} over the surfaces of the light sources of a scene. Starting paths this way is also called forwards path tracing or particle tracing.

For the next point on a path, assume the light has a spherical distribution. We can sample any direction ω on the sphere S^2, and cast a ray in this direction. If the ray hits a surface, we have the next point $\mathbf{p_1}$ of our path.

We do the same for point $\mathbf{p_1}$: We sample an outgoing direction and cast a ray to find the next point of the path. We continue this process until we either hit a sensor or a ray does not hit any surface, in which case it cannot contribute to I.

Assume we have some path $\overline{\mathbf{p}}$ created in this way, starting at a light source at $\mathbf{p_0}$ and ending at a sensor at $\mathbf{p_k}$ for some k. The probability of this path is then given by

$$q_A(\overline{\mathbf{p}}) = q_{\text{light}}(\mathbf{p_0})q_A(\mathbf{p_1})\dots q_A(\mathbf{p_k}), \tag{2.45}$$

where $q_A(\mathbf{p})$ is the probability density for point $\mathbf{p}$ in the scene. Notice that we sampled new points using directional probabilities, but the measurement equation and its estimator are in the area domain. We must convert between the two using the following relation, which follows directly from Equation 2.5 (see [16]),

$$q_A(\mathbf{p_i}) = q_\omega(\omega_{i-1}^{\text{out}}) \times \frac{|\cos\theta_i^{\text{in}}|}{\|\mathbf{p_{i-1}} - \mathbf{p_i}\|^2}, \tag{2.46}$$

where $\mathbf{p_i}$ is a point on the path, $\mathbf{p_{i-1}}$ is the previous point, $\omega_{i-1}^{\text{out}}$ is the outgoing direction at $\mathbf{p_{i-1}}$ (pointing to $\mathbf{p_i}$) and θ_i^{in} is the ingoing angle at point $\mathbf{p_i}$.

The full estimator for a single path is thus given by

$$\begin{aligned} I \approx & \frac{L_e(\mathbf{p_0} \to \mathbf{p_1})}{q_{\text{light}}(\mathbf{p_0})} \\ & \times \prod_{i=1}^{k-1} \frac{f(\mathbf{p_{i-1}} \to \mathbf{p_i} \to \mathbf{p_{i+1}}) G(\mathbf{p_{i-1}} \leftrightarrow \mathbf{p_i})}{q_A(\mathbf{p_i})} \\ & \times \frac{G(\mathbf{p_{k-1}} \leftrightarrow \mathbf{p_k}) W_e(\mathbf{p_{k-1}} \leftrightarrow \mathbf{p_k})}{q_A(\mathbf{p_k})}. \end{aligned} \tag{2.47}$$

To get rid of the geometry factor and work with directional probabilities, we calculate

$$\begin{aligned} \frac{G(\mathbf{p_i} \leftrightarrow \mathbf{p_{i+1}})}{q_A(\mathbf{p_{i+1}})} &= V(\mathbf{p_i} \leftrightarrow \mathbf{p_{i+1}}) \frac{|\cos\theta_i^{\text{out}}||\cos\theta_{i+1}^{\text{in}}|}{\|\mathbf{p_i} - \mathbf{p_{i+1}}\|^2} \cdot \frac{\|\mathbf{p_i} - \mathbf{p_{i+1}}\|^2}{q_\omega(\omega_i^{\text{out}})|\cos\theta_{i+1}^{\text{in}}|} \\ &= \frac{|\cos\theta_i^{\text{out}}|}{q_\omega(\omega_i^{\text{out}})}. \end{aligned}$$

Since the point $\mathbf{p_{i+1}}$ is generated by casting a ray from $\mathbf{p_i}$, we have $V(\mathbf{p_i} \leftrightarrow \mathbf{p_{i+1}}) = 1$. Using this equality, the estimator becomes

$$\begin{aligned} I \approx & \frac{L_e(\mathbf{p_0} \to \mathbf{p_1})}{q_{\text{light}}(\mathbf{p_0})} \\ & \times \prod_{i=1}^{k-1} \frac{f(\mathbf{p_{i-1}} \to \mathbf{p_i} \to \mathbf{p_{i+1}})|\cos\theta_{i-1}^{\text{out}}|}{q_\omega(\omega_{i-1}^{\text{out}})} \\ & \times \frac{W_e(\mathbf{p_{k-1}} \leftrightarrow \mathbf{p_k})|\cos\theta_{k-1}^{\text{out}}|}{q_\omega(\omega_{k-1}^{\text{out}})}. \end{aligned} \tag{2.48}$$

It is useful to define the throughput β for path $\overline{\mathbf{p}}$ as follows,

$$\beta(\overline{\mathbf{p}}) = \frac{1}{q_{\text{light}}(\mathbf{p_0})} \left(\prod_{i=1}^{k-1} \frac{f(\mathbf{p_{i-1}} \to \mathbf{p_i} \to \mathbf{p_{i+1}})|\cos\theta_{i-1}^{\text{out}}|}{q_\omega(\omega_{i-1}^{\text{out}})} \right) \frac{|\cos\theta_{k-1}^{\text{out}}|}{q_\omega(\omega_{k-1}^{\text{out}})}, \tag{2.49}$$

with which the estimator simplifies to

$$I \approx L_e(\mathbf{p_0} \to \mathbf{p_1}) \beta(\overline{\mathbf{p}}) W_e(\mathbf{p_{k-1}} \leftrightarrow \mathbf{p_k}). \tag{2.50}$$

2.2.3 Probability Distribution Transformations

To be able to estimate an integral using Monte Carlo, we need to draw from an arbitrary probability distribution, for example a distribution on the directions on the sphere. Most programming languages have standard libraries with built-in methods for generating uniformly distributed random values in the range $[0, 1]$. Mapping these to an arbitrary probability distribution is easy, as we will show here.

DISCRETE DISTRIBUTION Suppose we have a set of discrete events $X = \{X_1, \dots, X_n\}$ and a probability mass function p such that $\sum_i p(X_i) = 1$. We define a discrete cumulative distribution function $P_i = \sum_{j=1}^{i} p(p_j)$. To sample from p given a uniformly sampled $\xi \sim \mathcal{U}(0,1)$, we simply look for the i such that

$$P_{i-1} \leq \xi < P_i. \tag{2.51}$$

Since ξ is uniformly distributed, we have

$$p(P_{i-1} \leq \xi < P_i) = P_i - P_{i-1} = p(X_i). \tag{2.52}$$

CONTINUOUS DISTRIBUTION In the continuous case, we have a probability density function $p(x)$ with $\int p(x)\,\mathrm{d}x = 1$. The cumulative distribution function (CDF) is given by

$$P(x) = \int_{-\infty}^{x} p(y)dy. \tag{2.53}$$

Given a uniformly sampled $\xi \sim \mathcal{U}(0,1)$, we get a sample $X \sim p$ by solving the equation

$$\xi = P(X) \tag{2.54}$$

for X. This involves finding the inverse function of P. For an example, see Equation 3.3.

GENERALIZATION We have seen how to sample from any probability distribution using uniform samples in $[0,1]$, effectively transforming between two different distributions. We sometimes want to convert between distributions in different parameterizations, for example, between Cartesian and spherical coordinates of a sphere as shown in Section 2.1.2.

Given a d-dimensional random variable X with probability density function $p(x)$ and a bijection T, the densities are related by

$$p_T(y) = p_T(T(x)) = \frac{p(x)}{|\det J_T|}, \tag{2.55}$$

where J_T is the Jacobian matrix of T,

$$J_T = \begin{pmatrix} \frac{\partial T_1}{\partial x_1} & \dots & \frac{\partial T_1}{\partial x_d} \\ \vdots & \ddots & \vdots \\ \frac{\partial T_d}{\partial x_1} & \dots & \frac{\partial T_d}{\partial x_d} \end{pmatrix}.$$

See [19] for a derivation in the 1D-case, which is easily generalized to higher dimensions.

Consider, for example, the conversion between polar coordinates (r, θ) and Cartesian coordinates (x, y) of a disk. The transformation is given by

$$x = r\cos\theta, \tag{2.56}$$
$$y = r\sin\theta. \tag{2.57}$$

The Jacobian is given by

$$J_T = \begin{pmatrix} \frac{\partial x}{\partial r} & \frac{\partial x}{\partial \theta} \\ \frac{\partial y}{\partial r} & \frac{\partial y}{\partial \theta} \end{pmatrix} = \begin{pmatrix} \cos\theta & -r\sin\theta \\ \sin\theta & r\cos\theta \end{pmatrix}.$$

The determinant is given by $r(\cos^2\theta + \sin^2\theta) = r$, we thus have $p(x, y) = p(r, \theta)/r$. Sampling $(x, y) \sim p(x, y)$ for some $p(x, y)$, we get the density function for polar coordinates as

$$p(r, \theta) = rp(x, y). \tag{2.58}$$

In the three-dimensional case, we convert between spherical coordinates and Cartesian coordinates as follows. Given the conversion described in Equation 2.20, the Jacobian is given by

$$J_T = \begin{pmatrix} \frac{\partial x}{\partial \theta} & \frac{\partial x}{\partial \phi} & \frac{\partial x}{\partial r} \\ \frac{\partial y}{\partial \theta} & \frac{\partial y}{\partial \phi} & \frac{\partial y}{\partial r} \\ \frac{\partial z}{\partial \theta} & \frac{\partial z}{\partial \phi} & \frac{\partial z}{\partial r} \end{pmatrix} = \begin{pmatrix} r\cos\theta\cos\phi & -r\sin\theta\sin\phi & \sin\theta\cos\phi \\ -r\sin\theta & 0 & \cos\theta \\ r\cos\theta\sin\phi & r\sin\theta\cos\phi & \sin\theta\sin\phi \end{pmatrix}. \tag{2.59}$$

The determinant is $\det J_T = -r^2\sin\theta$. Hence, the conversion is as follows,

$$p(r, \theta, \phi) = r^2\sin\theta p(x, y, z). \tag{2.60}$$

2.2.4 Variance Reduction

There is a vast number of techniques to reduce the variance of a Monte Carlo estimator. A few of them are used widely in rendering [16], and will be introduced here.

IMPORTANCE SAMPLING Samples X_i where $|f(X_i)|$ is large contribute more to the value of the integral than those where $|f(X_i)|$ is small. It is useful for the probability p to assign a higher probability to points that have a higher contribution to the estimate. Since we can effectively use any distribution for sampling, as long as we weigh the contribution to the estimate accordingly (see Equation 2.39), we can use such a distribution without adding any bias to our estimator.

In rendering, one example is BSDF sampling [19]. When a ray hits a surface, the outgoing direction is sampled according to the BSDF of the material, sampling more in the direction where a higher fraction of energy is reflected to. In cosine-weighted hemisphere sampling for Lambertian materials, for example, the probability of a direction decreases with increasing angle be-

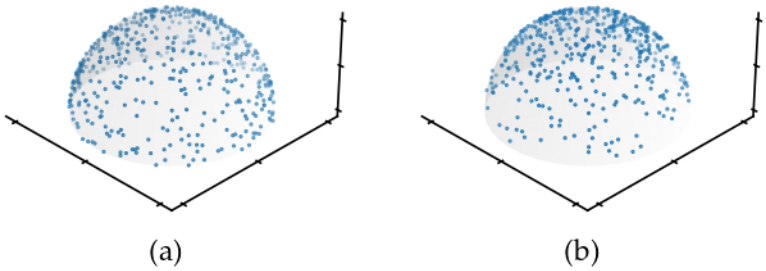


Figure 2.7: Comparison of (a) uniform hemisphere sampling and (b) cosine-weighted hemisphere sampling, with $n = 500$ samples.

tween direction and normal, as dictated by Lambert's law. See Figure 2.7 for a visualization of this, using $n = 500$ samples.

Another example of importance sampling is light or sensor sampling, where directions for the first or last vertex of a path are chosen based on some distribution on light sources or sensors. This is also called next event estimation [19].

MULTIPLE IMPORTANCE SAMPLING It sometimes may prove useful to sample from multiple distributions which match different areas of the domain of the integrand, see Figure 2.8. Multiple importance sampling (MIS) is a method to allow just that, by weighting samples from different distributions ensuring the correctness of the estimator. Given n densities p_i with n_i samples $X_{i,j}$, the MIS estimator is defined as

$$F_n = \sum_{i=1}^{n} \frac{1}{n} \sum_{j=1}^{n_i} w_i(X_{i,j}) \frac{f(X_{i,j})}{p_i(X_{i,j})}, \tag{2.61}$$

where w_i is the assigned weight.

There are different methods for weighting samples correctly. A common weighting function is the balance heuristic [16], which takes into account the likelihood of the current sample under all possible distributions,

$$w_i(x) = \frac{n_i p_i(x)}{\sum_j n_j p_j(x)}. \tag{2.62}$$

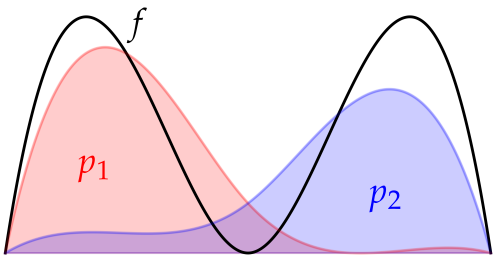


Figure 2.8: A function f for which we want to approximate the integral and two probability density functions p_1 and p_2 which cover different parts of the integral.

Another weighting heuristic is the so-called power heuristic. For any exponent β,

$$w_i(x) = \frac{(n_i p_i(x))^\beta}{\sum_j \left(n_j p_j(x)\right)^\beta}. \tag{2.63}$$

The authors of [19] suggest $\beta = 2$, as this works well in practice.

For a single sample, we pick a sampling density p_i with probability q_i and draw a sample X from it. The single sample estimator is then

$$F = \frac{w_i(X)}{q_i} \frac{f(X)}{p_i(X)}. \tag{2.64}$$

STRATIFIED SAMPLING Stratified sampling ensures a good coverage of the interval of the integral. For example, it may be used in the case of sampling directions on the hemisphere above a surface. To this end, we can divide the surface of the hemisphere Λ into strata Λ_i with

$$\bigcup_{i=1}^{n} \Lambda_i = \Lambda, \tag{2.65}$$

and sample points on each Λ_i independently [16]. The estimator for the whole domain Λ is

$$F = \sum_i v_i F_i, \tag{2.66}$$

where $v_i \in [0, 1]$ is the fractional volume of stratum i and F_i is defined as

$$F_i = \frac{1}{n_i} \sum_{j=1}^{n_i} f\left(X_{i,j}\right). \tag{2.67}$$

We will now show that the variance can be reduced by using this technique. For a stratum i, the expected value for a single sample $X_{i,j}$ is

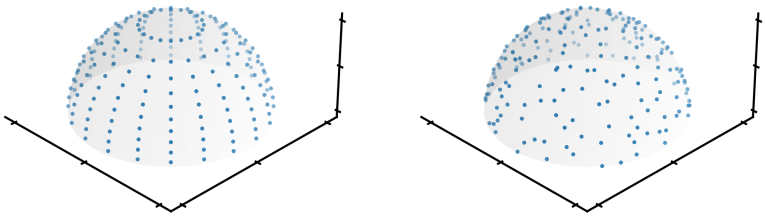
$$\mathbb{E}[f(X_{i,j})] = \frac{1}{v_i} \int_{\Lambda_i} f(x)\, \mathrm{d}x\,.$$

The variance is

$$\mathbb{V}[f(X_{i,j})] = \frac{1}{v_i} \int_{\Lambda_i} \left(f(x) - \mathbb{E}[f(X_{i,j})]\right)^2 \mathrm{d}x\,.$$

Given n_i samples in stratum i, we have

$$\mathbb{V}[F_i] = \mathbb{V}\left[\frac{1}{n_i} \sum_{j=1}^{n_i} f(X_{i,j})\right] = \frac{1}{n_i^2} \sum_{j=1}^{n_i} \mathbb{V}[f(X_{i,j})] = \frac{\mathbb{V}[f(X_{i,j})]}{n_i}.$$



(a) (b)

Figure 2.9: Comparison of (a) uniform stratified pattern and (b) uniform stratified pattern with jittering, with $n = 200$ samples.

For the overall estimator, the variance is thus

$$\begin{aligned}\mathbb{V}[F] &= \mathbb{V}\left[\sum_i v_i F_i\right] \\ &= \sum_i \mathbb{V}\left[v_i F_i\right] \\ &= \sum_i v_i^2 \mathbb{V}\left[F_i\right] \\ &= \sum_i \frac{v_i^2 \mathbb{V}[f(X_{i,j})]}{n_i}.\end{aligned}$$

If we assume that $n_i = v_i n$, i.e. the number of samples of a stratum is proportional to its volume, we have

$$\mathbb{V}[F] = \frac{1}{n}\sum_i v_i \mathbb{V}[f(X_{i,j})].$$

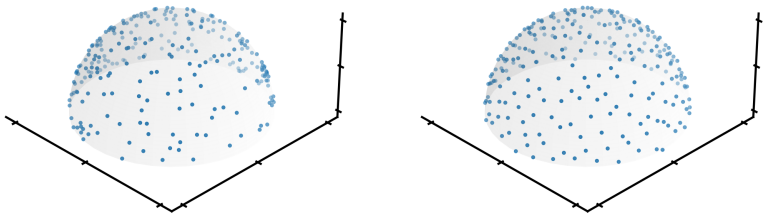
We can compare this to unstratified sampling by realizing that unstratified sampling is equivalent to picking a stratum I at random according to a probability density defined by the v_i's, and then picking a random sample X in it. The probability of X is conditional on I, which allows us to use conditional probability to show,

$$\mathbb{V}[F] = \mathbb{V}[F] - \sum_{i=1}^{n_i} v_i(\mu_i - Q)^2,$$

where Q is the mean of f over the complete domain Λ. See [19] for a detailed derivation. Note that $v_i(\mu_i - Q)^2 \geq 0$, so variance cannot increase by using stratified sampling.

Usually, we use one sample per strata. In a uniform stratified pattern, a single sample is placed in the center of each stratum. For less aliasing, jittering can be applied: Each point is moved uniformly at random inside its stratum. See Figure 2.9 for a comparison of the two methods.

HALTON SEQUENCE Jittered stratified samples can still sometimes clump together and leave regions on the surface of a hemisphere empty. An alter-



(a) (b)

Figure 2.10: Comparison of (a) uniform hemisphere sampling and (b) hemisphere sampling using the Halton sequence, with $b = 2$ for ξ_1, $b = 3$ for ξ_2 and $n = 250$ samples.

native approach for ensuring coverage of the interval of the integral is the use of low-discrepancy sequences.

A low-discrepancy sequence is a sequence which comes close to an equidistributed sequence, covering the domain evenly. One such sequence is the Halton sequence, named after J. H. Halton [20].

The Halton sequence is based on the radical inverse of a number. The radical inverse Φ_b is built by taking the digits of a number in a given base b, reflecting these about the radix point and interpreting the resulting number in base b. For example,

$$\Phi_{10}(42) = 0.24,$$
$$\Phi_2(2) = \Phi_2(10_2) = (0.01)_2 = \frac{1}{4}.$$

The Halton sequence for a given base b is then defined as

$$(\Phi_b(0), \Phi_b(1), \Phi_b(2), \dots). \tag{2.68}$$

For multidimensional sampling, we use different bases for the different dimensions to avoid correlation. This can be observed in Figure 2.10, where $b = 2$ was used for ξ_1 and $b = 3$ for ξ_2. The Halton-sequence is deterministic, but can be randomized by randomly permutating dimensions [19].

RUSSIAN ROULETTE Russian Roulette allows us to terminate rays with some probability, weighting non-terminated rays in a way such that the overall estimate stays correct. Given a termination probability q, which may be based on values that indicate a low contribution to the overall estimate, the ray is terminated with this probability or weighted by a factor $1/(1-q)$. For a uniform random value $\xi \in [0, 1]$, the new estimator is thus

$$F' = \begin{cases} \frac{F}{1-q} & \xi > q, \\ 0 & \text{else.} \end{cases} \tag{2.69}$$

The Russian Roulette estimator is unbiased (if F is unbiased), since

$$\mathbb{E}[F'] = (1-q)\left(\frac{\mathbb{E}[F]}{1-q}\right) + q \cdot 0 = \mathbb{E}[F].$$

2.3 ACOUSTICS

2.3.1 Fundamentals

WAVE EQUATION A sound wave traveling through a medium causes a change in local air pressure away from a given ambient static pressure. This change, denoted as p, is called acoustic pressure. Let $p(x,t)$ be the acoustic pressure at time t and position x. The one-dimensional wave equation is given by

$$\frac{\partial^2 p}{\partial x^2} - \frac{1}{c^2}\frac{\partial^2 p}{\partial t^2} = 0, \tag{2.70}$$

where c is the speed of sound. Since this thesis deals solely with the geometrical model of acoustics (see Section 2.3.2), we refer interested readers to Pierce [21] for a great reference.

DIFFRACTION AND REFRACTION Diffraction is the bending of waves, either around obstacles or through openings. It plays an important role in the propagation of sound waves. Diffraction follows directly from the Huygens-Fresnel principle.

The Huygens-Fresnel principle states that every point on a wavefront is itself a source of spherical wavelets. These wavelets can interfere constructively or destructively, creating a new wavefront. See also Figure 2.11.

The amount of diffraction depends on the wavelength as well as the size of the object. For an object which is much larger than the wavelength, there is barely any diffraction. If the object is much smaller than the wavelength, sound waves bend around the object easily, creating almost the same wavefront on the other side of the object. If the object is of similar size, diffraction is most noticeable. The waves will spread around the obstacle and fill the space behind it.

Waves may be refracted when entering a medium with a different speed of sound. This happens, for example, in the atmosphere, where temperature

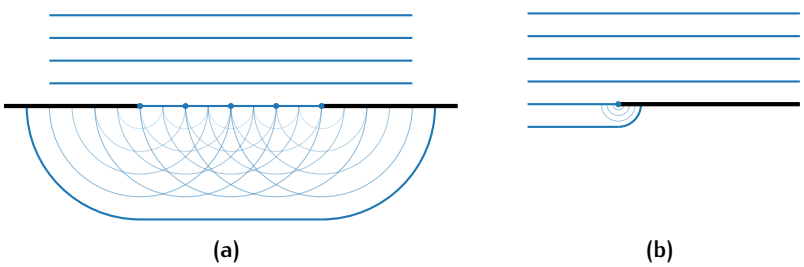


Figure 2.11: Wave diffraction due to the Huygens-Fresnel principle (a) through an opening and (b) around an obstacle.

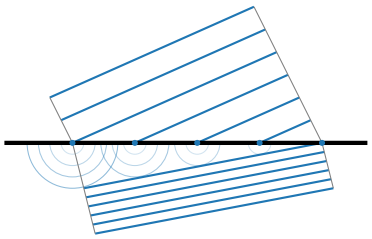


Figure 2.12: Wave refraction into a medium with lower speed of propagation, due to the Huygens-Fresnel principle.

and wind influence the speed of sound, leading to waves bending towards or away from the ground. The effect can also be explained by the Huygens-Fresnel principle, see Figure 2.12.

QUANTITIES Sound intensity E is the acoustic equivalent of irradiance (see Equation 2.34), and is given by

$$E = \frac{\phi}{A},$$

where ϕ is the power carried by the sound wave and A is the area where this power is measured. Specifically, sound intensity describes the power carried by a sound wave per unit area perpendicular to that area in $\mathrm{W\,m^{-2}}$.

Sound intensity decreases with distance due to the inverse-square law. Given a sound intensity at a distance of r_1 from a sound source, the sound intensity at a distance r_2 is given by

$$E(r_2) = E(r_1)\left(\frac{r_1}{r_2}\right)^2. \tag{2.71}$$

The sound intensity level is given by

$$L_E = 10\log_{10}\left(\frac{E}{E_0}\right)\mathrm{dB}, \tag{2.72}$$

where $E_0 = 1\,\mathrm{pW/m^2}$ is the reference sound intensity in air. The sound pressure level is given by

$$L_p = 20\log_{10}\left(\frac{p}{p_0}\right)\mathrm{dB}, \tag{2.73}$$

where $p_0 = 20\,\mu\mathrm{Pa}$ is the reference sound pressure in air.

Since $E \propto p^2$, we have

$$\begin{aligned} L_E &= 10\log_{10}\left(\frac{E}{E_0}\right)\mathrm{dB} \\ &= 10\log_{10}\left(\frac{p}{p_0}\right)^2\mathrm{dB} \\ &= L_p. \end{aligned} \tag{2.74}$$

To be able to quickly compare different noise footprints, we use the overall sound pressure level (OASPL) [22], which aggregates sound pressure levels at different frequency bands f in the following way:

$$\text{OASPL} = 10 \log_{10} \sum_f 10^{L_p(f)/10}. \tag{2.75}$$

FREQUENCY BANDS The audible range of sound is generally assumed to be 20 Hz to 20 000 Hz. This range may be split into sets of frequencies called bands, where each band covers a range of frequencies.

The usual division of the audible spectrum in acoustic analysis are octave bands and $1/3$ octave bands, starting from a middle frequency of 1000 Hz. An octave is defined by two frequencies f_1 and f_2 with $f_2 = 2f_1$. The center frequency of an octave band is defined by

$$f_c = \sqrt{2} f_{\text{min}} = \frac{f_{\text{max}}}{\sqrt{2}}, \tag{2.76}$$

where f_{min} and f_{max} denote the lower and upper bound of a frequency band.

For $1/3$ octave bands, we further divide each octave into three sections. The relationship between f_c, f_{min} and f_{max} is then

$$f_c = 2^{1/6} f_{\text{min}} = \frac{f_{\text{max}}}{2^{1/6}}, \tag{2.77}$$

since $\left(\sqrt{2}\right)^{1/3} = 2^{1/6}$.

DOPPLER EFFECT Frequencies of a moving source are shifted due to the Doppler effect. In this thesis, only stationary receivers are considered. Given an emitted frequency f_0, speed of sound c and speed of the sound source v, the observed frequency is given by

$$f = \left(\frac{c}{c \mp v} \right) f_0, \tag{2.78}$$

where the sign of v is positive if the source is moving away from the receiver, and negative otherwise.

In the three-dimensional case, we must take into account the unit direction of the wavefront $\mathbf{d}$, and the velocity becomes a vector $\mathbf{v}$. Let θ be the angle between $\mathbf{d}$ and $\mathbf{v}$, and let $\|\mathbf{d}\| = 1$,

$$f = \left(\frac{c}{c - \|v\| \cos\theta} \right) f_0 = \left(\frac{c}{c - \mathbf{d} \cdot \mathbf{v}} \right) f_0. \tag{2.79}$$

Applying the Doppler effect to frequency bands can lead to aliasing, i.e. artifacts due to the low resolution of frequency bands with respect to the frequency spectrum.

A-WEIGHTING To account for difference in perception of loudness perceived by the human ear, A-weighting [23] can be applied. A-weighting assigns lower weights to low frequencies ($< 10^3$ Hz) and to high

frequencies ($> 10^4$ Hz), as the human ear is less sensitive to these parts of the spectrum. The formulas for A-weighting are given as follows,

$$\begin{aligned} R_A(f) =& \frac{1}{(f^2+20.6^2)(f^2+12194^2)} \\ & \cdot \frac{12194^2 f^4}{\sqrt{(f^2+107.7^2)(f^2+737.9^2)}}, \\ A(f) =& 20\log_{10}(R_A(f)) - 20\log_{10}(R_A(1000)) \approx 20\log_{10}(R_A(f)) + 2. \end{aligned} \tag{2.80}$$

SPEED OF SOUND The speed of sound in the atmosphere depends on temperature T and specific humidity q,

$$c = \sqrt{\gamma_a R_a T(1+0.511q)}, \tag{2.81}$$

where $\gamma_a = 1.4$ is the ratio of specific heats for dry air and $R_a = 287.058\,\mathrm{m}^2/(\mathrm{s}^2\mathrm{K})$ is the gas constant for dry air [24]. The specific humidity may be ignored, leading to the formula

$$c = \sqrt{\gamma_a R_a T}, \tag{2.82}$$

as also used in other works dealing with atmospheric sound propagation such as [12] and [13].

2.3.2 Geometrical Acoustics

Solving the wave equation numerically is computationally expensive [25]. For far-field acoustics in complex outdoor scenes, computation can take weeks. Hence, a more efficient way of modeling sound must be used. One such model is geometrical acoustics (GA).

The central idea of GA is to model the wavefront of a sound wave as rays, as is done for light in physically based rendering [19]. Ray-based methods usually work in the domain of sound energy as opposed to sound pressure [26]. The ray carries the necessary information for modeling the underlying wave in this way, i.e. frequency and energy.

A major caveat of GA methods is that they do not inherently model wave phenomena such as diffraction and interference, albeit methods to emulate this exist (for example, see [27] and [28]). Depending on the geometry of a scene, diffraction may not be that relevant. The longest audible wavelength is about 17 m. In large outdoor scenes, the geometry is usually not detailed enough for diffraction to matter.

Diffuse and specular reflections, on the other hand, are easy to model. However, since we do not model the phase of sound, interference due to reflection will be treated as constructive interference.

2.3.3 Atmospheric Effects

The established approach to modeling ray refraction due to temperature and wind gradients in the atmosphere is to separate it into layers with linearly varying temperature and wind between layers and solve a differential equation for the ray path between layers [11] [12] [13]. We derive the differential equations here and show an alternative solution that can easily be integrated into classical ray tracing with linear rays. Afterwards, we take a look at a method for calculating the attenuation of sound in the atmosphere.

REFRACTION DUE TO VARYING PROPAGATION SPEED We have already looked at how rays refract on interaction with a medium with different refractive index, see Section 2.1.3. Here, we are instead interested in how rays refract in a medium with continuously changing speed of propagation.

The speed of sound in the atmosphere depends on temperature and humidity [24]. We assume a stratified atmosphere, i.e. an atmosphere where temperature and humidity only change with height. Let $c(y)$ be the speed of sound at height y.

Assuming a linear change in the speed of sound, we have

$$c(y) = c_0 + ky,$$

where $k = \frac{\mathrm{d}c}{\mathrm{d}y}$. The refractive index at height y is given by

$$n = \frac{c_0}{c(y)} = \frac{c_0}{c_0 + ky}.$$

From Snell's law, Equation 2.28, it follows that

$$n \sin(\theta) = \text{const.}$$

Taking the differential,

$$\frac{\mathrm{d}}{\mathrm{d}y}\left(n \sin(\theta)\right) = \frac{\mathrm{d}}{\mathrm{d}y}\left(\frac{c_0}{c(y)} \sin(\theta)\right) = 0.$$

Applying the product rule,

$$\frac{c_0}{c(y)} \frac{\mathrm{d}\sin(\theta)}{\mathrm{d}y} + \sin(\theta) \frac{\mathrm{d}}{\mathrm{d}y}\left(\frac{c_0}{c(y)}\right) = 0.$$

We first compute the derivative of the right side,

$$\frac{\mathrm{d}}{\mathrm{d}y}\left(\frac{1}{c(y)}\right) = -\frac{1}{c(y)^2} \frac{\mathrm{d}c(y)}{\mathrm{d}y} = -\frac{k}{(c_0 + ky)^2}.$$

Substituting this back into the previous expression,

$$\frac{c_0}{c(y)} \frac{\mathrm{d}\sin(\theta)}{\mathrm{d}y} + c_0 \sin(\theta)\left(-\frac{k}{c(y)^2}\right) = 0.$$

We divide by $c_0/c(y)$ to get

$$\frac{\mathrm{d}\sin(\theta)}{\mathrm{d}y} - \frac{k\sin(\theta)}{c(y)} = 0.$$

Applying the chain rule to the left term,

$$\frac{\mathrm{d}\sin(\theta)}{\mathrm{d}y} = \cos(\theta)\frac{\mathrm{d}\theta}{\mathrm{d}y},$$

yields

$$\cos(\theta)\frac{\mathrm{d}\theta}{\mathrm{d}y} - \frac{k\sin(\theta)}{c(y)} = 0.$$

Solving this for $\frac{\mathrm{d}\theta}{\mathrm{d}y}$,

$$\frac{\mathrm{d}\theta}{\mathrm{d}y} = \frac{k\sin(\theta)}{c(y)\cos(\theta)} = \frac{k}{c(y)}\tan(\theta).$$

We had defined $k = \frac{\mathrm{d}c}{\mathrm{d}y}$, therefore the equation becomes

$$\frac{\mathrm{d}\theta}{\mathrm{d}y} = \frac{\mathrm{d}c}{\mathrm{d}y}\frac{1}{c(y)}\tan(\theta). \tag{2.83}$$

This gives us the change of direction with height. For the horizontal change of position, we know that $\tan\psi = \frac{\mathrm{d}y}{\mathrm{d}x}$ when ψ is the angle between the horizontal and a ray. Since θ describes the angle between the vertical (normal) and a ray, we have to flip x and y. The change in x is therefore given by

$$\frac{\mathrm{d}x}{\mathrm{d}y} = \tan\theta. \tag{2.84}$$

These two ordinary differential equations allow us to model the ray propagation.

To be able to use classical rendering methods, i.e. simple line intersection tests and transmissive BSDFs, we approximate the above phenomenon by splicing the atmosphere in layers and applying Equation 2.28 when a ray intersects with a new layer. The number of layers highly influences the accuracy of this approach, as can be seen in Figure 2.13, where 4th-order Runge-Kutta was used to solve the ordinary differential equations (ODE).

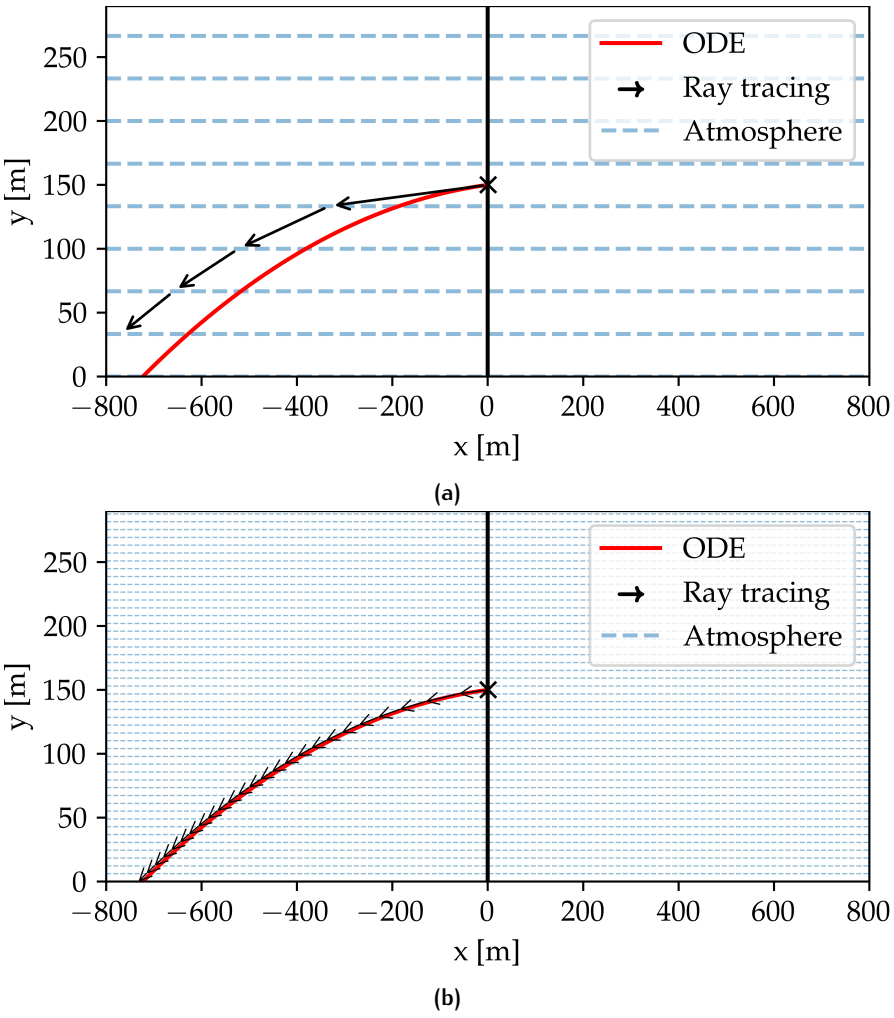


Figure 2.13: Approximation of refraction due to continuously changing speed of propagation. Propagation speed $c(0\text{m}) = 300\text{m/s}$ and $c(300\text{m}) = 340\text{m/s}$, similar to an atmospheric profile in the evening, where air is cooler towards the ground. Using 10 and 50 layers 0 m and 300 m in (a) and (b), respectively.

REFRACTION DUE TO WIND Movement of the medium also leads to refracting of rays. In the case of the atmosphere and sound, this movement is due to wind [29].

Given a wind velocity $w \ll c$, we have

$$\frac{\sin\theta}{c + w\sin\theta} = \text{const}, \tag{2.85}$$

as shown in [13]. Equivalent formulations can be found in [29] and [30]. The assumption that wind speed is much smaller than sound speed is important, as a ray path in a moving medium is not perpendicular to its wave front (refer to the eikonal equation in [21]), but it has been shown that using this assumption yields acceptable results [31].

This method is also called the effective sound speed approximation [31], since it applies Snell's law with propagation speed according to wind, i.e. given speed c, unit direction of the ray $\mathbf{s}$ and wind $\mathbf{w}$,

$$c_{\text{eff}} = c + \mathbf{s}\cdot\mathbf{w} = c + \mathbf{w}_\perp \sin\theta, \tag{2.86}$$

where the last equality is true in a two-dimensional setting if the $\mathbf{w}_\perp$ describes the horizontal wind speed and θ is the angle between the vertical and the ray: Since the direction is unit, we have

$$\mathbf{s}\cdot\mathbf{w} = \|\mathbf{w}\|\cos\psi,$$

where ψ is the angle between $\mathbf{s}$ and $\mathbf{w}$. In two dimensions with only horizontal wind, $\mathbf{w}$ spans the x-axis, so ψ is the angle between x-axis and direction. Since θ is the angle between y-axis and direction, we have $\psi = \pi/2 - \theta$, and finally

$$\|\mathbf{w}\|\cos\psi = \mathbf{w}_\perp \sin\theta.$$

Let us derive the differential equations for this case. We use the formulation used in [30],

$$\frac{c}{\sin\theta} + w = \text{const}. \tag{2.87}$$

It can easily be seen that this is equivalent to Equation 2.85. With the ray parameter t describing the point on a ray, we have

$$\frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{c}{\sin\theta} + w\right) = 0$$

Let us first look at the right-hand side, as it is easier:

$$\frac{\mathrm{d}w}{\mathrm{d}t} = \frac{\mathrm{d}w}{\mathrm{d}y}\frac{\mathrm{d}y}{\mathrm{d}t} = l\frac{\mathrm{d}y}{\mathrm{d}t},$$

where we set $l := \frac{\mathrm{d}w}{\mathrm{d}y}$. Now the left side:

$$\frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{c}{\sin\theta}\right) = c\frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{1}{\sin\theta}\right) + \frac{1}{\sin\theta}\frac{\mathrm{d}c}{\mathrm{d}t}.$$

The right side is familiar,

$$\frac{\mathrm{d}c}{\mathrm{d}t} = \frac{\mathrm{d}c}{\mathrm{d}y}\frac{\mathrm{d}y}{\mathrm{d}t} = k\frac{\mathrm{d}y}{\mathrm{d}t},$$

using $k := \frac{\mathrm{d}c}{\mathrm{d}y}$. For the left side,

$$c\frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{1}{\sin\theta}\right) = -\frac{c}{\sin^2\theta}\frac{\mathrm{d}\sin\theta}{\mathrm{d}t} = -c\frac{\cos\theta}{\sin^2\theta}\frac{\mathrm{d}\theta}{\mathrm{d}t}.$$

After substitution and transformation, we have

$$\frac{\mathrm{d}\theta}{\mathrm{d}t} = \frac{\frac{1}{\sin\theta}k\frac{\mathrm{d}y}{\mathrm{d}t} + l\frac{\mathrm{d}y}{\mathrm{d}t}}{c\frac{\cos\theta}{\sin^2\theta}} = \frac{\mathrm{d}y}{\mathrm{d}t}\left(\frac{k}{c\frac{\cos\theta}{\sin\theta}} + \frac{l}{c\frac{\cos\theta}{\sin^2\theta}}\right). \tag{2.88}$$

Applying basic trigonometry, we have

$$\frac{\mathrm{d}x}{\mathrm{d}t} = \sin\theta, \tag{2.89}$$

$$\frac{\mathrm{d}y}{\mathrm{d}t} = \cos\theta. \tag{2.90}$$

With these three differential equations in hand, and with starting conditions x_0, y_0 and θ_0, we can simulate the propagation of a ray as before.

Since the effective sound speed approximation turns the moving medium into a non-moving medium, we can use the adjusted version of Snell's law in Equation 2.85 for classical ray tracing. A comparison of the ray tracing method and solving the differential equation using 4th-order Runge-Kutta is shown in Figure 2.14. Here the number of layers has an even stronger effect on the accuracy, especially in areas of near-horizontal ray travel.

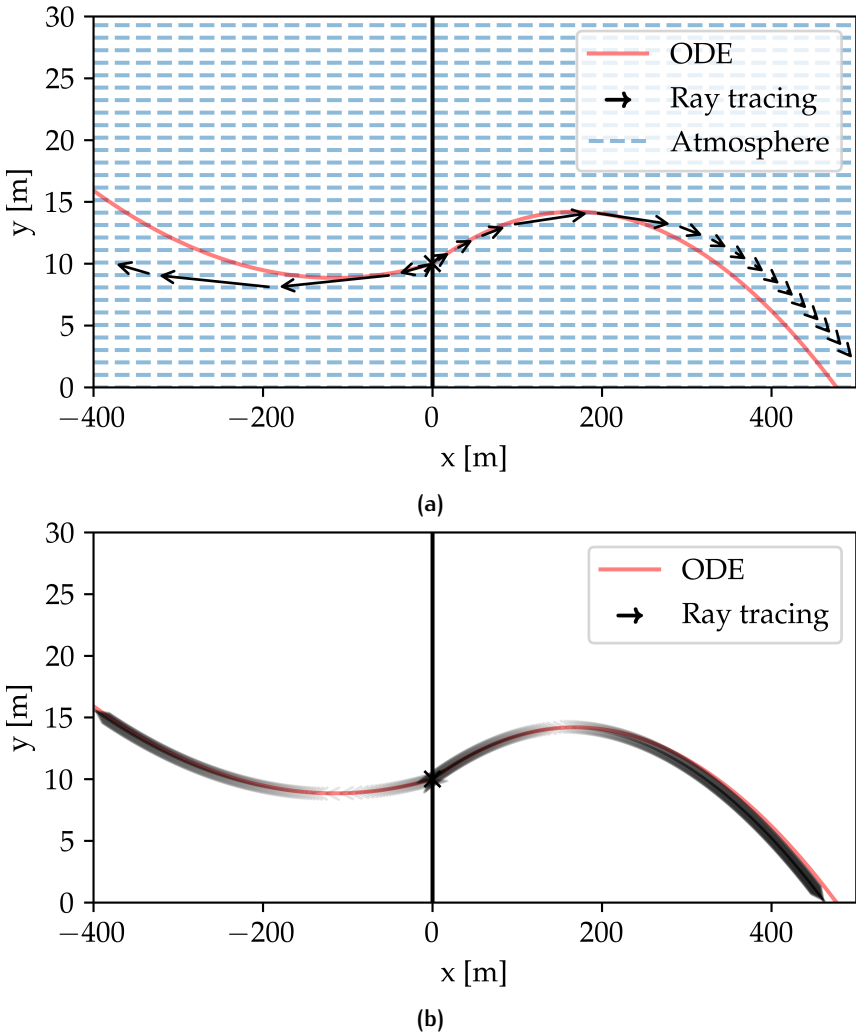


Figure 2.14: Approximation of refraction due to continuously changing speed of propagation and wind. Propagation speed $c(0\text{m}) = 339.2\text{m/s}$ and $c(100\text{m}) = 341.3\text{m/s}$, wind speed $w(0\text{m}) = 0\text{m/s}$ and $w(100\text{m}) = 8\text{m/s}$, sound source is at $y = 10\text{m}$. These parameters are taken from Figure 3 of [30]. Using 100 and 4000 layers between 0 m and 100 m in (a) and (b), respectively.

ATTENUATION Atmospheric attenuation was modeled using the method described in the ISO-9613-1 [32] standard. The vapor pressure of water at temperature T can be derived using the August-Roche-Magnus equation [33] as follows,

$$e_S = 6.1094\,\text{hPa} \cdot \exp\left(\frac{17.625 \cdot T_C}{T_C + 243.04\,^\circ\text{C}}\right),$$

where $T_C = T - 273.15$ is the temperature converted to Celsius. The actual vapor pressure is dependent on the relative humidity RH, and is given by

$$e = \text{RH} \cdot e_S.$$

The molar fraction is given by

$$h = \frac{e/10}{p} \cdot 100\,\%,$$

where p is the air pressure. Given the reference pressure $p_r = 101.325\,\text{kPa}$ and reference temperature $T_0 = 293.15\,\text{K}$, the relaxation frequencies are given by

$$f_{ro} = \left(\frac{p}{p_r}\right)\left(24 + 4.04 \cdot 10^4 \cdot h \cdot \left(\frac{0.02 + h}{0.391 + h}\right)\right),$$
$$f_{rn} = \left(\frac{p}{p_r}\right)\left(\frac{T}{T_0}\right)^{-0.5}\left(9 + 280 \cdot h \cdot \exp\left(-4.17\left(\left(\frac{T}{T_0}\right)^{-1/3} - 1\right)\right)\right).$$

The final atmospheric absorption coefficient is given by

$$\begin{aligned}\alpha = 8.686 \cdot f^2 \Bigg(& \left(1.84 \cdot 10^{-11} \cdot \frac{1}{p_a/p_r} \cdot \left(\frac{T}{T_0}\right)^{0.5}\right) \\ & + \left(\frac{T}{T_0}\right)^{-2.5}\Bigg(0.01275 \cdot \exp\left(-2239.1/T\right) \\ & \cdot \left(\frac{f_{ro} + \frac{f^2}{f_{ro}}}{1}\right) + 0.1068 \cdot \exp\left(-3352/T\right) \cdot \left(\frac{f_{rn} + \frac{f^2}{f_{rn}}}{1}\right)\Bigg)\Bigg).\end{aligned} \tag{2.91}$$

This factor is logarithmic with unit dB/m and describes the decrease in sound pressure level based on distance. Since for rendering we use sound radiance, we need to convert this factor accordingly.

For a given α, the difference in sound pressure level at a distance d is given by

$$\Delta\text{SPL} = \alpha d, \tag{2.92}$$

as described in [32]. From this, the sound intensity E' due to atmospheric absorption can be derived as follows,

$$\begin{aligned} & 10\log_{10}\left(\frac{E'}{E_0}\right) = 10\log_{10}\left(\frac{E}{E_0}\right) - \alpha d \qquad (2.93) \\ \iff & \frac{E'}{E_0} = \frac{E}{E_0}\cdot 10^{-\alpha d/10} \\ \iff & E' = E\cdot 10^{-\alpha d/10} \end{aligned}$$

Note that this method has varying accuracy based on the specific atmospheric conditions. For more details, we refer the interested reader to [32].

WIND PROFILE Only logarithmic wind profiles as described in [34] are considered here. Other profiles modeling different parts of the atmosphere exist, for example, see [35]. These could also be approximated using the introduced method of atmospheric ray tracing.

The logarithmic wind profile described in [34] was originally developed for structural design, but can also be applied to atmospheric sound propagation near a surface, up to a height of about 500 m. Given a velocity V_G at some altitude y_G and a surface roughness factor α, the wind velocity $V(y)$ at height y is given by

$$V(y) = V_G\left(\frac{y}{y_G}^{1/\alpha}\right). \qquad (2.94)$$

Different factors are given in [34]. For example, based on balloon observations of average wind velocities at an airfield in Cardington, England, $\alpha = 5.9$ for $y_G = 107\,\mathrm{m}$ and $V_G = 7.7\,\mathrm{m/s}$ was determined. This profile is shown in Figure 2.15.

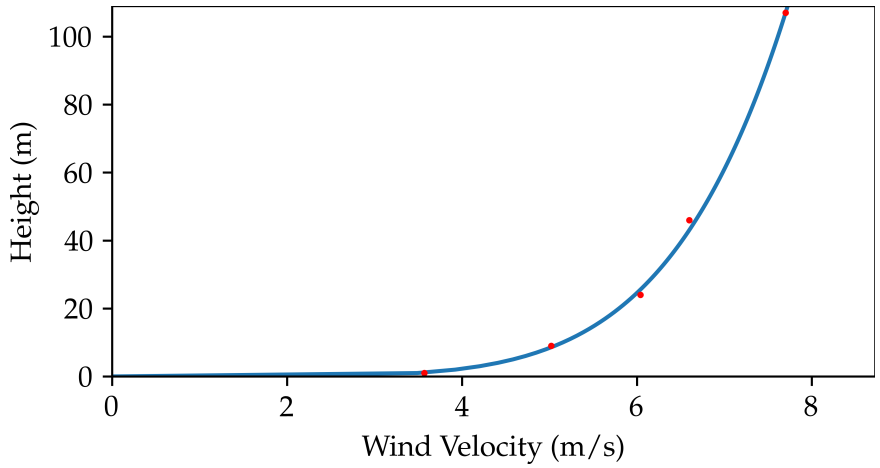


Figure 2.15: Wind profile based on recorded data at an airfield in Cardington, England, using $\alpha = 5.9$, $y_G = 107\,\mathrm{m}$ and $V_G = 7.7\,\mathrm{m/s}$, as described in [34], based on balloon observations of average wind velocities. Red points show recorded values.

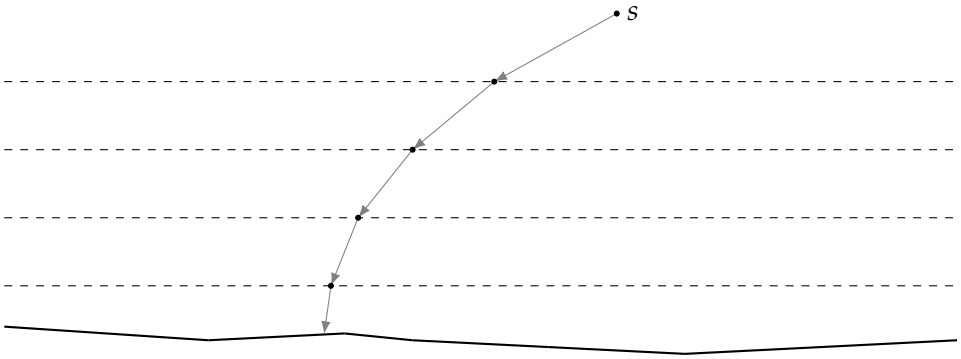


Figure 2.16: Path of a ray in a vertically stratified atmosphere. Starting from sound source s, the ray intersects the first atmospheric layer, which is a surface with a transmissive BSDF. A new ray is traced with a refracted direction. This repeats until a ray hits a surface, which acts as a sensor.

2.3.4 Acoustic Rendering

We can directly apply Monte Carlo path tracing as introduced in Section 2.2.2 to sound propagation. All of the radiometric qualities introduced in Section 2.1.5 also apply to the propagation of sound using the geometrical acoustics model [7].

Starting from a sound source position, rays are traced in directions on the sphere around the source. When a ray hits a sensor, the carried radiance is added to the "recording" associated with the sensor, weighted by the probability of the path as well as the BSDFs of all surfaces. Atmospheric layers are just surfaces with a transmissive BSDF. See also Figure 2.16 for a visualization of this process.

Sound sources are defined by their power level per frequency band $L_\phi(f)$, position $\mathbf{p}$ and velocity $\mathbf{v}$. Only point sources with a spherical directivity are considered. The initial intensity for frequency f carried by a ray is calculated by

$$\phi(f) = 10^{L_\phi(f)/10}\phi_0, \tag{2.95}$$

$$I(f) = \frac{\phi(f)}{4\pi}, \tag{2.96}$$

where $\phi_0 = 1\,\mathrm{pW}$ is the reference sound power in air. Note that $I(f)$ is not sound intensity, but intensity as defined in Section 2.1.5. We cannot get radiance, since a point source does not have any area. Point sources are generally a bit awkward to consider in a physical simulation since they are physically impossible, yet this does not invalidate the simulation results. This is also discussed shortly in [19].

Doppler shift is applied to each frequency band depending on initial direction $\mathbf{d}$ and velocity $\mathbf{v}$, and the resulting frequency is rounded to the nearest frequency band.

In related works, microphones are placed in a scene [12] [13]. Here, every triangle of a terrain in a scene is considered a microphone, as the computational complexity does not increase with the number of microphones. This has the advantage of providing a complete noise map for a terrain, but the disadvantage of not being able to model direct reflections from below a mi-

crophone. However, diffuse or specular reflections towards other terrain triangles are possible. It is generally assumed that diffuse reflections of sound also follow Lambert's cosine law as is the case for light [7] [26].

Microphones can have a Lambertian or uniform directivity modeled by the responsivity function W_e. A Lambertian directivity more closely represents a cardioid directionality in that sound reaching close to parallel to the surface contributes less to the measurement. It is inherently modeled in path tracing as we have derived in Section 3.1.3, since it is included in the geometry factor between last and second-to-last path vertex, thus $W_e(\omega_i) = 1$ for Lambertian microphones. For uniform microphones, we have $W_e(\omega_i) = 1/(\cos\theta_i)$ where θ_i is the angle between surface normal and ingoing direction ω_i.

Remember that the measurement equation is over surfaces and spheres. Thus, for any surface, given n samples and accumulated radiance S, the Monte Carlo estimator gives us the absorbed sound power,

$$\phi = \frac{S}{n}.$$

We get the sound intensity at the surface by dividing by the area of the surface, see Equation 2.34.

It is also possible to introduce time into the equation, which leads to the Room Acoustic Rendering Equation [7]. Time is essential for localization of sound, so it is necessary to introduce it for the purposes of auralization [36]. In this thesis, however, we do not take time into account as we are interested in the noise of a static scene.

VARIANCE REDUCTION We can apply importance sampling at two instances in our renderer: When sampling initial directions, and when sampling outgoing directions at surface reflections, i.e. BSDF sampling.

For the initial directions, we can choose to only sample the lower hemisphere instead of the whole sphere around a sound source. This improves performance significantly in scenes where the sound source is high above the terrain and the atmospheric profile is such that above-horizontal sound rays are refracted into the atmosphere. Going one step further, we can also use cosine-weighted hemisphere sampling for the initial directions, if we know that near-horizontal rays are unlikely to reach any terrain.

Multiple importance sampling has been used in bidirectional sound tracing for the connection of forwards and backwards paths, see [9]. In this thesis, we use it for the combination of specular and diffuse behavior in a single material.

The acoustic equivalent of light sampling is microphone sampling, where directions are chosen based on a distribution on all microphones. Due to the nature of the scenes, where every surface is a microphone, it does not make sense to apply this strategy. Furthermore, the presence of the atmosphere makes finding directions towards microphones non-trivial.

Stratified sampling and the Halton sequence can naturally be applied to the sphere or hemisphere around the sound source, as is done in [14].

Russian roulette is also straightforward in acoustic rendering, for example, see [6], where it is used to determine whether a reflection is diffuse or specular as well as for ray termination.

3 IMPLEMENTATION

In this chapter, we discuss the details of implementing the method described in the previous chapter. Starting with the implementation of the rendering engine, NoiseTracer, in Section 3.1, we move onto acceleration techniques in Section 3.2, before finally discussing our methodology to test and validate the implementation in Section 3.3.

NoiseTracer is developed entirely in C++. For a starting point, the guide "Ray Tracing in One Weekend" [37] proved very helpful.

3.1 RENDERING

The basic functionality of NoiseTracer is as follows. First, the scene has to be prepared. Heightmap data must be converted to triangles and the atmospheric data converted into layers in three-dimensional space. All of this is described in detail in Section 3.1.1.

Once the scene is set up, rays are traced from the sound source. We chose to cast rays starting from the sound source (forwards path tracing), as the probability of intersecting a sound source through atmospheric layers starting from any point on the terrain is relatively low. This also allows us to model the sound source as a point source (for which the probability of intersection would be zero). On intersection with an atmospheric layer, the ray is refracted accordingly. On intersection with a surface, the carried radiance is weighted by the aggregated throughput and measurement responsivity, and added to the accumulated radiance of that surface, before being reflected according to its BSDF. These matters will be discussed in Section 3.1.2 and Section 3.1.3.

When all rays have been traced, the sound levels are calculated per surface triangle from the accumulated radiance. This is then exported into one file per frequency band, as well as the OASPL and A-weighted OASPL. This is discussed shortly in Section 3.1.4.

3.1.1 Scene Creation

Heightmaps must be provided as Digital Terrain Elevation Data (DTED), encoded in the `TIFF` image format. A DTED file is simply a two-dimensional image with heights instead of colors, stored as 16-bit or 32-bit signed integers. This data is converted into a triangle mesh by creating two triangles per four points of heightmap data, yielding a three-dimensional approximation of the terrain. See Figure 3.1 for an example. Triangles are created such that all normals point towards the sky.

Scene configuration is passed in as a `YAML` file, see Appendix A for a full list of available parameters. The resolution of provided DTED files may be

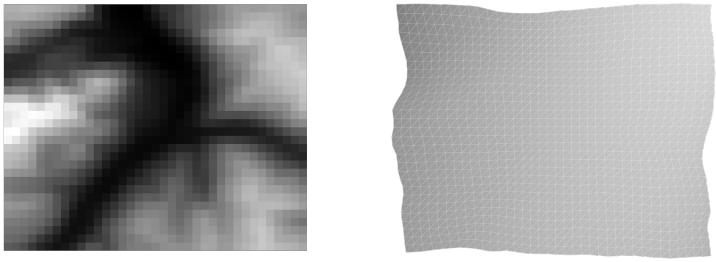


Figure 3.1: Creation of terrain mesh from DTED input. Left shows the height in grayscale, right shows the mesh created from this data.

low, leading to aliasing. To solve this issue, the user can provide the parameter `terrain.subdivisions`, which determines the per-axis subdivision for all tiles. The heights at the subdivision points are approximated using bilinear interpolation. Each tile is further split into two triangles. For example, if the parameter is set to two, each terrain tile will be split into four smaller tiles, each of which will be divided into two triangles. The effect of this parameter is shown in Figure 3.2.

Atmospheric profiles are provided in tabulated plaintext files, with one line per atmospheric layer. See Listing 3.1 for an example. The `tools` directory of the repository contains several scripts to generate atmospheric profiles.

For each atmospheric layer, two triangles are created, forming a rectangle at the specified height with width and depth of the terrain. The associated temperature, wind, pressure and relative humidity are stored in a data structure. The transmissive BSDF of atmospheric layers also stores the wind and sound speed above and below the layer (with respect to the normal of the triangle) for calculation of the refraction.

The sound power level of the sound source can either be defined over all frequency bands, using the `sound.powerLevel` parameter, or by specifying the path to a tabulated file containing one sound power level per frequency band, using the `sound.data` parameter. See the `tools` directory for a short script to generate such a file, and the `examples` directory for an example scene using a sound file.

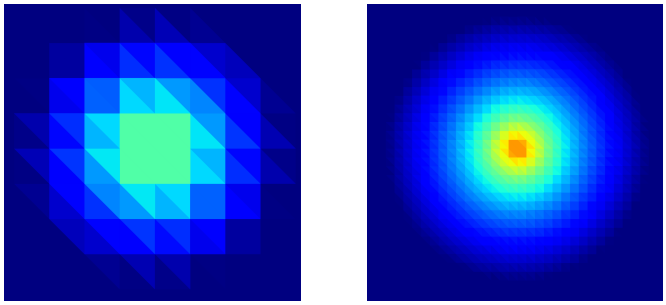


Figure 3.2: Comparison of different tile subdivision factors $s = 1$ in (a) and $s = 4$ in (b), using identical scene configuration and heightmap.

```
1    Elevation[m] Temp[K] Wind[x] Wind[z] Pressure[Pa] Rel.Humidity[%]
2    -2000        301.2   0       0       127780.0      40
3    -1500        297.9   0       0       120700.0      40
4    ...
```

Listing 3.1: Excerpt of the ISO Standard Atmosphere [38] in tabulated plaintext.

The height of the sound source can be defined using the `plane.position.heightAboveGround` parameter, and must be given in above ground level (AGL). To get the height of a sound source based on the created terrain, we can use the formula for barycentric coordinates adjusted for the xz-plane. For this, we need to calculate the area of a triangle in 2D, which we can do as follows,

$$A'(\triangle \mathbf{abc}) = \frac{1}{2} \left| \mathbf{a}_x(\mathbf{b}_z - \mathbf{c}_z) + \mathbf{b}_x(\mathbf{c}_z - \mathbf{a}_z) + \mathbf{c}_x(\mathbf{a}_z - \mathbf{b}_z) \right|.$$

Using this definition we can calculate (u, v, w) directly from Equation 2.12, and thus get the y-coordinate of the sound source as

$$\mathbf{p}_y = \left(u \cdot \mathbf{a}_y + v \cdot \mathbf{b}_y + w \cdot \mathbf{c}_y\right) + h,$$

where h is the AGL altitude. This is also called barycentric interpolation.

The materials described in the next section are either created globally using the parameter `terrain.material.reflect`, which determines whether reflections are enabled at all, and the parameter `terrain.material.diffuse`, which determines the fraction of sound which is reflected diffusely, or by specifying the path to a texture file in the parameter `terrain.texture`.

A texture file contains the following information for each terrain tile: If the material is absorptive, the fraction of sound reflected specularly, and, for each frequency band, the amount of sound that is reflected. This information is stored in binary. The `tools` directory in the repository contains a simple script for creation of trivial textures, and the `examples` directory contains an example scene using a texture file.

3.1.2 Materials

For every type of material, we need to define how we sample outgoing directions ω_o and the BSDF $f(\mathbf{p}, \omega_i \to \omega_o)$ that describes the amount of incident radiance which is reflected towards ω_o. Before we can do so, we need to take a quick look at a probability distribution for specular and transmissive materials.

DIRAC DELTA FUNCTION For materials that map every ingoing direction to a single outgoing direction, we require a probability density function with $q(\omega_o) = 1$ for the outgoing direction, and $q(\omega) = 0$ for every $\omega \neq \omega_o$. For

this, we use the Dirac delta function, which is the function which is zero everywhere except at zero, and whose integral over $\mathbb{R}$ is one, i.e.

$$\delta(x) = 0 \text{ if } x \neq 0, \tag{3.1}$$

$$\int \delta(x)dx = 1. \tag{3.2}$$

This is a probability density function which is 0 for all $x \neq 0$, so we can set $q(\omega) = \delta(\omega_o - \omega)$ for such materials. There is one issue: For $x \to 0$, $\delta(x) \to \infty$. We will later see that the delta function appears also in the BSDF, hence it will cancel out in the rendering equation.

DIFFUSE-SPECULAR Let us first derive the probability densities for diffuse and specular reflections in isolation. We will later see how we can combine the two using multiple importance sampling as introduced in Section 2.2.4.

In the case of a diffuse reflection, we can either sample the hemisphere above a point uniformly, or sample from a cosine-weighted distribution, which corresponds to BSDF sampling. We start with the uniform distribution.

Each direction on the hemisphere should have the same likelihood. A naive approach would be to sample θ uniformly from $[0, \pi/2]$ and ϕ uniformly from $[0, 2\pi]$. The problem with this lies in how the area on the surface of a sphere changes with spherical coordinates. As we have seen in Section 2.1.2, the infinitesimal area element $\mathrm{d}A$ on the sphere with radius r, dependent on the spherical coordinates, is given by

$$\mathrm{d}A = r^2 \sin\theta \,\mathrm{d}\theta \,\mathrm{d}\phi\,.$$

Sampling θ uniformly from $[0, \pi/2]$ would mean every interval of θ has the same probability. However, since $\mathrm{d}A$ contains a factor $\sin\theta$, the area represented by a small interval of θ changes across different θ values. Using the naive approach would lead to more samples around the poles. Thus, we derive the correct density function as follows.

For a valid probability density function, we must have

$$\int_{H^2} p(\omega)\,\mathrm{d}\omega = 1.$$

We are looking for a uniform distribution, so $p(\omega) = c$ for some c. It follows,

$$\int_{H^2} p(\omega)\,\mathrm{d}\omega = c \int_{H^2} 1\,\mathrm{d}\omega = 2\pi.$$

Thus, we have $p(\omega) = 1/2\pi$. To use spherical coordinates, we convert using the result from Section 2.2.3, with $p(\theta, \phi) = \sin\theta/2\pi$. We can get the marginal density for θ as follows,

$$p(\theta) = \int_0^{2\pi} p(\theta, \phi)\,\mathrm{d}\phi = \int_0^{2\pi} \frac{\sin\theta}{2\pi}\,\mathrm{d}\phi = \sin\theta.$$

And the conditional density for ϕ,

$$p(\theta \mid \phi) = \frac{p(\theta, \phi)}{p(\theta)} = \frac{1}{2\pi}.$$

We finally apply the inversion method to sample θ and ϕ given two uniform independent random samples $\xi_1, \xi_2 \in [0, 1]$. The CDFs are as follows,

$$P(\theta) = \int_0^\theta \sin\theta' \, \mathrm{d}\theta' = 1 - \cos\theta,$$
$$P(\phi \mid \theta) = \int_0^\phi (1/2\pi) \, \mathrm{d}\phi' = \frac{\phi}{2\pi}.$$

Finding the inverse functions yields

$$\theta = \cos^{-1} \xi_1, \tag{3.3}$$
$$\phi = 2\pi\xi_2. \tag{3.4}$$

Using Cartesian coordinates directly yields

$$x = \sin\theta\cos\phi = \sqrt{1 - \xi_1^2}\cos(2\pi\xi_2), \tag{3.5}$$
$$y = \cos\theta = \xi_1, \tag{3.6}$$
$$z = \sin\theta\sin\phi = \sqrt{1 - \xi_1^2}\sin(2\pi\xi_2). \tag{3.7}$$

For cosine-weighted sampling of the hemisphere, we want the probability density to be proportional to $\cos\theta$,

$$\begin{aligned}\int_{H^2} p(\omega) \, \mathrm{d}\omega &= \int_0^{2\pi} \int_0^{\pi/2} c\cos\theta\sin\theta \, \mathrm{d}\theta \, \mathrm{d}\phi \\ &= c2\pi \int_0^{\pi/2} \cos\theta\sin\theta \, \mathrm{d}\theta \, \mathrm{d}\phi \\ &= c\pi.\end{aligned}$$

Thus, $p(\theta, \phi) = (\cos\theta\sin\theta)/\pi$ and $p(\omega) = \cos\theta/\pi$. Instead of using marginal and conditional densities as for the uniform case, we can use Malley's method [19] to get cosine-distributed points. Malley's method works by projecting uniformly distributed points on the unit disk to the hemisphere, with the result following a cosine-weighted distribution.

First, we must get uniformly distributed points on the unit disk. The area of the unit disk is π, so the probability of any point must be $1/\pi$. In polar coordinates, we have $p(r, \phi) = r/\pi$, see Section 2.2.3. Thus, the marginal and conditional densities are given as follows,

$$p(r) = \int_0^{2\pi} p(r, \phi) \, \mathrm{d}\theta = 2r,$$
$$p(\phi \mid r) = \frac{p(r, \phi)}{p(r)} = \frac{1}{2\pi}.$$

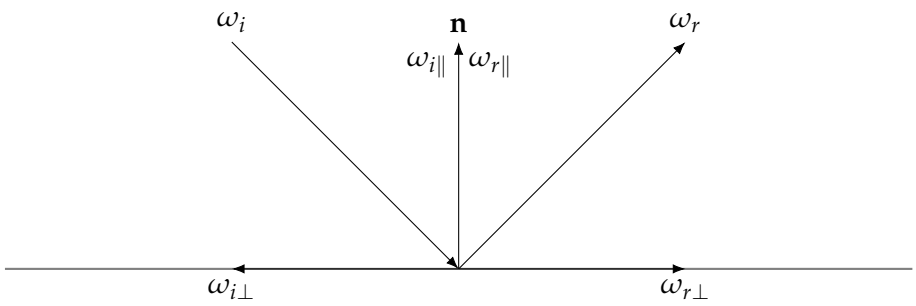


Figure 3.3: Reflection of a ray ω_i towards ω_r on a surface with normal $\mathbf{n}$.

After integration to get the CDFs and subsequent inversion, we are left with

$$r = \sqrt{\xi_1}, \tag{3.8}$$
$$\phi = 2\pi\xi_2. \tag{3.9}$$

Given a two-dimensional Cartesian point $\mathbf{p}$ generated in this way, we project it onto the hemisphere by setting $\mathbf{p}_y = \sqrt{1 - \mathbf{p}_x^2 - \mathbf{p}_z^2}$.

The specular reflection case is much simpler. First, we need to derive the reflected direction ω_r, shown in Figure 3.3. We can derive it as follows,

$$\begin{aligned}\omega_r &= \omega_{r\perp} + \omega_{r\parallel} \\ &= -(-\omega_{i\perp}) + (-\omega_{i\parallel}) \\ &= \omega_{i\perp} - \omega_{i\parallel} \\ &= (\omega_i - (\mathbf{n}\cdot\omega_i)\mathbf{n}) - (\mathbf{n}\cdot\omega_i)\mathbf{n} \\ &= \omega_i - 2\,(\omega_i\cdot\mathbf{n})\,\mathbf{n}.\end{aligned} \tag{3.10}$$

The probability density function is the Dirac delta function,

$$p(\omega) = \delta(\omega - \omega_r). \tag{3.11}$$

To combine diffuse and specular reflections in a single material, we use multiple importance sampling. The fraction of incident radiance that is reflected specularly is given by a user-defined $s \in [0,1]$, the diffusely reflected fraction of incident radiance follows as $d = 1 - s$. We sample the type of reflection with probabilities $q_s = s$ and $q_d = 1 - s$. Let $p_d(\omega)$ and $p_s(\omega)$ be the probability density functions for diffuse and specular reflections as we derived them in this section. The balance heuristic gives us

$$\begin{aligned}w_s(\omega) &= \frac{p_s(\omega)}{p_s(\omega) + p_d(\omega)}, \\ w_d(\omega) &= \frac{p_d(\omega)}{p_d(\omega) + p_s(\omega)} = \frac{p_d(\omega)}{p_d(\omega)} = 1.\end{aligned}$$

Note that in the case of a diffuse reflection, $p_s(\omega) = 0$, since it is not possible to sample the direction ω_r. The final probability for the specular case is given by

$$p(\omega) = \frac{q_s \cdot p_s(\omega)}{w_s(\omega)}, \tag{3.12}$$

and analogously for the diffuse case. Since we divide by $p(\omega)$ in our path tracer, this yields the correct estimate

$$\frac{w_s(\omega)}{q_s}\frac{F}{p_s(\omega)}.$$

The amount of sound reflected depends on whether we have a specular or diffuse reflection. In the former case, s of the incident sound is propagated along the outgoing ray ω_r. In the latter case, $(1-s)/\pi$ is reflected. The BSDF is therefore

$$f(\mathbf{p}, \omega_o, \omega_i) = \begin{cases} d/\pi & \text{if } \omega_o \neq \omega_r, \\ \delta(\omega_o - \omega_r) \cdot s/(\cos\theta_r) & \text{else,} \end{cases} \tag{3.13}$$

where θ_r is the angle between surface normal and specularly reflected direction ω_r. The $\cos\theta_r$ factor and Dirac delta function cancel out in the rendering equation.

We show that this is a valid BSDF. Let the functions g and h be defined as

$$g(\omega_o) = \begin{cases} d/\pi & \text{if } \omega_o \neq \omega_r, \\ 0 & \text{else,} \end{cases}$$

and $h(\omega_o) = \delta(\omega_o - \omega_r) \cdot s/(\cos\theta_r)$. We have $f(\mathbf{p}, \omega_o, \omega_i) = g(\omega_o) + h(\omega_o)$ for any ω_i and $\mathbf{p}$. It follows that

$$\begin{aligned} \int_{H^2(\mathbf{n})} f(\mathbf{p}, \omega_o, \omega_i) \cos\theta_o \, \mathrm{d}\omega_o = &\int_{H^2(\mathbf{n})\setminus\{\omega_r\}} \frac{d}{\pi} \cos\theta_o \, \mathrm{d}\omega_o \\ &+ \int_{H^2(\mathbf{n})} \delta(\omega_o - \omega_r) \frac{s}{\cos\theta_r} \cos\theta_o \, \mathrm{d}\omega_o \\ =&\frac{d}{\pi} \int_{H^2(\mathbf{n})\setminus\{\omega_r\}} \cos\theta_o \, \mathrm{d}\omega_o \\ &+ \frac{s}{\cos\theta_r} \int_{H^2(\mathbf{n})} \delta(\omega_o - \omega_r) \cdot \cos\theta_o \, \mathrm{d}\omega_o \\ =&(1-s) + \frac{s}{\cos\theta_r} \cos\theta_r \\ =&1. \end{aligned}$$

The second-to-last step holds because

$$\begin{aligned} \int_{H^2(\mathbf{n})\setminus\{\omega_r\}} \cos\theta_o \, \mathrm{d}\omega_o &= \int_0^{2\pi} \int_0^{\pi/2} \sin\theta_o \cos\theta_o \, \mathrm{d}\theta_o \, \mathrm{d}\phi_o \\ &= 2\pi \int_0^{\pi/2} \sin\theta_o \cos\theta_o \, \mathrm{d}\theta_o \\ &= \pi, \end{aligned}$$

and

$$\int \delta(x-y) f(x) dx = f(y).$$

TRANSMISSIVE For the transmissive atmospheric material, we use the adjusted Snell's law to calculate the refracted direction of the ray. From Equation 2.85 follows directly

$$\sin\theta_2 = \frac{c_2 \sin\theta_1}{c_1 + \sin\theta_1(u_1 - u_2)}, \tag{3.14}$$

where θ_1 is the ingoing angle, c_1 is the sound speed on the side of the surface ray is incident and u_1 is the wind speed on the side of the surface the ray is incident, adjusted for the plane the ray travels on. This is simply the dot product of the unit direction of the ray $\mathbf{s}$ and the wind $\mathbf{w}_1$:

$$u_1 = \mathbf{s} \cdot \mathbf{w}_1.$$

Variables θ_2, u_2 and c_2 are defined analogously on the opposite side of the surface. Using this equation, we get the refracted direction ω_r using the conversion from spherical to Cartesian coordinates,

$$\theta_2 = \operatorname{asin}(\sin\theta_2), \tag{3.15}$$
$$\phi_2' = \operatorname{atan2}(\omega_{iz}, \omega_{ix}), \tag{3.16}$$
$$\phi_2 = \begin{cases} \phi_2' + 2\pi & \text{if } \phi_2' < 0, \\ \phi_2' & \text{otherwise,} \end{cases} \tag{3.17}$$
$$\omega_r := (\sin\theta_2 \cos\phi_2, \cos\theta_2, \sin\theta_2 \sin\phi_2)\,. \tag{3.18}$$

If $\sin\theta_2 > 1$, the ray is instead reflected as derived in Equation 3.10. The probability density function is trivial,

$$p_\omega(\omega) = \delta(\omega - \omega_r). \tag{3.19}$$

The BSDF is the same as for specular materials,

$$f(\mathbf{p}, \omega_o, \omega_i) = \frac{\delta(\omega_o - \omega_r)}{\cos\theta_r}, \tag{3.20}$$

where $\theta_r = \theta_2$ in case of refraction, otherwise it is the angle between surface normal and specularly reflected direction. Note that since we are tracing radiance starting from the sound source, not incident radiance starting from a microphone, we do not need to add a factor that takes care of asymmetry [19].

3.1.3 Path Tracing

The path tracing algorithm is shown in Algorithm 1. We create paths incrementally, starting at the point sound source. Each ray carries the following information: Origin p, direction ω, the cosine of the outgoing angle with respect to the surface at the origin `cosTheta`, the probability of the direction q, radiance L and throughput β for each frequency band.

Algorithm 1 Path tracing algorithm for a single path given a sound source positioned at $\mathbf{s}$.

$\mathbf{p} \leftarrow \mathbf{s}$ ▷ *Source position*
$\omega \sim q_{\texttt{source}}$ ▷ *Sample initial direction*
$q \leftarrow q_{\texttt{source}}(\omega)$ ▷ *Probability of ω*
$L \leftarrow I$ ▷ *Initial radiance given by intensity, see Equation 2.96*
$\beta \leftarrow 1$ ▷ *Initial throughput*
$\texttt{cosTheta} \leftarrow 1$ ▷ *Cosine of direction with respect to normal*

$i \leftarrow 0$
while $i < \texttt{maxDepth}$ **do**
  $\mathbf{p}' \leftarrow \texttt{raycast}(\mathbf{p}, \omega)$
  $\mathbf{n} \leftarrow \texttt{normal}(\mathbf{p}')$ ▷ *Front-facing normal of the surface*

  if $(0,1,0) \cdot \mathbf{n} < 0$ **then**
    break ▷ *Ray hits from below the surface*

  $\xi \sim \mathcal{U}(0,1)$ ▷ *Sample uniformly from $[0,1]$*
  if $\xi < q_{\texttt{rr}}$ **then**
    break ▷ *Russian Roulette termination*
  $w \leftarrow 1/(1 - q_{\texttt{rr}})$ ▷ *Russian Roulette weighting*

  ▷ *Get atmospheric attenuation between origin and intersection point.* ◁
  $m \leftarrow \texttt{atmosphericAttenuationAt}(\frac{1}{2}(\mathbf{p} + \mathbf{p}'))$
  $d \leftarrow \|\mathbf{p} - \mathbf{p}'\|$ ▷ *Calculate Euclidean distance*
  $L \leftarrow L \cdot 10^{-m \cdot d/10}$ ▷ *Apply atmospheric attenuation*

  $\beta \leftarrow \beta \cdot w \cdot \texttt{cosTheta}/q$ ▷ *Update throughput*

  if $\mathbf{p}'$ is on microphone M **then**
    $S(M) \leftarrow S(M) + L \cdot \beta \cdot W_e(M)$ ▷ *Update accumulated radiance*

  $\omega' \sim q_f$ ▷ *Sample according to BSDF f of surface*
  $q \leftarrow q_f(\omega')$ ▷ *Update probability*
  $\texttt{cosTheta} \leftarrow \omega' \cdot \mathbf{n}$ ▷ *Update cosine factor*
  $\beta \leftarrow \beta \cdot f(\mathbf{p}', -\omega \rightarrow \omega')$ ▷ *Apply BSDF to throughput*

  $\mathbf{p} \leftarrow \mathbf{p}'$
  $\omega \leftarrow \omega'$
  $i \leftarrow i + 1$

Initially, the ray origin is the sound source position, the direction is chosen at random according to the selected method and distribution, i.e. either uniformly sampled from the hemisphere below or sphere around the sound source, or sampled from the hemisphere below using cosine-weighted sampling. The probability of the origin is $q_A(\mathbf{p_0}) = 1$ (since there is only a single sound source), thus the throughput is set to one. Since the normal is equal to the direction, the cosine is also set to one, and finally radiance is calculated based on the configured sound power level. If the Doppler effect is enabled, frequencies are shifted according to the velocity of the sound source.

On intersection with a surface, the throughput is updated according to the BSDF of the material of the surface, and the radiance is updated according to atmospheric attenuation, if enabled. If the surface is part of the terrain, the carried radiance is weighted by the throughput and added to the measurement of the surface. If a linear measurement responsivity is chosen, the value is further divided by the cosine of the angle between surface normal and ray direction. Then, a new direction is sampled according to the BSDF, and all parameters are set for the next iteration of the tracing loop.

3.1.4 Output

NoiseTracer outputs one file per frequency band, which contains the coordinates of each vertex of a triangle and the calculated sound intensity level of that triangle. Additionally, only the center of each triangle and the respective sound intensity level are returned.

The OASPL and A-weighted OASPL are returned for triangle centers, to give one indicator over all frequency bands. Also, the total amount of sound power received on all surfaces for the frequency band 16 Hz is returned.

Furthermore, an `OBJ` file is created to make visualization of the created triangle mesh possible. For visualization of the sound intensity levels in a three-dimensional mesh, a Python script using Matplotlib [39] is included in the repository.

3.1.5 Numerics

Since NoiseTracer is built to run on both CPU and GPU, we use 32-bit floating point numbers, as GPU hardware is optimized for floating point operations.

Floating point numbers are a limited approximation of $\mathbb{R}$. An IEEE 754 floating point number consists of a 1-bit sign, 8-bit exponent and a 23-bit mantissa. This limited storage can have a great impact on Monte Carlo simulations, where a lot of floating point operations have to be performed.

For example, the intersection point calculated using Möller-Trumbore (see Section 2.1.2) may be above or below the actual surface. When tracing the next ray, it may self-intersect with the surface it originates from [40]. It is therefore important to choose a t_{min} that is large enough to solve such issues, yet small enough to not miss close surfaces. To make sure rays originating beneath terrain do not contribute to the estimate, we cull any rays which

originate from below a surface by checking the ray direction against the normal of the surface.

Another problem we encountered was in accumulating billions of small numbers. Recall that for each surface in a scene, we store the absorbed sound radiance. The radiance that reaches a surface via a ray is usually pretty small. In one experiment, the values were in the range $[0.0001, 1]$.

Given many samples, the accumulated radiance may become a relatively large number. In one experiment using $n = 2^{34}$ samples, we could see that the radiance at the loudest point, right beneath the sound source, reached around 262179.

Doubling the number of samples roughly doubles the radiance for a surface, if using an unbiased estimator. After a certain point, in our experiment $n = 2^{32}$, this was not the case anymore. While the radiance values at most surfaces doubled, the one that reached around ϕ only increased insignificantly.

This is due to the nature of floating point addition and subtraction. Given two numbers with different exponents, the number with the smaller exponent is brought to the same exponent by shifting the mantissa to the left. In case the exponents differ significantly, this leads to loss of information in the smaller number. And this, in turn, can lead to billions of additions of a small number to a large number effecting almost no change. We thus use double-precision floating point numbers for the accumulated radiance of a surface.

3.2 ACCELERATION

3.2.1 CPU

BOUNDING VOLUME HIERARCHY We implemented the Surface Area Heuristic (SAH) for creation of BVH-trees. Though it takes more time to compute the tree, traversals are generally faster than in other methods. Since the goal of NoiseTracer is interactive use with a static scene but changing source location, we choose the method which is more efficient for tree traversal.

SPECIAL ATMOSPHERE TREATMENT If a sound source is high above the terrain, a ray must first traverse multiple atmospheric layers before it may intersect with any geometry. We can improve performance by making use of this fact.

When initializing a scene, we remember the lowest atmospheric layer above any terrain. If the origin of a ray is above this layer, we only need to check other atmospheric layers for intersection. If, on the other, the origin of the ray is on or below this layer, we check all triangles for intersection. Note that this also includes atmospheric layers, as it may be the case that there are atmospheric layers between the lowest and highest point of terrain.

PARALLELIZATION The algorithm introduced in Algorithm 1 is easily parallelizable: Each thread can trace a path independently. Only when updating the accumulated radiance must care be taken. If two threads trace paths that

hit the same microphone at the same time, one of the contributions may be lost. This problem is solved by using atomic operations.

OpenMP 3.0 [41] was used to parallelize computation on the CPU. OpenMP provides the `omp critical` pragma to ensure data is manipulated atomically. To ensure maximum thread occupancy, we use the `omp task` pragma to enable tasking. This is more efficient than splitting the samples among threads equally, as this could lead to a thread sleeping due to rays hitting absorptive materials or hitting no surfaces at all.

3.2.2 GPU

One of the goals of this thesis was near real-time capability. Although acceleration structures and parallelization improve performance, they can not compete with specialized hardware. Not only are GPUs optimized for parallel computations, modern GPUs have built-in hardware accelerators for building and traversing BVHs as well as for testing ray-triangle and ray-AABB intersection.

In this thesis, we use OptiX [42] and the CUDA framework to program Nvidia GPUs. A good introduction to the CUDA language and the intricacies of programming GPUs can be found in [43].

RANDOM NUMBERS CUDA provides a random number generation library called cuRAND. The problem with this number generator is that it has an expensive setup phase. Running it once per launch would lead to bad rendering performance.[1]

An alternative is to use an efficient random number sequence generator, such as a linear congruential generator (LCG). We still need a good initial seed that is uncorrelated to those of other samples. For this, we can generate random numbers on the CPU and move them to GPU memory before rendering, requiring only one expensive initialization, and yielding better random values than just using LCG. We use Mersenne Twister [44] to generate seeds.

Generating seeds this way puts a limit on the number of samples, depending on the available GPU memory. For example, if the available memory is 8 GB, and each random value is a 32-bit unsigned integer, we are limited to 2^{31} samples (not taking into account the memory required for storing scene geometry and sound data). Since this is more than the samples supported per OptiX launch anyway, we chose this approach. To convert unsigned integers generated by LCG to floating point numbers in the range of $[0, 1)$, we simply divide by the maximum number provided by the LCG.

LIMITATIONS One of the biggest limitations of CUDA is that memory cannot be allocated dynamically. All memory has to be allocated before execution of code on the GPU. Also, since GPUs are optimized for 32-bit numbers, care had to be taken to avoid numbers higher than 2^{32}. Only for the radical inverse function used in Halton sampling were 64-bit unsigned integers used, so that the overall number of samples may exceed 2^{32}, since using 32-

1 For example, see https://ingowald.blog/2018/11/21/rtow-in-optix-fun-with-curand/.

bit unsigned integers would lead to overflows, yielding duplicate numbers of the sequence, defeating the purpose of the Halton sequence.

OptiX has additional limitations that influenced development. The number of rays that may be traced in a single launch is 2^{30}. For a higher number of rays, multiple OptiX launches have to be carried out.

Additionally, the depth of recursion in OptiX is limited to 31 to ensure small stack sizes. This is the reason why we chose to implement a path tracer, as we wanted to support dense atmospheric layers, where a ray may travel through a high number of atmospheric layers before hitting a surface. For multi-sample Monte Carlo estimators, repeated recursive calls would need to be carried out to trace new rays, quickly hitting the limit of 31. Hence, an iterative approach is used, as shown in Algorithm 1.

3.3 TESTING AND VALIDATION

Three key aspects of NoiseTracer have to be validated: First, that the code works as expected. To this end, both unit tests, i.e. testing classes and functions in isolation, as well as functional tests, i.e. running example scenes and validating the results, were used. Automatic testing is performed on every commit to make sure changes don't break anything.

The second important aspect is the validity of the output as a physical simulation of a real-world phenomenon. Since these validation tests require simulations with a high number of samples, and hence a long computation time, it usually does not make sense to include them in the functional test suite, as they should not be executed on every commit. These validation tests can be found in the `validation` directory of the repository, and are discussed in Section 4.2.

Lastly, it is important to validate the effectiveness of variance reduction strategies introduced in Section 2.2.4. These experiments can also be found in the `validation` directory, and are discussed in Section 4.3.

4 ANALYSIS

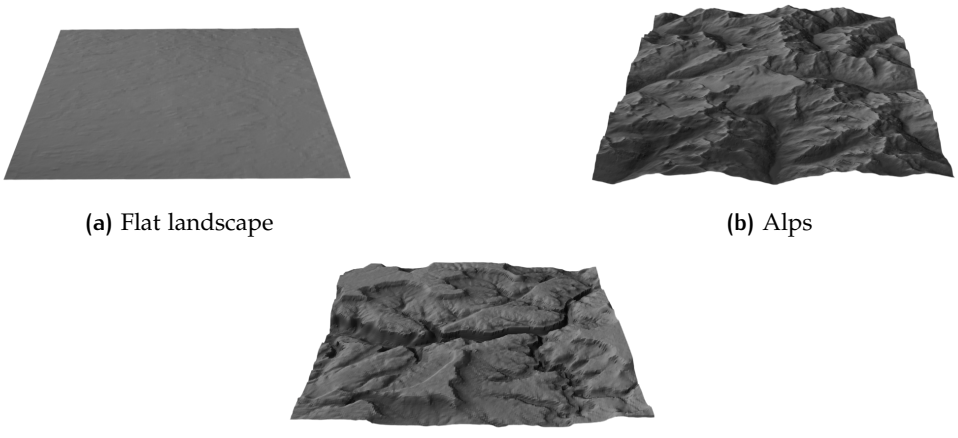
We start this chapter with an introduction of the test scenes that will be considered for our analysis in Section 4.1. We continue with validating the physical phenomena in Section 4.2, before analyizing the effectiveness of variance reduction strategies in Section 4.3. Finally, we discuss the performance of NoiseTracer in Section 4.4.

4.1 SCENES

We consider three types of terrain for our analysis: Flat landscapes, mountainscapes and canyons. All elevation data is based on the Shuttle Radar Topography Mission [45], improved in [46] by filling missing data from various sources, or, if no data was available, using spline interpolation. The data with a resolution of 3 arcseconds was downloaded using the SRTM Tile Grabber developed by Derek Watkins, which can be found at `https://dwtkns.com/srtm/`. For the sake of this analysis, we assume that all tiles are of size 90 m × 90 m. Note that this is not accurate, as the length of an arcsecond in latitude is not constant, see Equation 2.23.

Representative of flat landscapes, we use an area in Lower Saxony centered around 53.195° latitude and 10.269° longitude near the Lüneburg Heath, close to the hometown of the author. As can be seen in Figure 4.1a, the terrain has only small changes in elevation.

For the mountainscape, we chose an area in the Alps, centered around 46.829° latitude and 10.742° longitude. This is shown in Figure 4.1b.



(a) Flat landscape

(b) Alps

(c) Grand Canyon

Figure 4.1: Meshes of the three types of terrains considered in this analysis. Images were rendered using Blender 4.1.

For scenes representing a canyon, we chose an area in the Grand Canyon, centered around 36.397° latitude and −112.586° longitude, visible in Figure 4.1c.

If not otherwise specified, the ISO standard atmosphere [38] was used. Two test scenes were created for each terrain, one with a sound source high above ground, and one with a sound source close to ground. These are denoted `flathigh`, `flatlow`, `alpshigh`, `alpslow`, `canyonhigh` and `canyonlow`. The scenes were chosen to demonstrate NoiseTracer's broad applicability and to examine whether the geometry and source altitude affect the effectiveness of the variance reduction strategies. Additionally, to compare to methods that do not support terrains with elevation, a completely flattened scene is created.

4.2 PHYSICAL ACCURACY

DISTANCE ATTENUATION We can easily validate that NoiseTracer follows the inverse-square law, see Equation 2.71, a significant property of sound propagation, by disabling all atmospheric effects and considering a flattened terrain.

In Figure 4.2, the analytical results are compared against a simulation with $n = 2^{32}$ samples using a flat surface with dimensions 21 780 m × 19 350 m and a sound source located at $x = 10\,891$ m, $z = 9675$ m with an altitude of 10 000 m AGL. Stratified sampling on the hemisphere below the source was used. In the render shown, the per-triangle error of the simulation is $|\epsilon| < 0.05$ dB. We have repeated this experiment for $m = 20$ runs and found the mean absolute error over all triangles and all runs to be $\mu = 0.004$ dB, the standard deviation of the absolute error $\sigma = 0.003$ dB, with absolute errors in the range of 0 dB and 0.05 dB, showing an overall high accuracy.

We also compare our method to the one by Heath and McAninch [11], see Figure 4.3. Unfortunately, no data is available to do a precise comparison. A similar color map is used to allow some degree of comparison, and shows very similar noise levels across the whole plane.

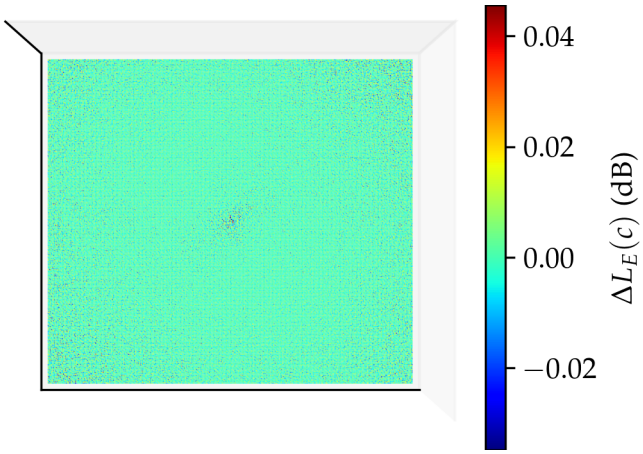
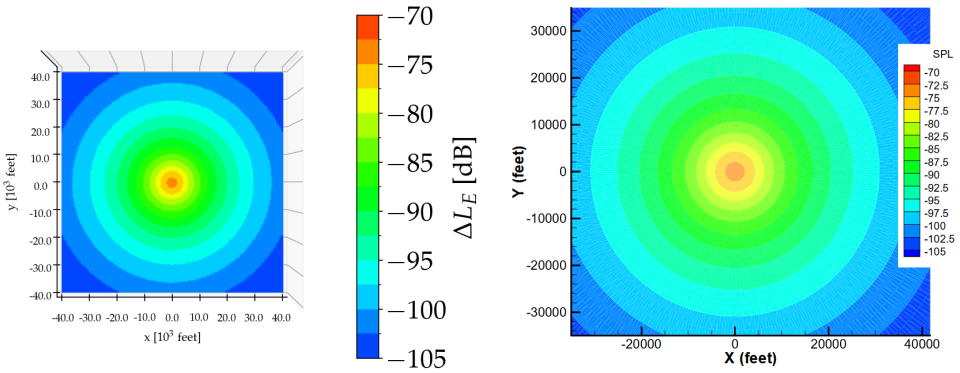
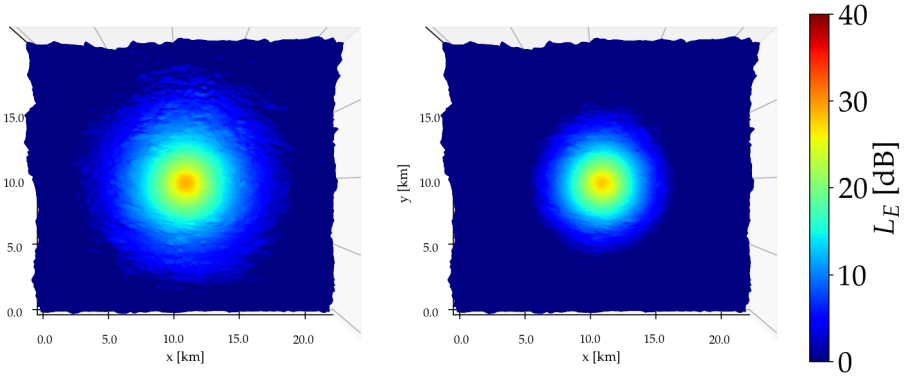


Figure 4.2: Differences in sound level between results of analytical calculation and a simulation using $n = 2^{32}$ samples.



(a) NoiseTracer **(b)** Heath and McAninch [11], Figure 5

Figure 4.3: Comparison of our method to a method developed by Heath and McAninch [11], with a sound source at 5000 ft and disabled atmospheric effects. The simulation was done with $n = 2^{30}$ samples.



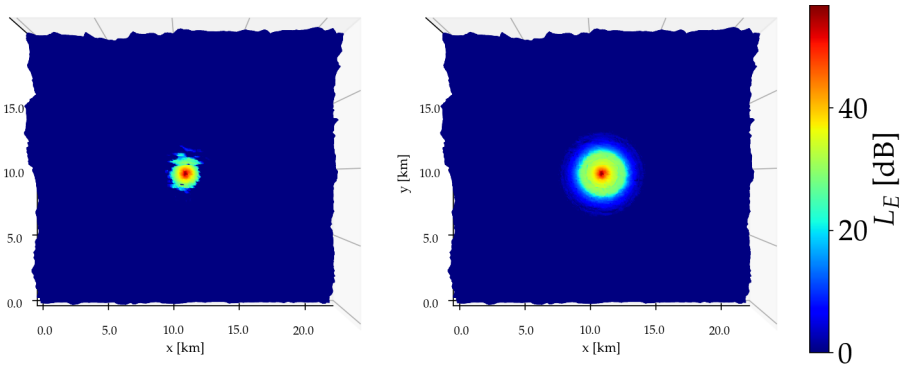
(a) $f_c = 16\,\text{Hz}$ (b) $f_c = 1600\,\text{Hz}$

Figure 4.4: Effect of atmospheric attenuation on frequency bands 16 Hz and 1600 Hz.

ATMOSPHERIC ATTENUATION Atmospheric attenuation as outlined in Section 2.3.3 applies mostly to high frequencies. In Figure 4.4, a simulation using the flat terrain introduced in Figure 4.1a with $n = 2^{30}$ samples was performed. The ISO standard atmosphere with a constant humidity of 40 % was used, with an atmospheric layer every $\Delta_y = 500\,\text{m}$. The sound source was located in the center with an altitude of 1000 m AGL and a sound power of $L_\phi = 100\,\text{dB}$. Comparing the frequency bands 16 Hz and 1600 Hz shows how low-frequency sound reaches about two kilometers farther.

TEMPERATURE GRADIENT In the ISO standard atmosphere, the temperature decreases with altitude. This is a typical temperature profile during the day. At night, however, air near the ground cools off faster than air higher up in the atmosphere, leading to a phenomenon called temperature inversion.

Since sound speed is dependent on temperature (see Equation 2.82), this also affects sound propagation: During a temperature inversion, sound rays are bent towards the ground. This can be observed in Figure 4.5, which shows a render using the terrain shown in Figure 4.1a. In this experiment, a sound source was placed at a height of 100 m AGL with a power of $L_\phi = 110\,\text{dB}$. The atmospheric profile was set such that $T_{\text{day}}(0\,\text{m}) = 295\,\text{K}$,



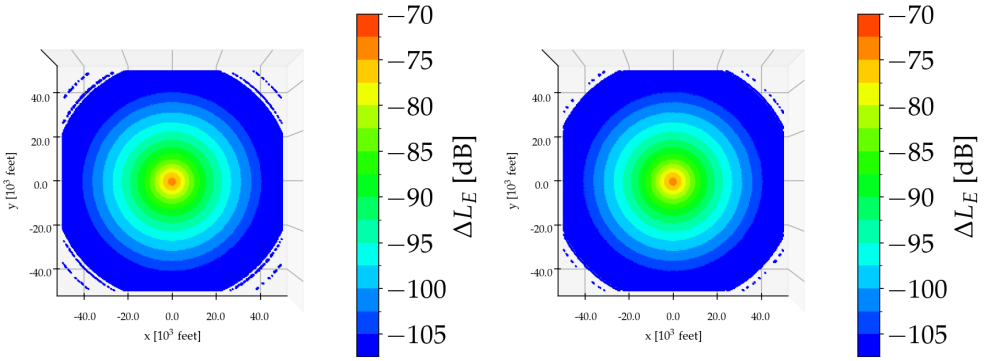
(a) day (b) night

Figure 4.5: Comparison of day and night atmospheric profile in a simulation with $n = 2^{30}$ samples.

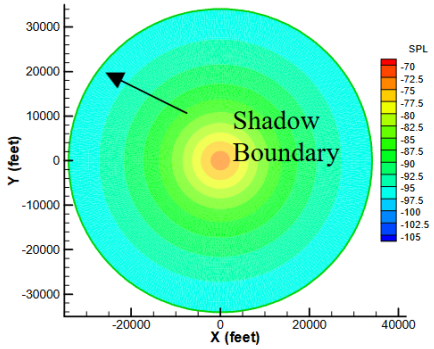
$T_{\text{day}}(200\,\text{m}) = 275\,\text{K}$, $T_{\text{night}}(0\,\text{m}) = 275\,\text{K}$ and $T_{\text{night}}(200\,\text{m}) = 295\,\text{K}$. For both day and night profiles, pressure and humidity were set to a constant value. Temperature was interpolated linearly in $\Delta_y = 0.5\,\text{m}$ increments. Note the stronger concentration of sound during the day, as more rays are refracted into the atmosphere.

See also Figure 4.6 for a comparison to the method by Heath and McAninch. Note that in our method, sound reaches points further away from the sound source. Using NoiseTracer, sound reaches beyond 40 000 ft, whereas in the method by Heath and McAninch, the shadow boundary is at about 34 000 ft away from the source. This shows the reduced accuracy of our method due to the linear approximation of ray propagation, as shown in Figure 2.14a. An increase in the detail of the atmosphere leads to less aliasing around the edges of the shadow boundary.

DOPPLER EFFECT The application of the Doppler effect is shown in Figure 4.7. Here, a simulation with a sound source at 5000 m AGL with a power of $L_\phi = 120\,\text{dB}$ over all frequency bands was performed, with a source velocity of $v_x = 80\,\text{m/s}$, using the terrain introduced in Figure 4.1a. The simulation was done with $n = 2^{30}$ samples with atmosphere disabled to focus on the Doppler effect. The experiment shows that sound in the 16 Hz frequency band is shifted towards higher frequencies in the direction of travel of the object.

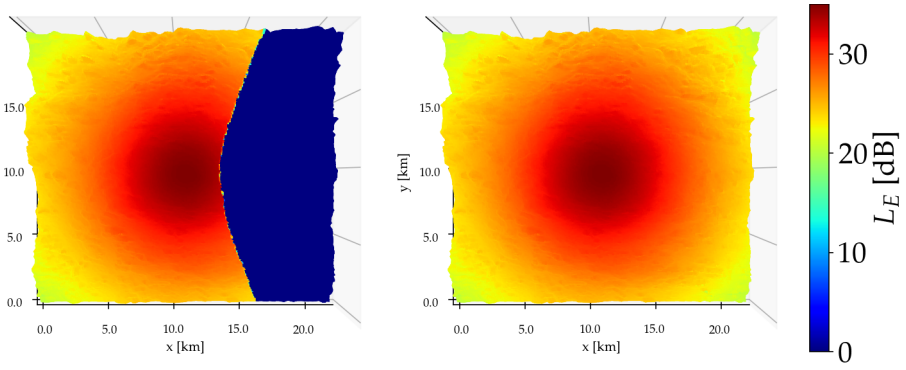


(a) NoiseTracer, $\Delta_y = 10\,\text{ft}$ **(b)** NoiseTracer, $\Delta_y = 3\,\text{ft}$



(c) Heath and McAninch [11], Figure 6

Figure 4.6: Comparison of our method to a method developed by Heath and McAninch [11], with a sound source at 5000 ft and a temperature gradient of $\beta = 0.003\,54\,°\text{F/ft}$. The atmosphere is interpolated linearly with $\Delta_y = 3\,\text{ft}$ and $\Delta_y = 1\,\text{ft}$. Simulations were done with $n = 2^{30}$ samples.



(a) $f_c = 16\,\text{Hz}$ (b) $f_c = 20\,\text{Hz}$

Figure 4.7: Example of the Doppler effect at (a) $f_c = 16\,\text{Hz}$ and (b) $f_c = 20\,\text{Hz}$.

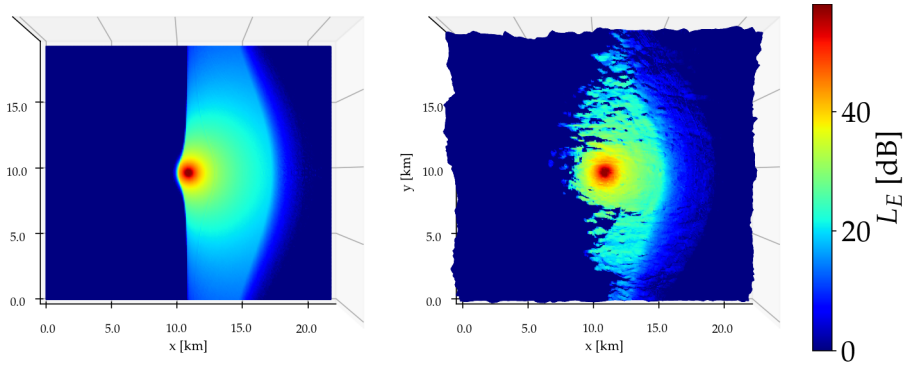
WIND Strong winds can lead to an acoustic shadow in the upwind area. This effect has been reproduced in Figure 4.8. For this experiment, the wind profile shown in Figure 2.15 was used. The sound source was positioned at $z = 105\,\text{m}$ AGL with a sound power of $L_\phi = 120\,\text{dB}$. To focus on the influence of a wind gradient, atmospheric attenuation was disabled, and temperature, humidity and pressure set to a constant value. The atmosphere was divided into $\Delta_y = 1\,\text{m}$ increments. The simulation was done with $n = 2^{33}$ samples.

An acoustic shadow towards the left of terrain is visible. Note that in reality, wind speeds are not constant. Acoustic shadows as depicted in this example are a rare phenomenon. Also note that when using the landscape shown in Figure 4.1a, the low altitude of the sound source leads to acoustic shadows due to elevations. To isolate the effect of wind, we also show a completely flattened terrain without any elevations. The flat landscape render also shows more sound reaching in the upwind area. Since rays towards the upwind area are bent upwards, the area that they may intersect in a completely flattened terrain is small. If there are elevations, however, the refracted paths may intersect terrain.

Figure 4.9 shows a comparison to APHRODITE [13]. One key difference between APHRODITE and NoiseTracer is the way microphones are modeled: In our method, microphones are surfaces. If we assume that the amount of energy reaching a surface depends on the angle between surface normal and incoming direction, i.e. we have a Lambertian responsivity, rays that are refracted upwards intersect geometry at a low angle and therefore have a low sound energy contribution, as shown in Figure 4.9a. In APHRODITE, however, microphones are just points in space. This can be emulated in NoiseTracer using a linear microphone responsivity, as shown in Figure 4.9b, where sound level loss similar to that of APHRODITE and a method by Olsman and Lummer [35] in Figure 4.9c can be seen.

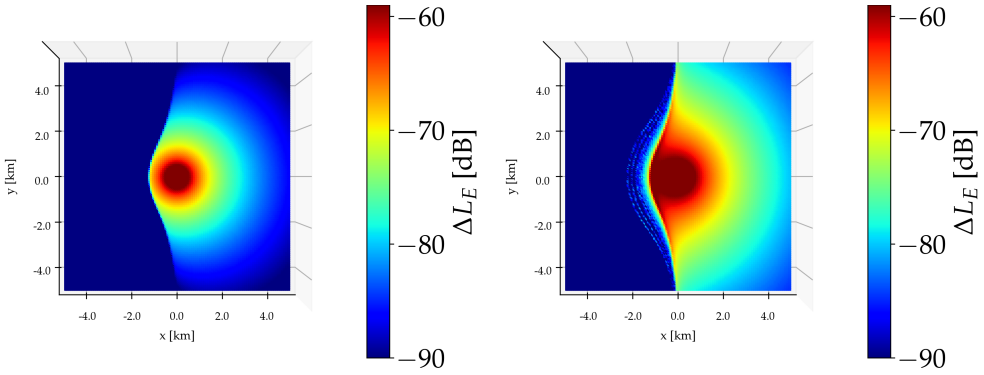
In Figure 4.10, we see similar sound level loss and distribution in NoiseTracer as in the method by Heath and McAninch [11]. Two key differences are noticeable. First, there are differences in the shadow boundary. In Heath and McAninch, the position $x = 0\,\text{ft}$ and $y = 20\,000\,\text{ft}$ is in the acoustic shadow, whereas NoiseTracer predicts sound there. Second, some sound reaches points towards the left of the shadow boundary. This effect is reduced by increasing the detail of the atmosphere.

GROUND REFLECTIONS The effect of ground reflections is especially noticeable in mountainous scenes. We therefore chose the Alps scene shown in Figure 4.1b to showcase this effect. Three renders with $n = 2^{32}$ samples are shown in Figure 4.11. The sound source is positioned in the center and has a sound power of $L_\phi = 120\,\text{dB}$ at an altitude of $10\,000\,\text{m}$ AGL. Atmosphere was disabled to focus on reflections. Notice how in the absorptive and specular case, there are areas of the terrain, behind elevations, that sound does not reach. In the diffuse case, sound is reflected towards these areas from elevations further away from the source.



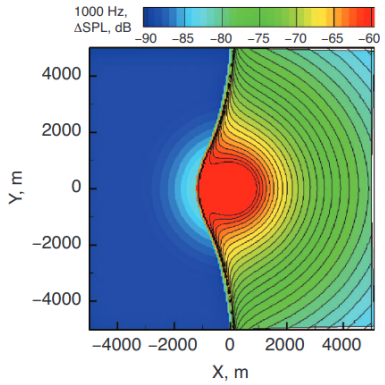
(a) flattened (b) flat landscape

Figure 4.8: Example of the effect of strong winds on sound propagation.



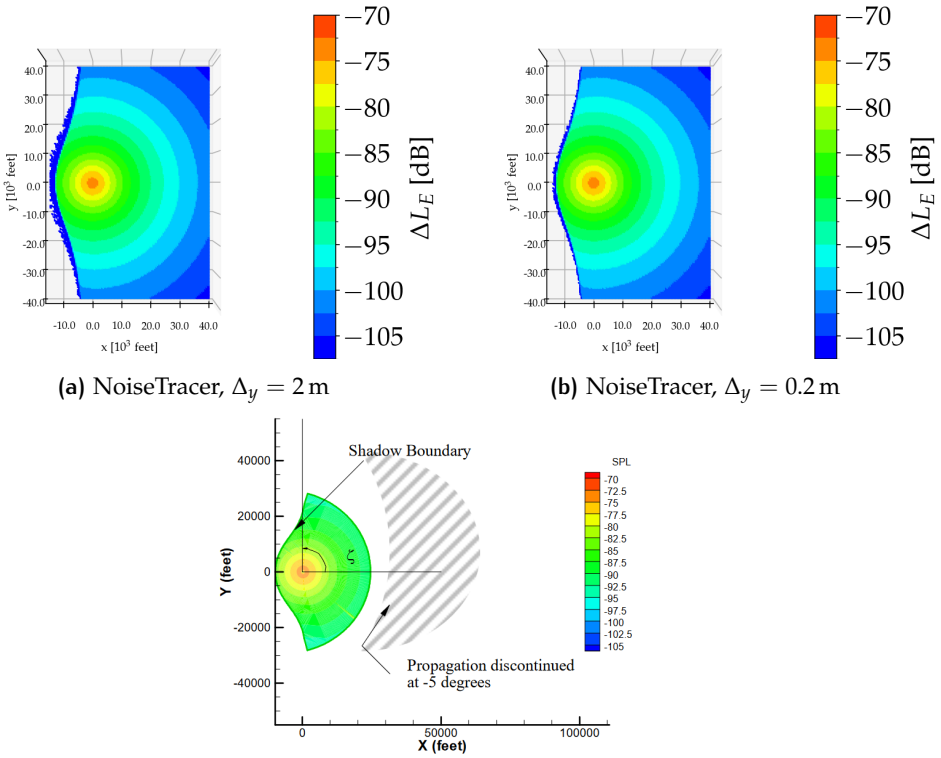
(a) NoiseTracer, with Lambertian responsivity

(b) NoiseTracer, with linear responsivity



(c) APHRODITE [13], Figure 12; colors are from APHRODITE, lines are from Olsman and Lummer [35]

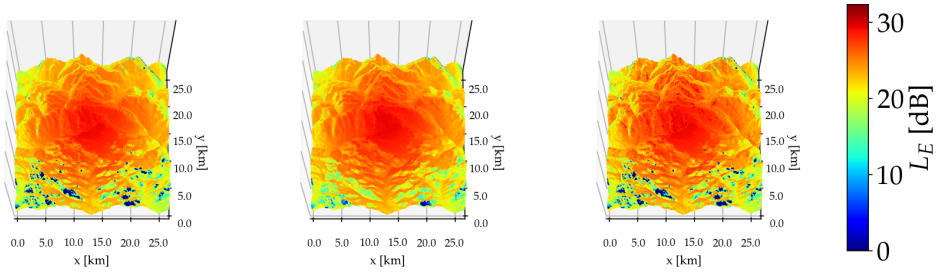
Figure 4.9: Comparison of our method to APHRODITE [13], using the wind profile defined in Figure 10 in [13], interpolated linearly in $\Delta_y = 1\,\mathrm{m}$ increments. Simulations were done with $n = 2^{34}$ samples.



(a) NoiseTracer, $\Delta_y = 2\,\text{m}$ **(b)** NoiseTracer, $\Delta_y = 0.2\,\text{m}$

(c) Heath and McAninch [11], Figure 10

Figure 4.10: Comparison of our method to a method by Heath and McAninch [11], using the wind profile in Figure 9 in [11], interpolated linearly in $\Delta_y = 2\,\text{m}$ and $\Delta_y = 0.2\,\text{m}$ increments, with a sound source at 5000 ft. Simulations were done with $n = 2^{30}$ samples.



(a) absorptive (b) diffuse (c) specular

Figure 4.11: Comparison of (a) absorptive, (b) diffuse and (c) specular ground materials.

4.3 VARIANCE REDUCTION

To measure the effectiveness of the variance reduction strategies introduced in Section 2.2.4, we need a way to compare different renders of a scene. We take the following approach: Similar to comparing the pixel values in optical rendering, we compare the received sound power for each triangle in a scene for a specific frequency band. In effect, for a given scene and frequency band, NoiseTracer returns a function mapping the scene triangles $t \in T$ to sound powers[1] in $\mathbb{R}^+$,

$$\phi : T \to \mathbb{R}^+.$$

Given the solution $\tilde{\phi}$, we calculate the error of an estimate ϕ as follows,

$$\mathrm{MSE}[\phi] = \frac{1}{|T|} \sum_{t \in T} \left(\phi(t) - \tilde{\phi}(t)\right)^2.$$

Since we are interested in the single-run performance of our renderer for the purposes of interactive use, we compare the efficiency of two variance reduction methods by considering their average Monte Carlo efficiency. For multiple runs with results $\overline{\phi} = \phi_1, \ldots, \phi_n$, this value is computed using the average MSE and average time as follows,

$$\begin{aligned}
\mathrm{MSE}[\overline{\phi}] &= \frac{1}{n} \sum_{i=1}^{n} \mathrm{MSE}[\phi_i], \\
\tau[\overline{\phi}] &= \frac{1}{n} \sum_{i=1}^{n} \tau[\phi_i], \\
\epsilon[\overline{\phi}] &= \frac{1}{\mathrm{MSE}[\overline{\phi}]\tau[\overline{\phi}]}.
\end{aligned}$$

We only take into account the 16 Hz frequency band, because we disable atmospheric attenuation for this analysis, as it is completely deterministic. Without atmospheric attenuation, and with a source that has the same sound power over all bands, the sound levels will be identical in every frequency band. All following experiments were done with 10 runs per variance reduction method, sample count and scene. A reference render with $n = 2^{40}$ samples was created for each scene.

RUSSIAN ROULETTE For a given maximum depth of $d_{\max}$ and minimum depth $d_{\min}$ of a path, we terminate rays with probability $q(d) = (d - d_{\min}) \,/\, (d_{\max} - d_{\min})$. Table 4.1 shows the Monte Carlo efficiencies for values $d_{\min} = 30$, $d_{\max} = 100$. For $n = 2^{30}$ samples, the efficiency increases in all but one scene. At a lower sample count of $n = 2^{25}$, `flatlow` and `canyonlow` show a decreased efficiency. This may be due to the lower computational times of $\tau < 1\,\mathrm{s}$, where the additional operations for Russian Roulette, i.e. calculating the termination probability and reweighing the throughput, weigh more heavily. These additional operations are amortized with a higher sample count.

1 The code was changed for this analysis to return sound powers instead of sound intensity levels.

n	Scene	MSE	τ	ϵ
2^{25}				
	`flathigh`	0.999	0.948	1.056
	`flatlow`	1.110	1.006	0.896
	`alpshigh`	1.000	0.958	1.044
	`alpslow`	0.986	1.000	1.013
	`canyonhigh`	0.995	0.953	1.054
	`canyonlow`	1.014	0.992	0.994
2^{30}				
	`flathigh`	1.002	0.904	1.104
	`flatlow`	0.993	0.993	1.014
	`alpshigh`	0.998	0.933	1.074
	`alpslow`	1.014	1.000	0.987
	`canyonhigh`	0.997	0.927	1.082
	`canyonlow`	0.944	0.987	1.074

Table 4.1: Relative mean squared error, rendering time and efficiency due to Russian Roulette. Values relative to the renders of the same scenes with Russian Roulette disabled.

IMPORTANCE SAMPLING Table 4.2 shows the effectiveness of the different importance sampling strategies for $n = 2^{20}$ and $n = 2^{25}$ samples. In the alps and canyon scenes, using hemisphere or cosine-weighted sampling for the source when it has a low altitude leads to a significant increase in the MSE relative to uniform sphere sampling, as in those cases there is terrain above the sound source. At high altitudes, hemisphere sampling yields a lower MSE in these scenes, and cosine-weighted sampling performs even better.

In the flat scene at high altitude, hemisphere sampling reduces the MSE and cosine-weighted sampling improves this further. At a low altitude, hemisphere sampling again yields an improvement in the error, but cosine-weighted sampling leads to an increase of the MSE. This is due to undersampling of directions close to the horizontal.

BSDF sampling is used for diffuse ground materials, using a cosine-weighted distribution on the hemisphere around the normal of the surface. We have barely found any improvements in the mean squared error using BSDF sampling. Only in `flathigh` and `flatlow` could some decrease in MSE be measured. To understand this, it is helpful to look at the normals of the triangles in a terrain. For the flat scene, the mean y-component of all normals is $\mu = 0.999$ with $\sigma = 0.002$, for the Alps scene, we have $\mu = 0.907$ with $\sigma = 0.072$ and finally for the canyon scene we have $\mu = 0.912$ with $\sigma = 0.123$. In all of these terrains, most of the surface normals point towards the sky. Since BSDF sampling leads to more samples towards the normal of any surface, this means that most rays are reflected back into the atmosphere where they cannot contribute to any measurement.

STRATIFIED AND HALTON SAMPLING Both stratified and Halton sampling ensure a good coverage of directions around the sound source, especially when using a low number of samples. Table 4.3 shows the results of renders with $n = 2^{20}$ and $n = 2^{25}$ samples. Both stratified and Halton sampling reduce the MSE significantly, and in all but one case, Halton sampling performs better than stratified sampling. An example render is shown in Figure 4.12.

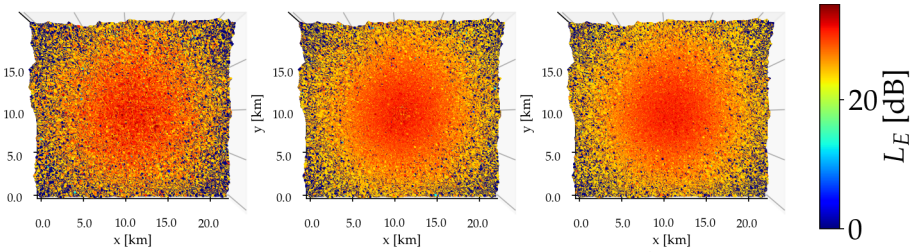
Stratified sampling is especially useful in scenes with winds. Given a low altitude and high wind speeds in the positive x direction, there are only a few paths that lead to sound reaching positions further away from the source in the negative x direction. Using uniform sphere sampling, these paths are undersampled. As can be seen in Figure 4.13, stratification solves this problem. Here, a simulation with $n = 2^{34}$ samples was done, once using uniform sphere sampling, and once using stratified sphere sampling. The terrain was completely flattened, and the wind profile from [13] was used and interpolated linearly in $\Delta_y = 10\,\text{m}$ increments. The sound source was positioned at 301 m AGL with a power of $L_\phi = 150\,\text{dB}$.

n	Scene	MSE (HS)	MSE (CS)	MSE (BSDF)
2^{20}				
	`flathigh`	0.506	0.316	0.999
	`flatlow`	0.456	1.134	0.969
	`alpshigh`	0.511	0.334	1.04
	`alpslow`	20.05	21.18	1.023
	`canyonhigh`	0.518	0.324	1.018
	`canyonlow`	552.0	557.0	1.035
2^{25}				
	`flathigh`	0.671	0.461	1.000
	`flatlow`	0.598	1.643	1.077
	`alpshigh`	0.846	0.642	1.293
	`alpslow`	630.0	632.6	1.332
	`canyonhigh`	1.065	0.795	1.405
	`canyonlow`	17929	17994	2.001

Table 4.2: Relative mean squared error for different importance sampling strategies: Hemisphere sampling (HS), cosine-weighted sampling (CS) and BSDF sampling. All values are relative to renders using uniform sphere sampling for initial directions.

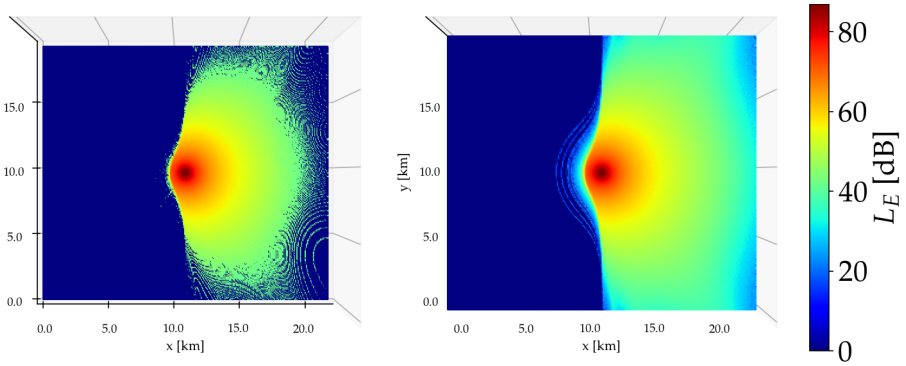
n	Scene	MSE (Stratified)	MSE (Halton)
2^{20}			
	`flathigh`	0.520	0.465
	`flatlow`	0.061	0.056
	`alpshigh`	0.586	0.525
	`alpslow`	0.170	0.152
	`canyonhigh`	0.548	0.492
	`canyonlow`	0.447	0.465
2^{25}			
	`flathigh`	0.246	0.216
	`flatlow`	0.035	0.033
	`alpshigh`	0.433	0.396
	`alpslow`	0.270	0.267
	`canyonhigh`	0.614	0.585
	`canyonlow`	0.885	0.866

Table 4.3: Relative mean squared error for stratified and Halton sampling. All values are relative to those using uniform sampling.



(a) uniform (b) stratified (c) halton

Figure 4.12: Comparison of stratification and Halton sampling for a low sample ($n = 2^{20}$) render of `flathigh`.



(a) uniform (b) stratified

Figure 4.13: Effect of stratification in a scene with wind.

4.4 PERFORMANCE

The impact of the different acceleration methods on rendering time is shown in Table 4.4. CPU performance is improved using a separate treatment of the atmosphere for intersection tests, and even further improved by using an acceleration structure. A 6-core/12-thread AMD Ryzen 5600 with 32 GB DDR4 RAM was used for the CPU benchmarks. GPU performance beats all CPU methods, thanks to hardware-accelerated BVH traversal, intersection tests and massively parallel processing. A Nvidia RTX 3080 with 10 GB GDDR6X was used for GPU benchmarks.

Note that setup time is not included in this analysis. This includes BVH construction in the case of CPU rendering, which, depending on the scene, can take a considerable amount of time. We have measured up to 300 s for the `alpshigh` scene. There is a great potential to optimize this by choosing a more efficient method for BVH-tree construction. Since we rely on the GPU for efficient calculations, we did not implement any such method.

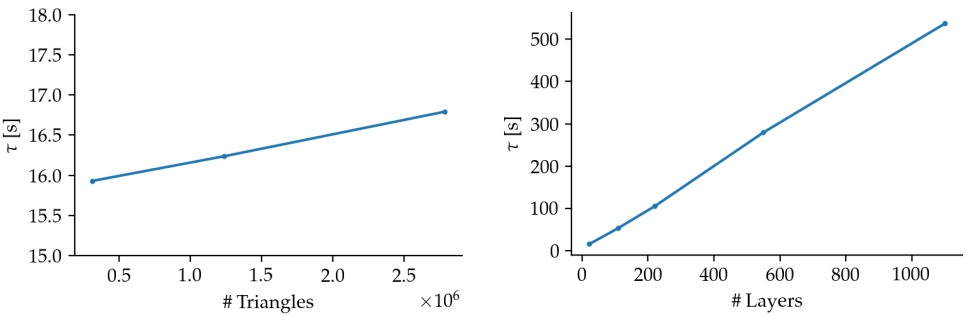
For GPU renders, there is also time required for setting up the OptiX pipeline and BVH-tree creation on the GPU, which is also excluded in Table 4.4. This takes about 300 ms for our example scenes. For a higher number of samples, this additional computational cost is amortized.

Due to hardware-accelerated tree creation, tree traversal and intersection tests, the number of triangles, and therefore the number of microphones, does not have a significant impact on performance when running simulations on the GPU, see Figure 4.14a. The number of atmospheric layers, however, has a bigger impact due to the increased path length, see Figure 4.14b. As we have seen in Section 4.2, specifically in Figure 4.6 and Figure 4.10, the number of layers impacts the accuracy of the simulation, especially at the shadow boundary. Figure 2.14a has shown that for a sound source located at 15 m, a resolution of $\Delta_y = 1\,\mathrm{m}$ lead to loss in accuracy for the shown rays, compared to the result of integration of the ordinary differential equation. Better results were achieved with a high resolution of $\Delta_y = 0.025\,\mathrm{m}$ in Figure 2.14b. Interactive simulations with such a dense atmosphere are not feasible, since the computation would take a considerable time, as shown in Figure 4.14b. Furthermore, the minimum ray intersection distance $t_{\min}$ must be smaller than the distance between atmospheric layers. We have used $t_{\min} = 0.1\,\mathrm{m}$, as this yields good results, whereas smaller distances lead to rendering artifacts, further limiting the feasibility of dense atmospheric rendering.

To compare the efficiency of our method to that of APHRODITE [13], we split the atmosphere into $\Delta_y = 100\,\mathrm{m}$ segments and run three simulations with differing sample counts on the GPU using Halton sampling on the sphere, see Figure 4.15. We have $\tau(2^{20}) = 0.384\,\mathrm{s}$, $\tau(2^{25}) = 0.723\,\mathrm{s}$ and $\tau(2^{30}) = 11.395\,\mathrm{s}$ with 309478 microphones. For $n = 2^{20}$ and $n = 2^{25}$ samples, the performance of NoiseTracer beats that of APHRODITE. There, the fastest render with about 250000 microphones took about 1.3 s on an Intel Xeon E5-2650 V2 processor [13]. Note that this CPU is from 2013. A fairer comparison using a modern CPU will show less difference in performance.

Scene	CPU	CPU (spec. atmos.)	CPU (BVH)	GPU
`flathigh`	3.39 s	0.12 s	0.008 s	0.0006 s
`flatlow`	4.09 s	0.50 s	0.009 s	0.0006 s
`alpshigh`	10.85 s	0.99 s	0.011 s	0.0007 s
`alpslow`	16.06 s	2.94 s	0.013 s	0.0006 s
`canyonhigh`	1.84 s	0.16 s	0.007 s	0.0007 s
`canyonlow`	3.27 s	0.79 s	0.011 s	0.0007 s

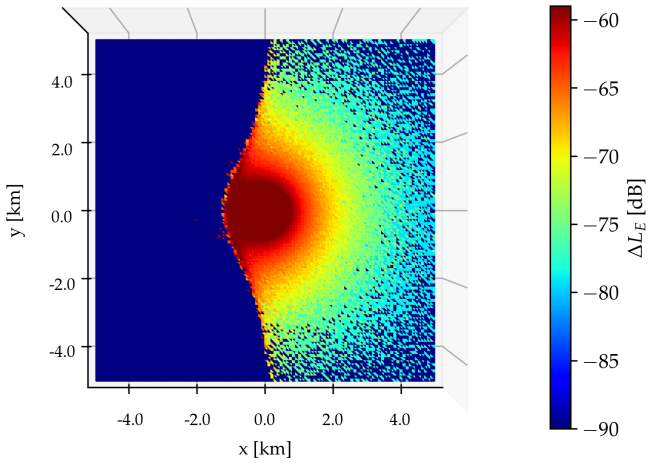
Table 4.4: Comparison of rendering time for `alpshigh` with $n = 2^{10}$ samples, excluding scene setup time.



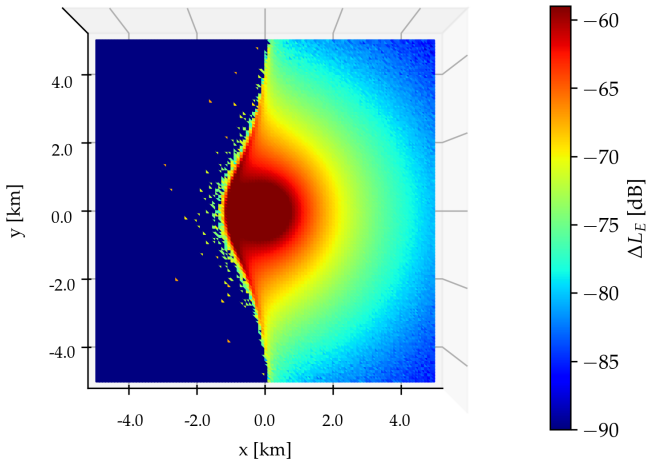
(a) Rendering time versus triangle count

(b) Rendering time versus layer count

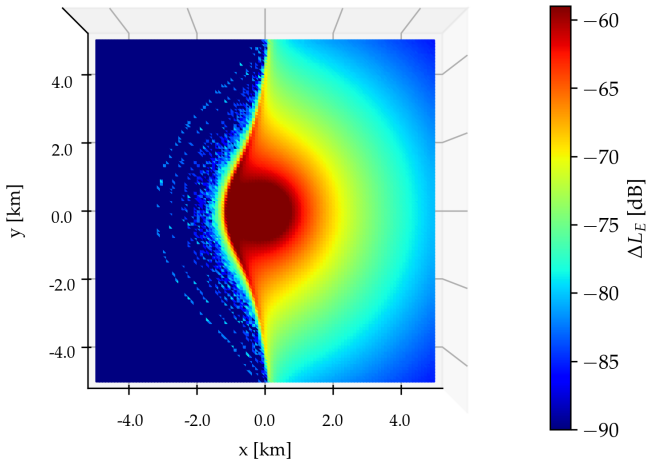
Figure 4.14: Impact of triangle and atmospheric layer count on rendering time. Using a flat 50 km × 50 km scene with sound source at 9500 m AGL and atmospheric profile taken from Figure 13 in [13]. Number of triangles was set using the subdivisions parameter. Simulations were run on the GPU with Halton sampling and $n = 2^{30}$ samples. Ground reflections were disabled.



(a) $n = 2^{20}$ samples



(b) $n = 2^{25}$ samples



(c) $n = 2^{30}$ samples

Figure 4.15: Renders of scene from [13] for different sample counts. Rendering times are $\tau(2^{20}) = 0.384\,\text{s}$, $\tau(2^{25}) = 0.723\,\text{s}$ and $\tau(2^{30}) = 11.395\,\text{s}$ with 309478 microphones.

5 CONCLUSION

We have implemented an efficient method for approximating the ground noise levels for an elevated sound source based on atmospheric conditions. We have shown the validity of various physical phenomena which are simulated using this method. Furthermore, we have shown the differing effectiveness of variance reduction methods in different scenes as well as the significant performance increase due to hardware acceleration.

For scenes with a disabled atmosphere, NoiseTracer returns close to analytical results. In scenes with wind and temperature gradients, we have shown results similar to those published, yet we have also shown the reduced accuracy of the linear ray segment approximation. Although high accuracy renders with dense atmospheres are possible, they are not feasible for interactive use.

The degree of desired accuracy depends heavily on the use case. For approximate estimation of the sound distribution for given atmospheric conditions, NoiseTracer may provide a helpful starting point. For more detailed analysis, such is for example required for the noise certification of aircraft, more accurate methods are required.

LIMITATIONS AND FUTURE WORK The accuracy of the linear ray segment approximation has only been treated superficially. A more in-depth analysis, including the reproduction of data of other methods for quantitative comparison, would offer a clearer understanding of its limitations and areas for improvement.

Alternatively, analytical solutions to the refraction in linearly varying stratified media exist, for example, see [13]. Since the goal of this thesis was to implement an efficient method using GPU accelerated traversal and, more importantly, intersection tests, we chose the linear ray model. Whether closed-form solutions on the GPU can reach similar performance remains to be investigated.

There is some promising recent work on diffraction in the geometrical acoustics model [27] and wave-optics ray tracing [28]. The accuracy of NoiseTracer may be improved by integrating these advances.

Although basic texture functionality is supported by NoiseTracer, ground impedance models used in outdoor sound propagation such as the Delany-Bazley model, see for example [13], were not implemented, and may further improve the accuracy of NoiseTracer.

The assumption of a sound source with spherical directivity is limited. The extension of NoiseTracer to support directional sound sources is straightforward and would increase its applicability.

In the case of highly elevated sound sources, it may take several seconds for sound to reach terrain. The assumption of a constant wind profile during this propagation time lacks realism. Introducing the time domain would

allow for dynamic wind profiles, which could more accurately represent real-world conditions.

Additional optimization potential exists in the way that intersections with atmospheric layers are tested. First, this functionality is not included in the GPU implementation where it would lead to even better performance. Second, instead of considering all atmospheric layers for intersection when above the highest point of terrain, only the next layer may be tested. We predict that this would yield significantly better performance in atmospheres with dense layers.

BIBLIOGRAPHY

[1] Charles D Ross. "Outdoor Sound Propagation in the US Civil War." In: *Applied Acoustics* 59.2 (Feb. 2000), pp. 137–147. ISSN: 0003682X. DOI: 10.1016/S0003-682X(99)00022-5.

[2] Ray Kirby. "Atmospheric Sound Propagation in a Moving Fluid above an Impedance Plane: Application of the Semi-Analytic Finite Element Method." In: *The Journal of the Acoustical Society of America* 149.2 (Feb. 2021), pp. 1285–1295. ISSN: 0001-4966, 1520-8524. DOI: 10.1121/10.0003567.

[3] Susana Quirós Y Alpera, Finn Jacobsen, Peter Møller Juhl, and Vicente Cutanda Henríquez. "A BEM Approach to Validate a Model for Predicting Sound Propagation over Non-Flat Terrain." In: *Applied Acoustics* 64.8 (Aug. 2003), pp. 781–791. ISSN: 0003682X. DOI: 10.1016/S0003-682X(03)00033-1.

[4] Vladimir E. Ostashev and Timothy Van Renterghem. "Equations for Finite-Difference, Time-Domain Simulation of Sound Propagation in Moving Media with Arbitrary Mach Numbers." In: *The Journal of the Acoustical Society of America* 153.4 (Apr. 2023), p. 2203. ISSN: 0001-4966. DOI: 10.1121/10.0017834.

[5] B. Kapralos, M. Jenkin, and E. Milios. "Sonel Mapping: Acoustic Modeling Utilizing an Acoustic Version of Photon Mapping." In: *Proceedings. Second International Conference on Creating, Connecting and Collaborating through Computing*. Ottawa, Canada: IEEE, 2004, pp. 1–6. ISBN: 978-0-7803-8817-8. DOI: 10.1109/HAVE.2004.1391872.

[6] Bill Kapralos, Michael Jenkin, and Evangelos Milios. "Sonel Mapping: A Probabilistic Acoustical Modeling Method." In: *Building Acoustics* 15.4 (Dec. 2008), pp. 289–313. ISSN: 1351-010X, 2059-8025. DOI: 10.1260/135101008786939973.

[7] Samuel Siltanen, Tapio Lokki, Sami Kiminki, and Lauri Savioja. "The Room Acoustic Rendering Equation." In: *The Journal of the Acoustical Society of America* 122.3 (Sept. 2007), pp. 1624–1635. ISSN: 0001-4966, 1520-8524. DOI: 10.1121/1.2766781.

[8] Lakulish Antani, Anish Chandak, Lauri Savioja, and Dinesh Manocha. "Interactive Sound Propagation Using Compact Acoustic Transfer Operators." In: *ACM Transactions on Graphics* 31.1 (Jan. 2012), pp. 1–12. ISSN: 0730-0301, 1557-7368. DOI: 10.1145/2077341.2077348.

[9] Chunxiao Cao, Zhong Ren, Carl Schissler, Dinesh Manocha, and Kun Zhou. "Interactive Sound Propagation with Bidirectional Path Tracing." In: *ACM Transactions on Graphics* 35.6 (Nov. 2016), pp. 1–11. ISSN: 0730-0301, 1557-7368. DOI: 10.1145/2980179.2982431.

[10] Johan B.H.M. Schulten. *Computation of Aircraft Noise Propagation through the Atmospheric Boundary Layer*. Tech. rep. Amsterdam, The Netherlands: National Aerospace Laboratory, July 1997.

[11] Stephanie Heath and Gerry McAninch. "Propagation Effects of Wind and Temperture on Acoustic Ground Contour Levels." In: *44th AIAA Aerospace Sciences Meeting and Exhibit*. Reno, Nevada: American Institute of Aeronautics and Astronautics, Jan. 2006. ISBN: 978-1-62410-039-0. DOI: 10.2514/6.2006-411.

[12] Philipp Schäfer and Michael Vorländer. "Atmospheric Ray Tracing: An Efficient, Open-Source Framework for Finding Eigenrays in a Stratified, Moving Medium." In: *Acta Acustica* 5 (2021), p. 26. ISSN: 2681-4617. DOI: 10.1051/aacus/2021018.

[13] Marthijn Tuinstra. "A Fast Atmospheric Sound Propagation Model for Aircraft Noise Prediction." In: *International Journal of Aeroacoustics* 13.5-6 (Oct. 2014), pp. 337–361. ISSN: 1475-472X, 2048-4003. DOI: 10.1260/1475-472X.13.5-6.337.

[14] Eduard Deines. "Acoustic Simulation and Visualization Algorithms." PhD thesis. 2008.

[15] Joacim Stålberg and Mattias Ulmstedt. "GPU Accelerated Ray-tracing for Simulating Sound Propagation in Water." MA thesis. 2019.

[16] Eric Veach. "Robust Monte Carlo Methods for Light Transport Simulation." PhD thesis. Stanford, CA, USA: Stanford University, 1998.

[17] James T. Kajiya. "The Rendering Equation." In: *ACM SIGGRAPH Computer Graphics* 20.4 (Aug. 1986), pp. 143–150. ISSN: 0097-8930. DOI: 10.1145/15886.15902.

[18] Tomas Möller and Ben Trumbore. "Fast, Minimum Storage Ray-Triangle Intersection." In: *Journal of Graphics Tools* 2.1 (Jan. 1997), pp. 21–28. ISSN: 1086-7651. DOI: 10.1080/10867651.1997.10487468.

[19] Matt Pharr, Wenzel Jakob, and Greg Humphreys. *Physically Based Rendering, fourth edition: From Theory to Implementation*. 4th ed. Cambridge: The MIT Press, 2023. ISBN: 978-0-262-04802-6.

[20] J. H. Halton. "On the Efficiency of Certain Quasi-Random Sequences of Points in Evaluating Multi-Dimensional Integrals." In: *Numerische Mathematik* 2.1 (Dec. 1960), pp. 84–90. ISSN: 0029-599X, 0945-3245. DOI: 10.1007/BF01386213.

[21] Allan D. Pierce. *Acoustics: An Introduction to Its Physical Principles and Applications*. Cham: Springer International Publishing, 2019. ISBN: 978-3-030-11213-4. DOI: 10.1007/978-3-030-11214-1.

[22] Jeffrey H. Miles. *Method of Representation of Acoustic Spectra and Reflection Corrections Applied to Externally Blown Flap Noise*. Technical Memorandum NASA-TM-X-3179. National Aeronautics and Space Administration, 1975.

[23] *Electroacoustics - Sound Level Meters - Part 1: Specifications*. IEC Standard 61672-1:2013. International Electrotechnical Commission, 2013.

[24] Vladimir E. Ostashev and D. Keith Wilson. *Acoustics in Moving Inhomogeneous Media*. 2nd ed. CRC Press, Aug. 2015. ISBN: 978-0-415-56416-8. DOI: 10.1201/b18922.

[25] Samuel Siltanen, Tapio Lokki, and Lauri Savioja. "Rays or Waves? Understanding the Strengths and Weaknesses of Computational Room Acoustics Modeling Techniques." In: *Proceedings of the International Symposium on Room Acoustics*. Melbourne, Australia, 2010.

[26] Heinrich Kuttruff. *Room Acoustics*. 6th ed. Boca Raton: CRC Press, Oct. 2016. ISBN: 978-1-315-37215-0. DOI: 10.1201/9781315372150.

[27] Chunxiao Cao, Zili An, Zhong Ren, Dinesh Manocha, and Kun Zhou. *BEDRF: Bidirectional Edge Diffraction Response Function for Interactive Sound Propagation*. 2023. DOI: 10.48550/arXiv.2306.01974.

[28] Shlomi Steinberg, Ravi Ramamoorthi, Benedikt Bitterli, Eugene d'Eon, Ling-Qi Yan, and Matt Pharr. *A Generalized Ray Formulation For Wave-Optics Rendering*. Jan. 2024. DOI: 10.48550/arXiv.2303.15762. arXiv: 2303.15762 [cs].

[29] John William Strutt 3rd Baron Rayleigh. *The Theory of Sound*. 1st ed. Cambridge University Press, 2011 (1877). ISBN: 978-1-108-03220-9. DOI: 10.1017/CBO9781139058087.

[30] D. Hohenwarter and F. Jelinek. "Snell's Law of Refraction and Sound Rays for a Moving Medium." In: *Acta Acustica united with Acustica* 86 (2000), pp. 1–14.

[31] Oleg A. Godin. "An Effective Quiescent Medium for Sound Propagating through an Inhomogeneous, Moving Fluid." In: *The Journal of the Acoustical Society of America* 112.4 (Oct. 2002), pp. 1269–1275. ISSN: 0001-4966, 1520-8524. DOI: 10.1121/1.1504853.

[32] *Attenuation of Sound during Propagation Outdoors - Part 1: Calculation of the Absorption of Sound by the Atmosphere*. ISO Standard 9613-1:1993. International Standards Organisation, 1993.

[33] O.A. Alduchov and R.E. Eskridge. *Improved Magnus' Form Approximation of Saturation Vapor Pressure*. Tech. rep. DOE/ER/61011–T6, 548871. Nov. 1997, DOE/ER/61011–T6, 548871. DOI: 10.2172/548871.

[34] A. G. Davenport. "Rationale for Determining Design Wind Velocities." In: *Journal of the Structural Division* 86.5 (May 1960), pp. 39–68. ISSN: 0044-8001, 2690-3377. DOI: 10.1061/JSDEAG.0000521.

[35] W.F.J. Olsman and M. Lummer. "Influence of Wind on the Noise Footprint of a Helicopter Landing." In: *Proceedings of the 38th European Rotorcraft Forum*. Amsterdam, The Netherlands, Jan. 2012.

[36] Michael Vorländer. *Auralization*. RWTHedition. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008. ISBN: 978-3-540-48829-3. DOI: 10.1007/978-3-540-48830-9.

[37] Peter Shirley, Trevor Black, and Steve Hollasch. *Ray Tracing in One Weekend*. https://raytracing.github.io/books/RayTracingInOneWeekend.html.

[38] *Standard Atmosphere*. ISO Standard 2533:1975. International Standards Organisation, 1975.

[39] The Matplotlib Development Team. *Matplotlib: Visualization with Python*. Zenodo. Aug. 2024. DOI: 10.5281/ZENODO.592536.

[40] A. Woo, A. Pearce, and M. Ouellette. "It's Really Not a Rendering Bug, You See." In: *IEEE Computer Graphics and Applications* 16.5 (Sept. 1996), pp. 21–25. ISSN: 0272-1716. DOI: 10.1109/38.536271.

[41] OpenMP Architecture Review Board. *OpenMP Application Program Interface Version 3.0*. May 2008. URL: http://www.openmp.org/mp-documents/spec30.pdf.

[42] Steven G. Parker et al. "OptiX: A General Purpose Ray Tracing Engine." In: *ACM Transactions on Graphics* 29.4 (July 2010), pp. 1–13. ISSN: 0730-0301, 1557-7368. DOI: 10.1145/1778765.1778803.

[43] David B. Kirk and Wen-mei W. Hwu. *Programming Massively Parallel Processors, Third Edition: A Hands-on Approach*. 3rd. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., Nov. 2016. ISBN: 978-0-12-811986-0.

[44] Makoto Matsumoto and Takuji Nishimura. "Mersenne Twister: A 623-Dimensionally Equidistributed Uniform Pseudo-Random Number Generator." In: *ACM Transactions on Modeling and Computer Simulation* 8.1 (Jan. 1998), pp. 3–30. ISSN: 1049-3301, 1558-1195. DOI: 10.1145/272991.272995.

[45] Tom G. Farr et al. "The Shuttle Radar Topography Mission." In: *Reviews of Geophysics* 45.2 (June 2007), 2005RG000183. ISSN: 8755-1209, 1944-9208. DOI: 10.1029/2005RG000183.

[46] A. Jarvis, E. Guevara, H. I. Reuter, and A. D. Nelson. *Hole-Filled SRTM for the Globe : Version 4 : Data Grid*. 2008.

A NOISETRACER PARAMETERS

Name	Type	Description
`samples`	`size_t`	number of samples
`maxRecursionDepth`	`size_t`	maximum depth of recursion
`sourceSamplingMethod`	independent, stratified, halton	method for sampling initial directions at sound source
`sourceSamplingDistribution`	uniformHemisphere, cosineHemisphere, uniformSphere	distribution for sampling initial directions at sound source
`diffuseSamplingDistribution`	uniform, bsdf	distribution for sampling directions on reflection
`russianRouletteTermination`	`bool`	Russian Roulette ray termination
`bvh`	`bool`	Bounding Volume Hierarchy for CPU rendering

Table A.1: Rendering Parameters

Name	Type	Description
`terrain.heightmap`	`string`	relative path to heightmap data
`terrain.tileWidth`	`float`	width of a tile in meters
`terrain.tileHeight`	`float`	height of a tile in meters
`terrain.subdivisions`	`int`	subdivision of tiles
`terrain.flatten`	`bool` [optional]	flatten terrain
`terrain.material.reflect`	`bool`	ground reflections
`terrain.material.diffuse`	`float`	amount of sound that is reflected diffusely as opposed to specularly
`terrain.microphones.responsivity`	cosine, linear	directionality of microphones
`terrain.texture`	`string` [optional]	path to texture file, overrides `terrain.material` options

Table A.2: Terrain Parameters

Name	Type	Description
`position.x`	`float`	x-coordinate of plane
`position.z`	`float`	z-coordinate of plane
`position.heightAboveGround`	`float`	AGL altitude
`sound.powerLevel`	`float`	sound power level in dB
`sound.data`	`string` [optional]	path to file containing sound power level per frequency band
`sound.velocity.x`	`float`	x-velocity for Doppler effect
`sound.velocity.y`	`float`	y-velocity for Doppler effect
`sound.velocity.z`	`float`	z-velocity for Doppler effect

Table A.3: Sound Source Parameters

Name	Type	Description
`atmosphere.enabled`	`bool`	atmospheric refraction
`atmosphere.data`	`string`	relative path to atmospheric profile
`atmosphere.specialIntersectionTreatment`	`bool`	special atmospheric treatment for CPU rendering
`atmosphere.attenuation`	`bool`	enable frequency-dependent attenuation

Table A.4: Atmosphere Parameters

DECLARATION

I hereby declare, that I am the sole author and composer of my thesis and that no other sources or learning aids, other than those listed, have been used. Furthermore, I declare that I have acknowledged the work of others by providing detailed references of said work.

I also hereby declare that my thesis has not been prepared for another examination or assignment, either in its entirety or excerpts thereof.

Place, Date	Signature