

Large-Scale Noise Simulation and Visualization of Moving Point Sources

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Medieninformatik

eingereicht von

Clemens Arbesser

Matrikelnummer 0625176

an der
Fakultät für Informatik der Technischen Universität Wien

Betreuung: Ao.Univ.Prof. Dipl.-Ing. Dr.techn. Eduard Gröller

Mitwirkung: Univ.Ass. Dipl.-Ing. Johanna Schmidt

Wien, 12.09.2013

(Unterschrift Verfasser)

(Unterschrift Betreuung)

Large-Scale Noise Simulation and Visualization of Moving Point Sources

MASTER'S THESIS

submitted in partial fulfillment of the requirements for the degree of

Diplom-Ingenieur

in

Media Informatics

by

Clemens Arbesser

Registration Number 0625176

to the Faculty of Informatics
at the Vienna University of Technology

Advisor: Ao.Univ.Prof. Dipl.-Ing. Dr.techn. Eduard Gröller
Assistance: Univ.Ass. Dipl.-Ing. Johanna Schmidt

Vienna, 12.09.2013

(Signature of Author)

(Signature of Advisor)

Erklärung zur Verfassung der Arbeit

Clemens Arbesser
Schlossweg 1, 8724 Spielberg

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit - einschließlich Tabellen, Karten und Abbildungen -, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

(Ort, Datum)

(Unterschrift Verfasser)

Acknowledgements

I would like to thank my family and friends, who always stood at my side and supported me with words and deeds all the way throughout my studies and beyond. Special thanks go to my father, who provided me with standardized sonars and noise measurements and also performed additional field experiments to help to verify and falsify this thesis' results. I would also like to extend my warmest thanks to the supervisor of my thesis, Ms. Johanna Schmidt, for her insightful comments and feedback.

Abstract

Noise pollution is an ever increasing problem not just in urban environments but also in more rural areas such as small villages, along country roads or even in very sparsely populated regions. The demands of the industry and local governments often clash with the interests of people in the neighborhood, creating areas of conflict that often end up in court. Though in many countries noise assessments are mandatory in order to obtain building permission, these documents are usually not suited or sometimes conceivably not even intended to convey the impact of projects on their environment to the general public.

The purpose of this master's thesis is to propose ways to simulate and visualize noise pollution in large-scale, non-urban environments in order to help communicate the impact of new sound emitters on affected neighbors. Knowledge of noise propagation, the influence of the terrain and other obstacles as well as how different emitters add up can provide valuable insights and help in the decision-making process. This knowledge may be particularly helpful when trying to decide on suitable locations for noise screens and/or when trying to find good places to offset some of the local noise emitters.

The tool developed uses NVIDIA's CUDA architecture and the European norm *ISO 9613-2: Attenuation of sound during propagation outdoors* to create real-time visualizations in both 2D and 3D. Results are compared against ground truth data obtained by taking noise measurements in the field.

Kurzfassung

Lärmverschmutzung ist heutzutage nicht nur ein Problem in städtischen Umgebungen sondern zunehmend auch in ländlichen Gegenden wie kleinen Dörfern, entlang von Landstraßen, oder sogar in sehr spärlich bewohnten Gegenden. Forderungen der Industrie und der jeweiligen lokalen Regierungen treffen oft auf die Interessen der Nachbarn, wodurch Konflikte entstehen, die mitunter schließlich vor Gericht landen. Zwar sind in vielen Ländern Schallgutachten notwendig, um Baugenehmigungen für (Lärm erzeugende) Projekte zu erhalten, doch sind diese Dokumente nicht geeignet bzw. gegebenenfalls bewusst nicht dafür konzipiert, den Einfluss dieser Projekte auf ihre Umwelt einer breiten Öffentlichkeit zu kommunizieren.

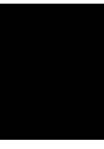
Ziel dieser Diplomarbeit ist es deshalb, Mittel und Wege zur Simulation und Visualisierung von Lärmverschmutzung in großräumigen, nicht-städtischen Umgebungen vorzustellen, um den Einfluss von neuen Schallemittern betroffenen Nachbarn zu kommunizieren. Wissen über die Lärmausbreitung, den Einfluss von Terrain und anderer Hindernisse sowie Kenntnis darüber, wie verschiedene Emitter aufaddiert werden, kann wertvolle Einsichten liefern und im Entscheidungsprozess helfen, Lärmschutzwände an geeigneten Stellen aufzustellen und/oder geeignete Plätze zu finden, um Teile der lokalen Schall-Emitter auszulagern.

Die entwickelte Software basiert auf NVIDIA's CUDA Architektur sowie auf der europäischen Norm „ISO 9613-2: Attenuation of sound during propagation outdoors“, um Echtzeit-Visualisierungen in 2D und 3D zu erzeugen. Die erzielten Resultate werden mit konkreten Schallmessungen verglichen und auf ihre Genauigkeit überprüft.

Contents

1	Introduction	1
1.1	Definitions	2
1.2	Theoretical Foundations	3
1.3	Problem Statement	4
1.4	Methodology	4
1.5	Structure of the Work	6
2	Related Work	7
2.1	Noise Simulation and Visualization	7
2.2	Commercial Software Packages	8
2.3	GPGPU with CUDA	10
3	Noise Simulator - A real-time Noise Simulation/Visualization Tool	15
3.1	Overview	15
3.2	Ensuring Memory and Other Constraints	16
3.3	User Interface and Features	17
3.4	Implementation Specifics	20
3.5	Performance of <i>Noise Simulator</i>	21
4	Real-Time Noise Simulation	23
4.1	Calculation along Sampled Grids of the Terrain	24
4.2	A_{div} : Geometric Sound Attenuation	25
4.3	A_{atm} : Atmospheric Sound Attenuation	25
4.4	A_{gr} : Ground Sound Attenuation	28
4.5	A_{bar} : Sound Attenuation by Barriers	30
4.6	A_{misc} : Miscellaneous Sound Attenuation	33
4.7	Convex Distance Calculation in CUDA	34
4.8	Noise Calculation in CUDA	36
5	Real-Time Noise Visualization	39
5.1	2D Noise Overlays	41
5.2	Noise Propagation Terrain Slices	41
5.3	3D Immission Graphs	46

6	Noise Simulation Results	51
6.1	Ground Truth Data and Expected Simulation Errors	51
6.2	The Red Bull Ring in Styria, Austria	52
6.3	Construction of a Street Bypass to lower Traffic Noise in Canyon Village	56
6.4	Green Valley, Elevation of Highway Noise Screens	59
7	Noise Visualization Results	65
7.1	The Red Bull Ring in Styria, Noise Barrier Evaluation	66
7.2	Canyon Village, Highway Noise Visualization	69
7.3	Green Valley, Noise Screen Refurbishment versus Speed Reduction	71
8	Conclusion and Future Work	73
8.1	Conclusion	73
8.2	Future Work	74
	Bibliography	77



Introduction

The simulation and visualization of noise has many applications, not the least of which being the evaluation of future noise emitting projects in order to obtain building permission from the government. These noise assessments are mandatory in many countries¹, but are usually not intended to be communicated to the general public or even just affected neighbors. This is mainly due to the following reasons:

- Noise assessments are usually conducted on behalf of the company that seeks building permission and naturally does not have an interest to agitate or mobilize affected citizens
- Local governments are often interested in attracting new projects and companies to increase their municipality's wealth and influence and might similarly choose to hinder or downright obstruct public criticism or mobilization

Furthermore, noise assessors who are charged with creating the necessary assessments might be biased since they are employed by a company that is interested in a *positive* assessment. All of the above may lead to affected neighbors being steamrolled by corporate projects and having their quality of living severely diminished.

The software presented in this thesis is not a tool designed to empower individual citizens; it rather offers opportunities to identify problems before they arise and - most of all - facilitate communication between the government, companies and neighbors. Including neighbors into the planning process from the beginning not only lowers the possibility of said neighbors forming citizens' initiatives and filing potentially very long and expensive court cases, it also improves a company's public appearance. We therefore argue that there is no good reason *not* to develop projects in close cooperation with affected neighbors. In this sense *Noise Simulator* is intended to facilitate the communication of complex noise scenarios between experts and amateurs.

¹Such as countries in the European Union.

It is important to note that the software developed does not replace existing noise simulation or visualization software - it is intended to be a natural extension to them. Rapid computation with NVidia's GPGPU architecture 'Compute Unified Device Architecture' (henceforth called CUDA, see NVidia Corporation [26]) as well as the use of moving *point* sound sources as opposed to *line* sound sources, an approximation used by many noise simulation software packages like IMMI [39] and CadnaA [11] (see also Chapter 2), allows experts to evaluate a variety of scenarios, e.g. by creating or removing streets, populating them with vehicles, placing noise screens and generally observing how various aspects of the scene contribute to the simulated noise pollution in the neighborhood.

The generated visualizations are designed to be easily understandable and suitable to be communicated to the general public. However, the software itself is solely intended to be used by domain experts as a supplementary tool to facilitate the creation of noise assessments and the like.

In the remainder of this chapter, after a brief definition of the terms *noise* and *large-scale* in Section 1.1, some noise propagation models are discussed and the usage of ISO 9613-2 [16] is motivated in Section 1.2. Finally, the problem this thesis tackles is stated and formalized (Section 1.3) and the methodology involved is discussed (Section 1.4). At the end, the structure of this thesis is outlined (Section 1.5).

1.1 Definitions

Noise Noise in the sound domain can be defined simply as any kind of *annoying* sound. Since the term *annoying* requires a subjective interpretation, so does the term *noise*. To simplify matters, we restrict noise to those sound sources that affect large outdoor areas and emit a sound pressure far above the environmental basic noise level. This definition specifically entails traffic noise, but also non-moving emissions such as the kind emitted by industrial sites. The use of ISO 9613 [16] (see Section 1.3) further limits the number of available noise emitters since it is *not* applicable to ships, planes and any activities resulting in pressure waves such as explosions. The resulting noise emitter set is still very large and covers almost all common sources of environmental noise.

Large-scale Since the calculation method that we adapt in this thesis, namely the European norm ISO 9613 [16], does only specify accuracy values for distances below one kilometer we will also only consider areas within a one kilometer radius around each emitter. In the case of moving emitters, this area is instead the union of all possible one kilometer areas along the road. This suffices for many, but not all practical situations. Especially when considering very loud emitters (e.g., racing cars), the potential influence area may be significantly larger than one kilometer. Since ISO 9613 is, despite its shortcomings, the *de-facto* standard for noise propagation models, we will exclude these situations from this thesis.

1.2 Theoretical Foundations

Physically, sound travels by compressing and decompressing (rarefaction) air molecules which in turn generate compression and rarefaction waves that are propagated outwards at the speed of sound. Unfortunately, unless one considers only sound propagation in controlled environments, the calculation of sound wave propagation is very elaborate and for larger distances impractical or even impossible, because of the multitude of factors influencing this process. These factors include (but are not restricted to) the following:

- Temperature and humidity of the medium, i.e., the atmosphere. In large-scale outdoor areas, these values are likely to be non-constant.
- Sound refraction by sound barriers, the ground and possibly the atmosphere itself. Furthermore, different materials exhibit different noise refraction qualities.
- Sound reflection by sound barriers and the ground.
- The effect of air absorption, i.e., the diminishment of waves along the propagation path.

Although formulas have been suggested to approximate this process (see for example the works of Storeheier [34] and the more sophisticated models suggest by Plovsing et al. [31] and Kraugh et al. [20]), they are still simplified sound propagation models and are computationally quite expensive to calculate². These models are very valuable when simulating sound propagation along specified directions. Their full evaluation for an entire terrain grid and multiple moving point sources is, at present times, not feasible. Consider for example a moderately sized sampled terrain of 400 samples in both directions and 256 point sources. This evaluates to $400^2 * 256 = 4.09 * 10^7$ full evaluations *per frame*, if the goal is to capture the dynamics of moving point sources in a real-time application. Even modern CPUs are incapable of achieving this kind of performance.

Since the problem of simulating sound propagation is nevertheless very common and important for peoples' lives, governments had to develop easily computable standardized procedures and guidelines applicable to a multitude of scenarios. The result is a host of technical reports, guidelines and formulas, each for a very specific purpose (e.g., train traffic noise, industrial noise, parking garage noise, etc.). To unify these efforts, the European Union issued ISO 9613, the „Attenuation of sound during propagation outdoors“ [15] [16]. The calculation formulae can be found in the second part of this European norm ([16]) and will be referenced hereafter with ISO 9613-2.

ISO 9613-2 specifies a simplified noise propagation model based on physical propagation models, but is nonetheless *empirical* in nature (regression fitting). This model assumes that five factors influence the sound propagation outdoors, namely the geometric attenuation of noise A_{div} , the atmospheric attenuation A_{atm} , the ground attenuation A_{gr} , the sound barrier attenuation A_{bar} and the miscellaneous attenuation A_{misc} (vegetation, housing etc.). These terms are further discussed in Chapter 4. ISO 9613-2 greatly simplifies the problem of sound propagation,

²The most costly operation being the approximation of the various sound refractions, which usually requires the evaluation of a numerical integral [13].

mainly by assuming that sound propagates along lines and by using pre-calculated coefficients in lookup-tables for most of the above terms. Noise refraction is limited to a maximum of 2 noise screens (atmospheric refraction is disregarded). Noise reflection is split up in ground reflections (which are covered by A_{gr}) and mirror reflections which occur in the vicinity of walls and building facades. Unsurprisingly, ISO 9613-2 is very cautious about estimating prediction accuracy and only provides values for the simple case of free sound propagation without refractions and reflections. See Table 6.1 for these values.

1.3 Problem Statement

As will be outlined in Chapter 2, existing noise simulation approaches are entirely CPU-based and simulations may take up hours at a time. These approaches are therefore not suited for exploratory processes or simulations and/or visualizations capturing the dynamics of noise propagation. Allowing the change of parameters at run-time while maintaining at least interactive frame rates can help in both planning and evaluation processes. Consequently, there is a great need for increased performance of noise simulators and visualizations.

Although parallelizing the problem seems to be rather straightforward and promises much better overall performance, not all formulas of ISO 9613-2 can be easily adapted to GPU programming - this is particularly true for noise refractions and reflections. The objective of this thesis is therefore to simplify part of the calculation formulas in ISO 9613-2 to make real-time noise simulation and visualization feasible using CUDA. This not only cuts down on the amount of time needed by experts to simulate large outdoor noise scenarios by one to several orders of magnitude, but also allows them to rapidly create and evaluate a variety of scenarios by inserting and/or changing simulation parameters on-the-fly. For all this to be valid, the revised formulas need to be evaluated as well to see how they hold in real-world scenarios. This is especially important since ISO 9613-2 does not make any accuracy claims itself beyond the simplest case of free, unobstructed sound propagation. Both simulation parameters and simulation results have to be evaluated carefully to ensure the validity and accuracy of the changed formulas.

Finally, to properly communicate the simulated noise values, visualizations have to be designed to be understandable by the general public. On the other hand, they must not sacrifice *accuracy* in order to satisfy the needs of experts as well.

1.4 Methodology

The software developed makes use of NVIDIA's CUDA architecture [26] to allow for the rapid evaluation of the visualization's underlying noise simulation, which in turn uses the European norm ISO 9613-2 to actually compute noise values. To evaluate noise simulation results, field measurement results with standardized sonars (see below) will be compared to simulations of the corresponding scenes using the developed software. Noise visualizations will be created and evaluated in cooperation with both domain experts and laymen.

Noise Measurements, Equipment and Setup Both long-term (several hours to days) and short-term physical noise measurements are necessary to evaluate any given noise simulation



(a) Sample long-term measurement setup consisting of a weather station (to the right of the laptop, sensor outside) and a standardized sonar, in this case a 01dB Type Symphonie (in the suitcase, microphone outside).



(b) Close-up of the Brüel & Kjaer 2270 Sound Level Meter.



(c) Outdoor placement of a standardized sonar. To avoid reflection biasing, the microphone needs to be set up apart from buildings and other structures.



(d) Noise emission measurement of vehicles on an Austrian highway at a distance of 19 meters. Results can be found in Figure 6.7.

Figure 1.1: Noise measurements were taken using either the 01dB Type Symphonie (a) or a Brüel & Kjaer 2270 Sound Level Meter (b). Figures (c) and (d) illustrate the use of these devices for both long-term and short-term noise evaluations.

software. Long-term measurements are required to establish ground truth data to compare simulated values with. Short-term measurements (of individual emitters) are needed to provide the input data for the noise simulation. To acquire the ground truth data, existing noise measurements in suitable areas were adapted (see Sections 6.2, 6.3 and 6.4 for a detailed description of these areas). Figure 1.1a shows a sample setup used in long-term noise measurements, consisting of a weather station and a standardized sonar, usually placed in a sheltered room. Both weather sensor and microphone are placed outside (see Figure 1.1c). For short-term measurements of individual emitters, the portable device Brüel & Kjaer 2270 Sound Level Meter (see

Figure 1.1b) was used. Specifically, noise measurements were taken for DTM racing cars as well as passenger cars and trucks on a sample Austrian highway as seen in Figure 1.1d (see Figure 6.7 for measurement results). Since ISO 9613-2 essentially tries to estimate *maximum* noise immission values we will similarly use the *loudest* measured emission values as input to the presented simulation application (e.g., on highways during rush hour). Certain environmental variables like the weather and especially the wind and wind direction can heavily influence physical noise measurements in ways that ISO 9613-2 can not capture. This however is a shortcoming of ISO 9613-2 and not the simulator. To achieve greater accuracy, the calculation model of ISO 9613-2 can just as easily be replaced by more sophisticated models, as long as they too assume that noise travels along geometric lines. The purpose of this thesis is not to evaluate the accuracy of ISO 9613-2, but to evaluate the accuracy of the *adapted* formulae (see Chapter 4). This ensures that any additional errors introduced by changing these formulae are *small* for the given set of scenarios.

1.5 Structure of the Work

In Chapter 2, existing work and the two most commonly used industrial noise simulators, namely IMMI and CadnaA, are shown. Then, in Chapter 3, the technical details of the developed software are discussed. This includes a breakdown of all implemented features as well as a discussion of memory constraints and run times. Chapter 4 describes the process of parallelizing noise computations using CUDA. Each formula is discussed separately and in as much detail as necessary. After that, the developed visualization techniques are presented in Chapter 5. Both Chapters 4 and 5 focus on the technical aspects of noise simulation and visualization. Results are shown separately in Chapters 6 and 7. Finally, the work is concluded in Chapter 8 and possible future work is discussed.

Related Work

In this chapter the current state of the art concerning the main topics of this thesis is outlined. In particular, this involves the comparison and discussion of various sound propagation models (Section 2.1) and commercial software packages (Section 2.2). A brief introduction to GPGPU-computing using CUDA concludes this chapter (Section 2.3).

2.1 Noise Simulation and Visualization

A summary and evaluation of existing approaches to the problem of noise simulation and visualization can be found in the work of Michel [23]. Evaluations of ISO 9613-2 have already been carried out by others [5, 9, 32]. Though they have found ISO 9613-2 to be sometimes lacking in terms of prediction accuracy, it is still the industry standard of noise propagation. Since in our work - as will be outlined in Chapter 4 - certain formulae in ISO 9613-2 were changed, these existing evaluations are only partly applicable to this thesis and additional evaluations are necessary to verify or falsify our findings (see Chapter 6). Much more relevant to our work are the findings of Parzych [30], who described common misinterpretations of the formulae in ISO 9613-2, specifically pertaining to the calculation of noise barriers.

A considerable number of authors have already tackled the problem of noise simulation and visualization. These approaches are typically tailored to specific use-cases (e.g., indoor sound propagation, urban noise simulation, static or moving sound sources etc.). It is also important to distinguish the simulation of *sound* in general from the simulation of *noise*. Both simulate the propagation of sound waves, but the former is interested in the actual *quality* of sound whereas the latter is merely interested in the *amount* of it.

Examples of *sound* simulation and visualizations can be found in the works of Yakota et. al. [42], Khouri et. al. [18] and Betram et. al. [4]. These simulations often make use of techniques originally developed in the computer graphics domain. Well-known approaches like photon mapping [17] and ray-tracing [2] have their equivalent in the sound domain with phonon mapping [4] and sound-tracing [3].

One of the most popular research areas in the domain of noise is the topic of urban noise prediction [22,29,35]. However, as with most other techniques mentioned here, these approaches are usually static and focus on the sound propagation in areas of rather limited size. Real-time approaches to the problem of simulation or visualization of sound or noise are rare (see for example Park et. al. [29] and Yang et. al. [41]).

As illustrated above, a lot of research has already been done on the subject of noise simulation and/or visualization frameworks, but the research community has shown surprisingly little incentive to either incorporate recent advances in GPGPU-computing into existing approaches or to use them to generate entirely new frameworks. While in the case of phonon mapping or sound-tracing one could simply adapt existing GPU approaches in the visual domain, such is not the case for noise simulations, since their calculation methods and equations are quite different and they do not have an equivalent computer graphics approach. To the best of my knowledge, this thesis will be the first attempt to accelerate noise simulation by using the GPU and, consequently, use the GPU's capabilities to create new visualizations that are not feasible to create on the CPU.

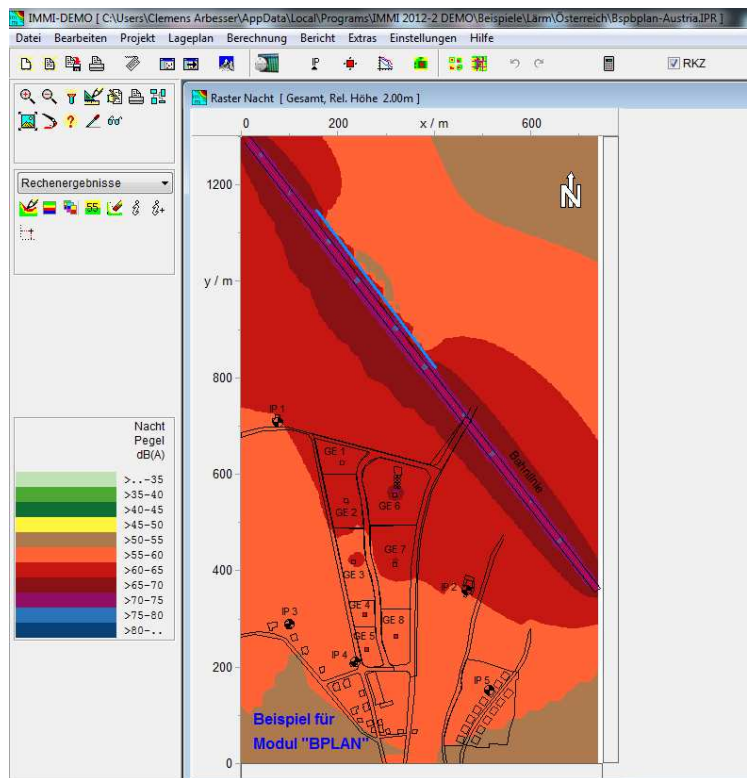
2.2 Commercial Software Packages

Since noise assessments are often necessary to obtain building permissions, it is not surprising to find that there exist several commercially available software packages to simulate noise. In Austria, the two most commonly used programs are IMMI [39] and CadnaA [11]. Both offer the expert a large variety of tools to compute and visualize noise in various scenarios. The array of implemented guidelines and features also includes ISO 9613-2. IMMI as well as CadnaA are CPU-based and as such, computations may take up hours at a time. To speed up this process, both packages allow for the distributed computation of noise simulations. At the time of writing (April 2013), however, they do not implement any kind of GPGPU technology.

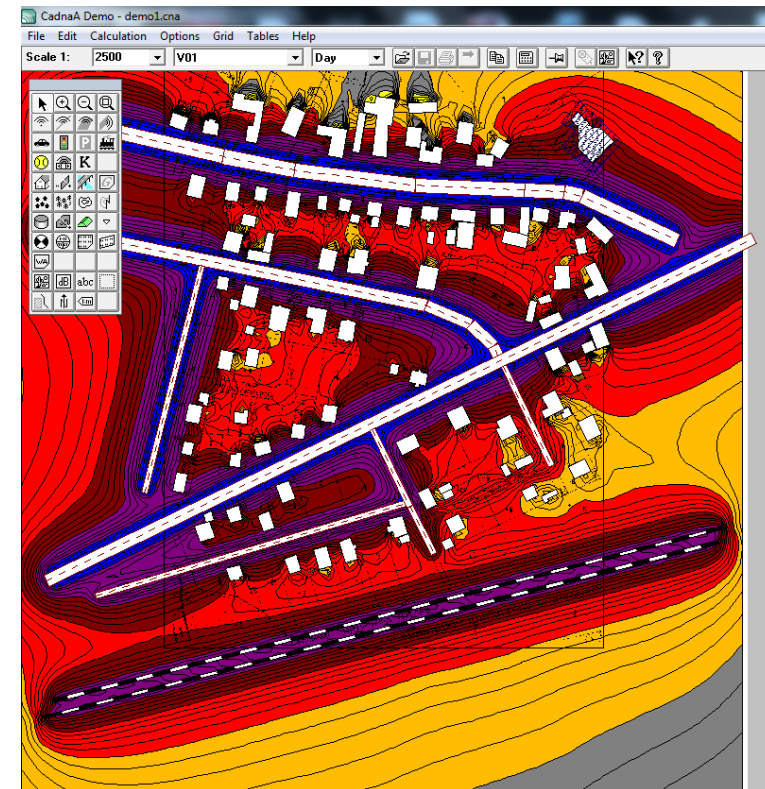
User Interface and Usability IMMI and CadnaA are very similar in their appearance and usage. The user typically loads a map or a plan of the area of interest and then proceeds with placing streets, immission points, obstacles and other objects in the scene. Every time simulation parameters change, the user has to manually launch the simulation to update immission values. Clearly, these programs are not intended to be used as a prototyping or planning, but rather as an evaluation tool, applicable to situations where all parameters are static and the simulation has to be performed only once. Figures 2.1a and 2.1b show screenshots of IMMI and CadnaA, respectively; simulation and visualization capabilities are quite similar.

Advantages and Disadvantages Since IMMI and CadnaA are quite similar in their feature set and applicability, we will not attempt to evaluate them separately. They both offer the user a vast array of possibilities to not only simulate and visualize *noise* using a variety of different guidelines and computation methods, but also related problems like air pollution.

Prices are on request only. In Austria at the time of writing, IMMI is about €4900 for one licence of the basic package (possibly more, depending on the additional modules requested),



(a) Screenshot of a sample session of the demo version of 'IMMI - The Noise Mapping Software'. The effect of a railway track on affected neighbors is displayed using Iso noise areas.



(b) Screenshot of a sample session of the demo version of 'CadnaA'. The effect of various emitters on the local neighborhood is visualized using Iso noise areas.

Figure 2.1: Demo sessions in IMMI and CadnaA. Note that both Demo versions falsify computation results, these visualizations therefore only serve illustrative purposes.

CadnaA starts at €6700 (one norm or guideline of choice) or €9300 (the full set of regulations and guidelines). Both are geared towards the needs of noise assessors and provide them with the necessary tools to facilitate their work. As outlined at the beginning of Chapter 1, however, these assessments are not without problems and are arguably not very well suited or even intended to be communicated to a broad audience.

2.3 GPGPU with CUDA

GPGPU - General-purpose computing on graphics processing units - refers to an ongoing trend in computer sciences to use the graphics cards of regular workstations to compute any kind of large tasks that can be parallelized. To accomplish this, several APIs exist that allow developers to access graphics cards' resources by coding in their language of choice (usually C or C++). The two dominant GPGPU frameworks are OpenCL [19](open) and NVIDIA's CUDA [26] (proprietary). *Noise Simulator* uses the latter, CUDA, to speed up noise simulation computations.

Application Design with CUDA Developing applications with CUDA usually adheres to the following scheme:

1. Write the *gold algorithm*, a CPU implementation of the task, to later compare the GPU implementation to.
2. Divide the code in *host* (CPU) and *device* (GPU) code. Each task that is to be performed by the GPU must be contained in a *Kernel*, a separate piece of code written according to CUDA guidelines.
3. Compile *host* code with the compiler of choice (linking with CUDA libraries) and *device* code with NVIDIA's own compiler *nvcc* [28].
4. Repeat until result (Kernel) = result (gold algorithm):
 - a) Copy all necessary input data to GPU memory.
 - b) Call the *Kernel*, wait for the device to finish computation and retrieve the results.
 - c) Compare the results with the results of the gold algorithm.

Note that the GPU's arithmetic units are different from the CPU's, which will result in small floating point inaccuracies (see Whitehead and Fit-Florea [38] for details on this issue). If accuracy is an issue, more recent NVIDIA graphics cards also support double precision floating point operations (starting with GT200 and compute capability 1.3 and later, see the CUDA C Programming Guide [25] for details).

Creating a CUDA Thread Grid To benefit from the speed of GPUs one has to divide the task at hand into a (large) number of calculations that can be executed concurrently. In CUDA, all threads for a kernel are organized in a single *grid* containing *blocks* of threads that share a common *shared memory* address space (GPU memory will be discussed below, see also Figure

2.2). Grids can be up to 3-dimensional and usually reflect the problem structure. In our case, individual threads correspond to terrain samples and the complete grid represents the sampled terrain. Each thread knows its position inside the thread grid via predefined local variables such as *blockIdx*, *blockDim* and *threadIdx*. For example, in 1-dimensional thread blocks with *grid_width* threads each, individual threads have an index that can be computed as follows:

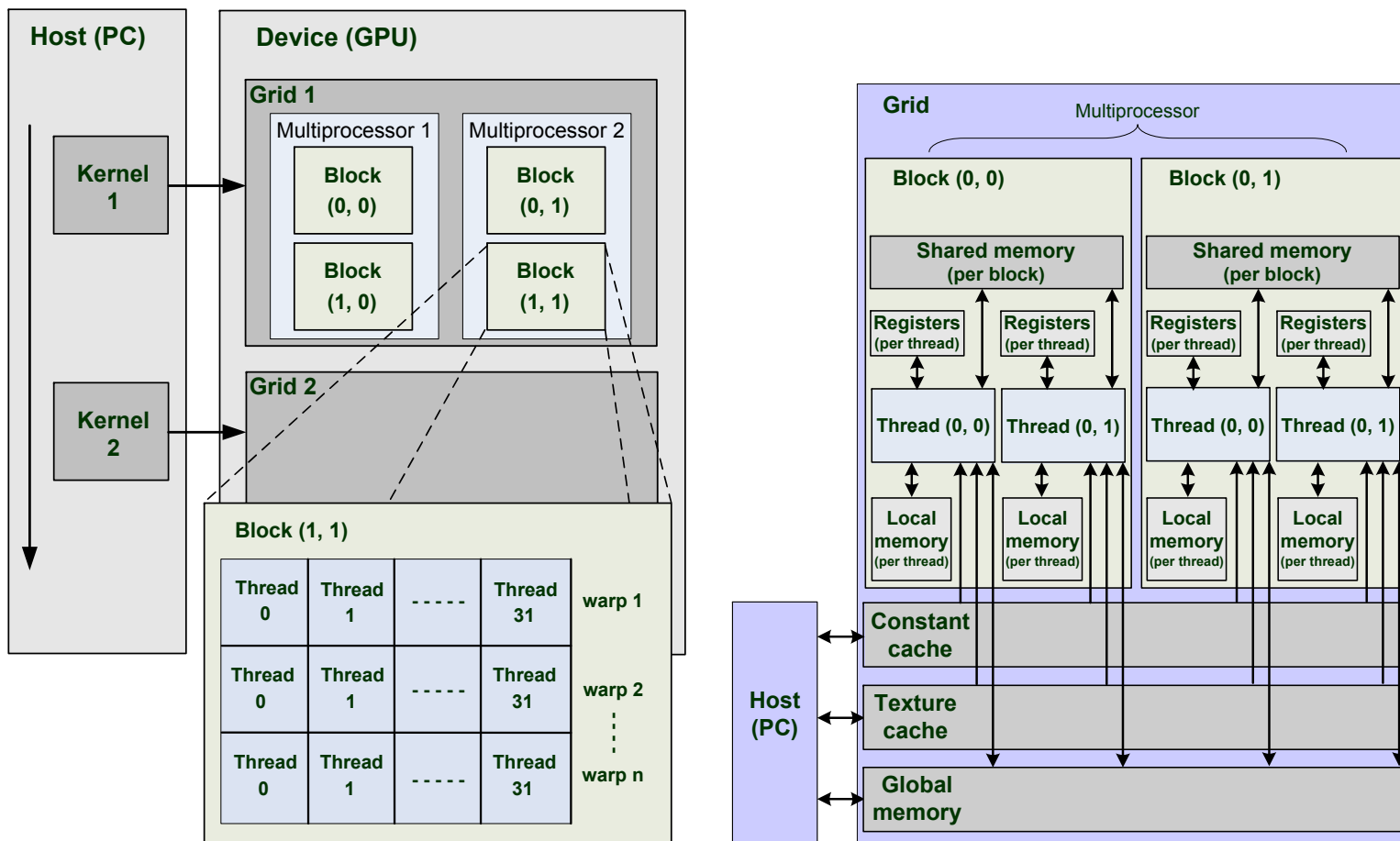
```
unsigned int x = blockIdx.x*blockDim.x + threadIdx.x;
unsigned int y = blockIdx.y*blockDim.y + threadIdx.y;
unsigned int index = (y * grid_width) + x;
```

For some problems the step of organizing threads into grids is straightforward, but other times it can be a daunting challenge for the developer. For algorithms that are difficult to parallelize it is usually better to start from scratch and write a completely new algorithm for CUDA rather than trying to make the existing one work. Chapter 4 explains this process of parallelization for the case of *Noise Simulator*. We will see that the formulae of ISO 9613-2 (which form the core of our CUDA computations) are moderately easy to parallelize, though some alterations and concessions are necessary.

Coding Principles in CUDA Coding in CUDA differs from usual, CPU based development. Indeed the differences are quite large and numerous, and only a small selection will be discussed here. For details on CUDA coding principles as well as guidelines and good practices the reader may refer to the CUDA programming guide [25].

Performance is always at the center of application development in CUDA. This issue is tightly interwoven with memory constraints and parallelizing decisions, all of which have to be considered by the developer if he/she wishes to create a fast application. Arguably the two most important principles when coding in CUDA are the following:

- Each thread must be *independent* from each other (or at least as much as possible). This issue entails (among other things):
 - Avoiding memory bank conflicts which arise when two threads simultaneously access the same memory address.
 - Be aware of and avoid concurrency issues like dead-locks (threads waiting for each other).
- Each thread's task must be *sequential* and should avoid control-flow altering code, because GPU computation is at its fastest when all threads agree on their control-flow:
 - Try to avoid or unroll loops as well as constructs such as 'if' or 'switch', since they must be evaluated at run-time for each thread separately.
 - (Unreflected) use of recursions is strongly discouraged and only available for devices of CUDA compute capability of 2.0 and above.



(a) CUDA organizes threads in grids and blocks. At runtime, threads are executed in groups of 32, called *warps*.

(b) The CUDA memory model.

Figure 2.2: The CUDA thread and memory models. Graphics by Hasan et al. 2011 [14].

What remains is to take care of memory constraints and to use the most appropriate memory for each task. In CUDA, several types of memory with vastly different performances are available (see Figure 2.2b), ranging from the slow global memory to the very fast shared memory and register memory as well as texture memory and constant memory that fall somewhere in-between. Depending on how the data values are likely to be accessed, these memory types perform differently. For example, memory that will be accessed randomly should be placed in texture memory (and not global memory). For a complete breakdown of the benefits and disadvantages of CUDA memory types the reader may refer to the CUDA programming guide [25].

A common strategy to increase performance is to minimize accesses to global memory by writing its contents to shared memory. This can be applied to all problems where threads in the same block will access the same data elements repeatedly. A good example for this is a CUDA implementation of a simple matrix multiplication: The task can be split up in blocks that correspond to independent sub-matrices of the original matrix and shared memory can be used to store these sub-matrices for each block separately, resulting in a large speed-up compared to the global memory implementation (see the official CUDA samples [27] for details).

Noise Simulator - A real-time Noise Simulation/Visualization Tool

To address the issues and shortcomings of existing approaches discussed in Chapter 2, a tool to both simulate and visualize moving point sound sources has been developed. It should be stressed that the focus of this work lies on the *visualization* and not the *simulation* part, although the latter is still an accurate implementation of the calculations outlined in ISO 9613-2 for most outdoor scenes.

This chapter describes in detail the tool developed by first outlining the tool itself, i.e., development platform, features etc. (Section 3.1). Then, implementation specifics regarding the *simulation* part, specifically issues concerning memory constraints are discussed (Section 3.2). After briefly showing the user interface of *Noise Simulator* in Section 3.3, some implementation specifics concerning the terrain and specifically noise screens are mentioned (Section 3.4). Finally, some statistics and runtimes are presented (Section 3.5). Note that solutions and issues are discussed from a *technical* point of view. For results and limitations as well as possible future work see Chapters 6, 7 and 8.

3.1 Overview

Noise Simulator is a 32-bit Windows Application written in C++ to perform real-time noise simulation and visualization. It uses the following 3rd party libraries:

- DevIL image library [40] (GNU Lesser General Public License)
- Fast Light Toolkit 1.32 stable, i.e., FLTK [36] (GNU General Public License)
- TinyXML [37] (ZLib License)

Since very little platform-specific code was used, ports to other operating systems should be fairly easy to build, but no efforts have been made thus far. The following lists the most important implemented features:

- 2D/3D view of the current session
- Tools to draw streets, immission points, emission points and vehicles
- Tools to change the terrain by inserting houses, bridges and forests
- **(Simulation)** real-time calculation of the incoming noise at every point in the dataset (see Section 4.8)
- **(Simulation)** long term simulation (1 hour) for specific, user-chosen immission points (see Section 4.8)
- **(Visualization)** 2D overlay in both 2D and 3D view of all calculated noise values (see Section 5.1)
- **(Visualization)** Noise Propagation Terrain Slices, i.e., slices through the terrain showing how sound travels in specific directions (see Section 5.2)
- **(Visualization)** 3D Immission Graphs, i.e., breaking up immission values into individual contributors for a given immission point (see Section 5.3)

These features will be discussed in more detail in their specific chapters, namely Chapters 4 and 5.

Prerequisites and Setup

Noise Simulator uses NVIDIA's CUDA GPGPU architecture (see Section 2.3) to parallelize certain parts of the computation (see Chapter 4 for details), therefore a CUDA capable graphics card (compute capability of 1.0 suffices) with at least 512 MB dedicated memory (recommended: 1 GB) is required for the program to run. Additionally, two pieces of information are required for the user to provide: (1) A 2D orthographic map of the region of interest, and (2) the corresponding 3D data. For Austria, the former may be obtained free of charge¹ from GIS (Geographic Information System) websites of the respective region (i.e., the Styrian GIS site [21]). 3D data in a grid resolution of 10 meters may also be purchased from GIS². Two Reference points have to be set manually by the user in order for the program to correctly associate 3D data with the provided 2D map. For this thesis, all 3D data and 2D data (maps) were obtained from GIS Austria.

Finally, to obtain meaningful input values for traffic density and noise emission characteristics of vehicles it is *critical* to perform *in-situ* measurements with adequate devices (see Sections 1.4 and 6.1).

3.2 Ensuring Memory and Other Constraints

Calculation of distances and noise values requires extensive amounts of GPU memory. This effectively restricts both the number of threads that can be executed simultaneously on a given graphics card and the size of datasets to process. Table 3.1 lists the approximate memory requirements for terrain data with a spatial resolution of 10 meters and 256 samples in both X

¹For non-commercial purposes.

²By browsing the online catalogue and filling out the corresponding form, both of which are provided at <http://www.gis.steiermark.at/cms/ziel/74005/DE/>.

and Y directions (a 2.56 km² region), 128 possible emitters and a terrain sample rate of 64 per thread. Individual terms will be discussed in the subsequent sections.

Table 3.1: CUDA Memory Requirements of NoiseSimulator

Item	Number * Type	Total
Global Memory (persistent):		
d_{S_SM}	$(256 * 256 * 128) * \text{float}$	32 MB
d_{SM_M}	$(256 * 256 * 128) * \text{float}$	32 MB
d_{M_RM}	$(256 * 256 * 128) * \text{float}$	32 MB
d_{RM_R}	$(256 * 256 * 128) * \text{float}$	32 MB
$mean_height$	$(256 * 256 * 128) * \text{float}$	32 MB
$height_{SM}$	$(256 * 256 * 128) * \text{float}$	32 MB
$height_M$	$(256 * 256 * 128) * \text{float}$	32 MB
$height_{RM}$	$(256 * 256 * 128) * \text{float}$	32 MB
d_{S_M}	$(256 * 256 * 128) * \text{float}$	32 MB
d_{M_R}	$(256 * 256 * 128) * \text{float}$	32 MB
emitters	$128 * (\text{size}(\text{emitter}) = 60 \text{ Byte})$	7.5 KB
textures (overlay, terrain)	$256 * 256 * 2 * \text{float4}$	2 MB
high-resolution terrain offset texture	$2048 * 2048 * \text{float4}$	64 MB
Global Memory Total		$\approx 386 \text{ MB}$
Local Memory (per calculation):		
terrain sample cache	$256 * 256 * 64 * \text{float4}$	64 MB
helper variables	$\approx 256 * 256 * 50 * \text{float}$	12.5 MB
Local Memory Total		$\approx 76.5 \text{ MB}$
Total CUDA Memory Requirements		$\approx 462.5 \text{ MB}$

Note that these requirements do not contain additional memory requirements imposed by the GUI and especially memory consumption by OpenGL for rendering purposes. In test sessions using the above settings, the application was found to consume between 500 and 700 MB of GPU memory. The resolution of the thread grid obviously is the deciding factor and, depending on the graphics card used, effectively limits the attainable accuracy. For GIS terrain data in Austria, which at the time of writing is only available in spatial resolutions of 10 meters and above, sufficiently large regions (up to 2 km²) are supported on most graphics cards. Since noise calculation in areas larger than 2 km² is very inaccurate and unreliable the current memory consumption is efficient enough for most scenes. Larger regions have to be subsampled in an adequate way.

3.3 User Interface and Features

Figure 3.1 illustrates the user interface and features of *Noise Simulator*. Most elements are self-explanatory and implemented in a straightforward way. The terrain-altering tools (Figure 3.1, number 6), however, do require some additional discussion.

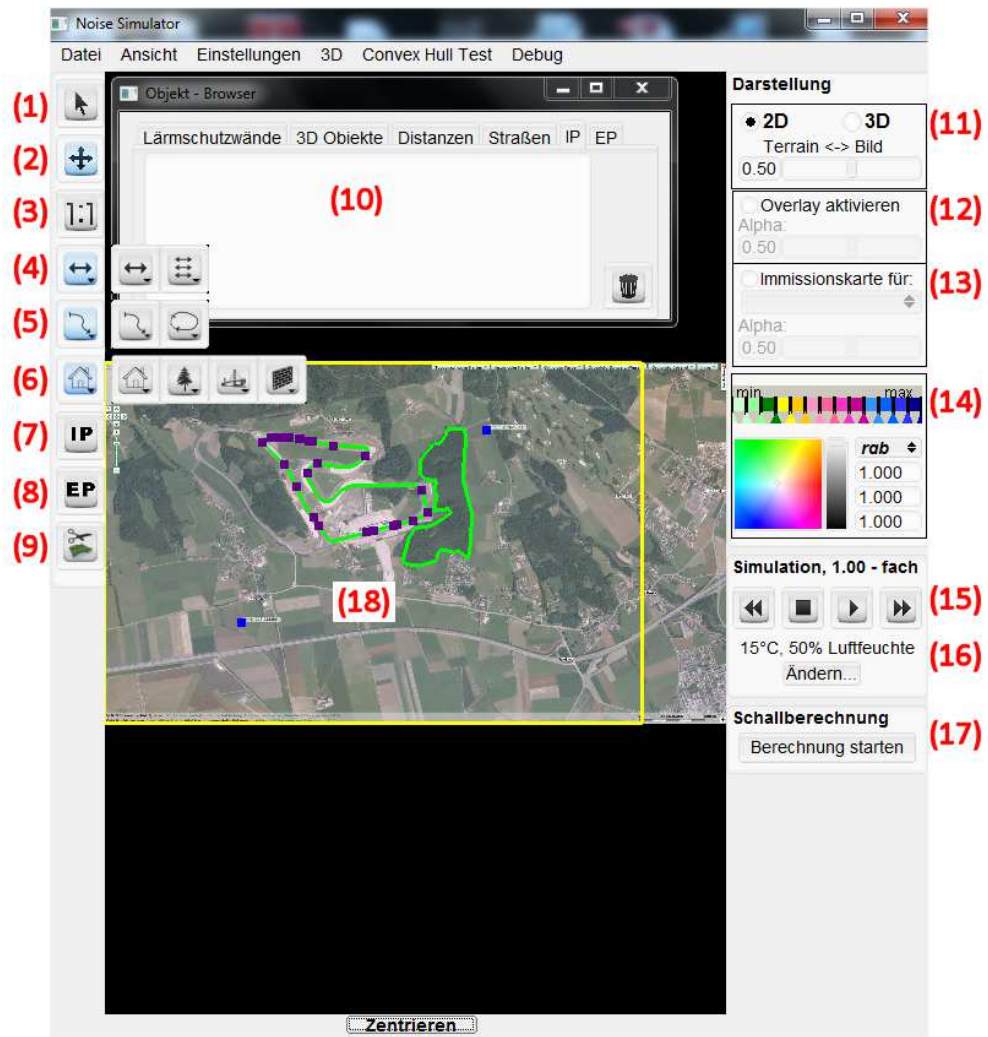


Figure 3.1: The User Interface of *Noise Simulator*.

- | | |
|--|---|
| 1. Object Selection Tool | 10. Object Browser |
| 2. Navigation Tool (Panning) | 11. Switch 2D/3D Perspective |
| 3. Map Scale | 12. 2D Overlay Settings |
| 4. Single- and Multi-Distances | 13. 3D Immission Graph Settings |
| 5. Open and Closed Streets | 14. Noise Transfer Function |
| 6. Building-, Forest-, Bridge- and Wall-
Placement Tool | 15. Simulation Controls |
| 7. Immission Point Placement Tool | 16. Environmental Variables |
| 8. Emission Point Placement Tool | 17. Start/Stop Noise Simulation |
| 9. Terrain Slice Tool | 18. Map Display (2D) or Terrain Rendering
(3D) |

Bridge Placement Tool

Since GIS terrain data does not reflect artificial terrain changes like buildings and bridges it was necessary to develop suitable tools. The Bridge Tool lets the user fill up gaps in the terrain by first selecting two control points on one side of the gap (spanning the whole width of the street) and then placing a third point on the destination side. Terrain values between these points are consequently interpolated to generate a smooth transition between the old terrain and the bridge.

Noise Screen Placement Tool

Contrary to other terrain-changing tools, noise screens are handled using 2D lines instead (see Section 3.4 for the reasons behind this). The user draws point-to-point noise screens and then specifies their height.

3.4 Implementation Specifics

Though most features are implemented in a straightforward way, some specifics regarding terrain-altering techniques are instructive because they portray some of the difficulties when dealing with sampled terrains - they are therefore shortly described below.

Terrain Offset Texture

To allow the user to modify the terrain (e.g., by inserting bridges, houses etc.) without changing the original values, *Noise Simulator* uses a high-resolution offset texture in which all terrain changes are saved. At run-time, terrain values are sampled twice - once for the original terrain and once for the offset texture - and summed up. A resolution of $2048 * 2048$ ensures that most types of buildings and objects can be stored in the offset texture without noticeable sampling artifacts. Note, however, that this texture can still not capture high-frequency objects like thin noise screens. They have to be dealt with outside the offset texture. For this purpose we store noise screens separately as 2D lines which we project onto the terrain. At run-time, we perform a simple line-intersection to determine whether the line Emission Point - Immission Point has intersections with any noise screen and elevate the appropriate terrain samples by the height of the noise screens. In this way we can simulate thin noise screens without introducing additional sampling errors.

Point Inclusion Test

The point inclusion test in *Noise Simulator* is handled by the very efficient code developed by W. Randolph Franklin [12]. Algorithm 3.1 lists the entire code for point inclusion testing for arbitrary polygons.

input : The number of vertices $nvert$, the vertices of the polygon in the arrays $vertx$ and $verty$ as well as a test point $testx$ and $testy$.

output: A boolean specifying whether the test point is inside the given polygon or not.

```

1 i, j, c = 0; for i = 0; j = nvert-1; i < nvert; j = i++ do
2   if ((verty[i]>testy) != (verty[j]>testy)) && (testx < (vertx[j]-vertx[i]) *
   (testy-verty[i]) / (verty[j]-verty[i]) + vertx[i]) then
3     c = !c;
4   end
5 end
6 return c;
```

Algorithm 3.1: Point inclusion test by W. Randolph Franklin [12].

3.5 Performance of Noise Simulator

The performance of *Noise Simulator* is very dependent on the GPU hardware used. Table 3.2 lists timings for different hardware setups and use-cases. Timings include both the frame rates and the precomputation times needed to calculate the convex noise distances (see Section 4.7). These distances have to be computed for every terrain sample, but only *once* unless the terrain changes e.g., by adding or deleting houses, noise screens and the like. The three scenarios are ordered by their dataset size, ranging from 2.98 km² to 6.13 km². Precomputation times range from 1.1 to 6 seconds. Performance generally is very strongly dependent on the graphics card used and less dependent on the available CPU. The relatively bad results for the Laptop 1 configuration in contrast to the Laptop 2 configuration - even though the latter used a less powerful GPU - can be explained by the fact that the latter's GPU was a *dedicated* GPU, while the former had to perform additional calculations as well (OS graphics etc.). The desktop configuration clearly outperforms both laptop configurations. Still, we can see that even on slow systems, *Noise Simulator* retains at least interactive frame rates. With a regular desktop pc configuration, high frame rates can be maintained in all scenes. This is especially true since most scenarios are likely to be no larger than 3 - 4 km² due to ISO 9613-2 only stating prediction accuracies for areas with a radius of 1 km and less. Unless one is interested in simulating long parts of streets at a time (which is the case for the Green Valley scenario), terrain sizes usually stay manageable. This ensures that most real-world scenarios both fit into the GPU memory and can be computed at fast rates.

Table 3.2: *Noise Simulator* Performance for Various Scenes and Configurations

	Laptop Config 1		Laptop Config 2		Desktop PC	
	4 x 2.3 GHz 16 GB RAM GeForce GT 650M		2 x 2.3 GHz 8 GB RAM GeForce GT 555M		4 x 2.67 GHz 12 GB RAM GeForce GTX 580	
Session	fps	precomp. time [sec]	fps	precomp. time [sec]	fps	precomp. time [sec]
Canyon Village (2.98 km ²)	15	4	32	2	121	1.1
Red Bull Ring (4.61 km ²)	12	6	17	2.5	121	1.3
Green Valley (6.13 km ²)	2.5	6	5	3.5	41	1.5

Real-Time Noise Simulation

In this Chapter a specific approach to the problem of real-time noise simulation will be discussed. This approach is based on the fundamentals layed out in Chapter 1 and specifically implements and adapts part of ISO 9613-2 in CUDA. The problem lies in calculating the following equation:

$$L_{fT}(DW) = L_W + D_C - A \quad (4.1)$$

$L_{fT}(DW)$ is the equivalent continuous downwind octave-band sound pressure level at an immission point for *one* emission point and a specific frequency band, L_W is the octave-band sound power level of the emission point in decibel (relative to a reference sound power on one picowatt), D_C is the oriented emission adjustment term in decibel (which we can assume to be 0 dB when dealing with point emissions) and A is the sum of all sound attenuation terms:

$$A = A_{div} + A_{atm} + A_{gr} + A_{bar} + A_{misc} \quad (4.2)$$

A_{div} encodes the distance attenuation, A_{atm} describes the atmospheric attenuation of noise, A_{gr} is the ground attenuation term and A_{bar} and A_{misc} refer to sound barrier attenuation and miscellaneous attenuation (e.g., vegetation), respectively.

To account for the *perceived* loudness by humans, which depends not only on the sound pressure, but also the frequency of the sound, individual frequency bands are multiplied by standardized constants in a process called A-weighting (see the international standard IEC 61672-1 [1]). Different weighting methods exist, but A-weighting is the most common one and is also used by ISO 9613-2. The eight frequency bands and their associated A-weights are shown in Table 4.1. For a multitude of emitters and these eight frequency octave bands, the equivalent continuous A-weighted downwind (5 m/s) sound pressure level $L_{AT}(DW)$ is defined as

$$L_{AT}(DW) = 10 * \log \left\{ \sum_{i=1}^n \left[\sum_{j=1}^8 10^{0.1 * [L_{fT}(ij) + A_f(j)]} \right] \right\} \text{ dB} \quad (4.3)$$

We see that $L_{AT}(DW)$ is the sum of the eight octave band noise pressures (between 63 Hz and 8 kHz, denoted by the index j), summed up over all n noise emitters. $L_{fT}(i, j)$ is the noise

pressure of emitter i in the octave band j , $A_f(j)$ is the A-weight factor of the j 'th octave band as specified in the international standard IEC 61672-1 [1]. For convenience, these values are given in Table 4.1.

Table 4.1: A-weight factors for the Octave Bands between 63 Hz and 8 kHz as specified in the international standard IEC 61672-1 [1].

Frequency [Hz]	A-weight [dB]
63	-26.2
125	-16.1
250	-8.6
500	-3.2
1000	0
2000	1.2
4000	1.0
8000	-1.1

After a brief summarization of problems and solutions to calculate noise along a sampled grid of the terrain (see Section 4.1) the different terms of Equation 4.2 will then be discussed separately in the remainder of this chapter.

4.1 Calculation along Sampled Grids of the Terrain

For any GPU-based approach to simulation and visualization, the problem needs to be expressed in a way that allows for massive parallelization. In the context of sound propagation across large-scale terrain fields, the obvious way to do this is to discretize the terrain by sampling it. The resulting grid should contain enough sample points to accurately reflect the terrain it is based on. Choosing too many sample points, however, may result in CUDA implementation issues (e.g., memory constraints). To parallelize the computation, we only have to generate one thread per sample and have these threads run simultaneously.

Avoiding Sampling Errors Since terrain data is usually gathered along grids anyway, we can use those grids to define our CUDA grid of threads. In other words, we can create a CUDA thread grid of dimensionality equal to the provided terrain, unless said terrain is uncommonly large or exceedingly densely sampled. Specifying the thread grid in this way prevents the introduction of additional sampling errors.

Organizing the CUDA Grid in Blocks and Threads As outlined before, interpreting terrain sample grids as thread grids in the CUDA architecture is a natural way to approach parallel computing on terrains. More importantly, however, this lets us use natural spatial coherency of terrain data to enhance the performance. Generally speaking, one should avoid using thread branching instructions (*if*, *for*, *while* etc.) when programming in CUDA, since massive parallel computation is at its fastest when all threads agree on a singular instruction flow (see Section

2.3). Nevertheless, these instructions are sometimes unavoidable. In those cases, performance hits may be mitigated if at least all 32 concurrently executed threads (also called *thread warp*) have the same or at least similar program flows. Since large-scale terrain data is usually quite smooth we can assume that adjacent threads' calculations will be similar and hence fast.

Divide and Conquer: Splitting-up the Calculation in Individual Kernels As we will see later, calculating sound distances of points in 3D is a non-trivial problem in its own right. However, we can assume these distances to stay constant for most of the time and can hence precompute and store them in a large array to be used as a lookup-table (LUT) later on. In other words, we split-up the noise computation into two kernels:

- Precomputation Phase: Precompute convex sound distances and store them in a LUT table that resides in GPU memory (Section 4.5)
- Run-time Phase: Simulate the noise propagation for all points of the terrain (Sections 4.2, 4.3, 4.4 and 4.6)

4.2 A_{div} : Geometric Sound Attenuation

The geometric sound attenuation term A_{div} describes the spherical, unobstructed attenuation of sound. According to ISO 9613-2, it is calculated as follows:

$$A_{div} = [20\lg(d/d_0) + 11]dB \quad (4.4)$$

where d is the euclidean distance between sender and receiver and d_0 is the reference distance (= 1 m). This equation shows one of the fundamental properties of sound attenuation - the logarithmic relationship between distance and sound pressure. Calculation in CUDA is straightforward and fast - every thread simply has to calculate its distance to every sound emitter and then calculate equation 4.4.

4.3 A_{atm} : Atmospheric Sound Attenuation

The atmospheric sound attenuation term A_{atm} describes how sound of a certain frequency is attenuated when propagating inside an atmosphere of given temperature and humidity. As Cramer [10] points out, the effect of air pressure is negligible. ISO 9613-2 defines the atmospheric sound attenuation as follows:

$$A_{atm} = \alpha d / 1000 \quad (4.5)$$

where α is the air attenuation coefficient in decibel per kilometer and d is the distance in meters. According to ISO 9613-1, the air attenuation coefficient α for a given frequency f [Hertz], temperature T [Kelvin], air humidity h [in percent] and air pressure p_a [kPa] can be calculated as specified in Equations 4.6, 4.7 and 4.8:

$$\begin{aligned}
\alpha = & 8.686 f^2 \left(\left[1.84 * 10^{-11} \left(\frac{p_a}{p_r} \right)^{-1} \left(\frac{T}{T_0} \right)^{\frac{1}{2}} \right] + \right. \\
& + \left(\frac{T}{T_0} \right)^{\frac{-5}{2}} * \left\{ 0.01275 \left[\exp \left(\frac{-2239.1}{T} \right) \right] \left[f_{rO} + \left(\frac{f^2}{f_{rO}} \right) \right]^{-1} + \right. \\
& \left. \left. + 0.1068 \left[\exp \left(\frac{-3352.0}{T} \right) \right] \left[f_{rN} + \left(\frac{f^2}{f_{rN}} \right) \right]^{-1} \right\} \right)
\end{aligned} \tag{4.6}$$

Where f_{rO} and f_{rN} denote the oxygen and nitrogen relaxation frequencies in hertz, which can be calculated as follows:

$$f_{rO} = \frac{p_a}{p_r} \left(24 + 4.04 * 10^4 h \frac{0.02 + h}{0.391 + h} \right) \tag{4.7}$$

$$f_{rN} = \frac{p_a}{p_r} \left(\frac{T}{T_0} \right)^{\frac{-1}{2}} * \left(9 + 280h * \exp \left\{ -4.170 \left[\left(\frac{T}{T_0} \right)^{\frac{-1}{3}} - 1 \right] \right\} \right) \tag{4.8}$$

p_r and T_0 are reference pressure and temperature values, which for the purpose of this thesis and in accordance with ISO 9613-1 are assumed to be $p_r = 101.325$ kPa and $T_0 = 293.15$ K.

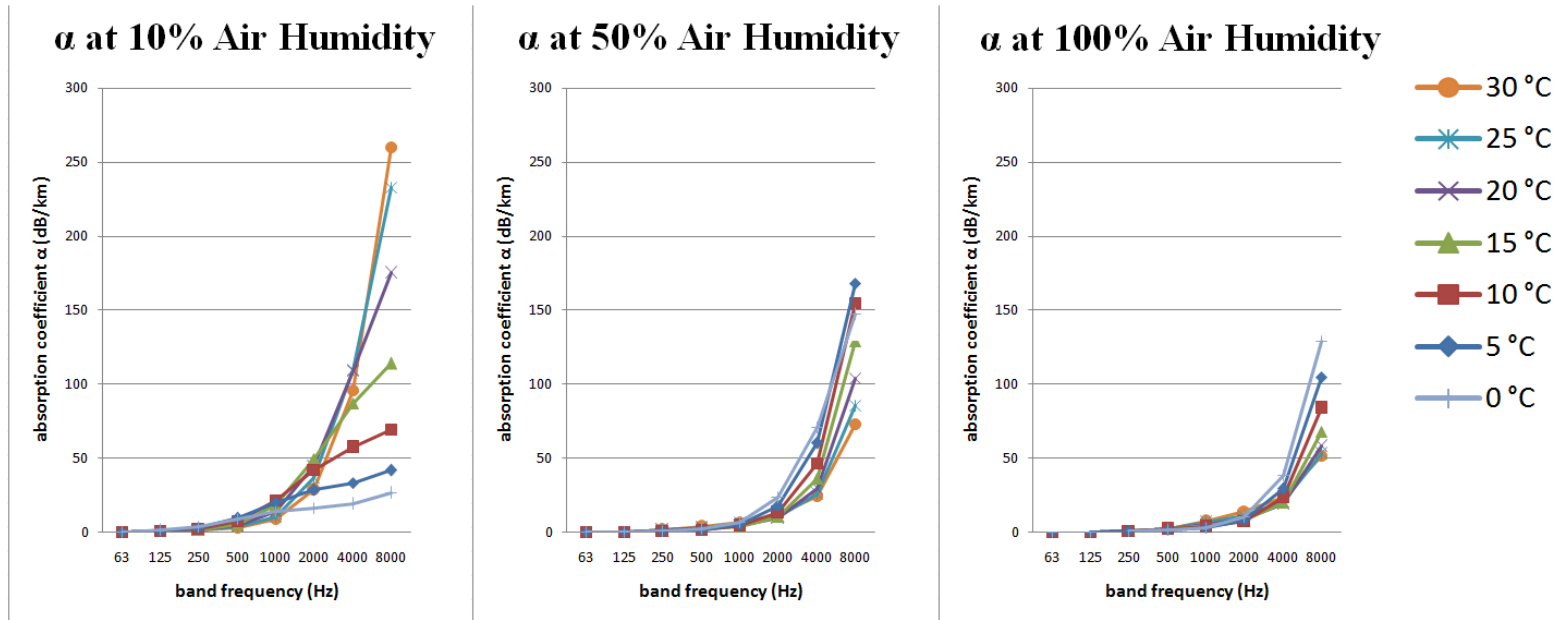


Figure 4.1: The absorption coefficient α at fixed air humidity values as a function of temperature and frequency. High frequency sounds are attenuated stronger than low frequency sounds. Note also that the influence of temperature varies strongly across different values for air humidity.

Figure 4.1 shows the air absorption coefficient as a function of temperature and frequency. Regardless of temperature, high frequency sounds are stronger attenuated than low frequency sounds. The influence of temperature is very strong and further increases with higher frequencies.

Low frequency sounds are much harder to contain than high frequency ones, due in no small part to the relationship between frequency and temperature shown in Figure 4.1. Section 4.5 further explores this issue. In practice, air humidity and temperature are usually *not* controllable factors (although consider the case of racing tracks or events in general that rely on good weather), but frequency sometimes *is*: consider for example the case of engines or exhaust pipes, which may be modified to produce higher frequency sounds.

If the frequency of a noise emission is unknown, ISO 9613-2 advocates the use of a simplified calculation method, which assumes emissions to occur at a fixed frequency of 500 hertz. Figure 4.2a shows the air absorption coefficient as a function of temperature and air humidity. At low air humidity levels, temperature very strongly affects sound propagation, with lower temperatures being more conducive than higher ones. With rising air humidity values, however, air absorption rates across various temperatures seem to conflate to values of $\approx 2 - 3$ dB/km and starting at about 30% humidity, the influence of temperature is reversed.

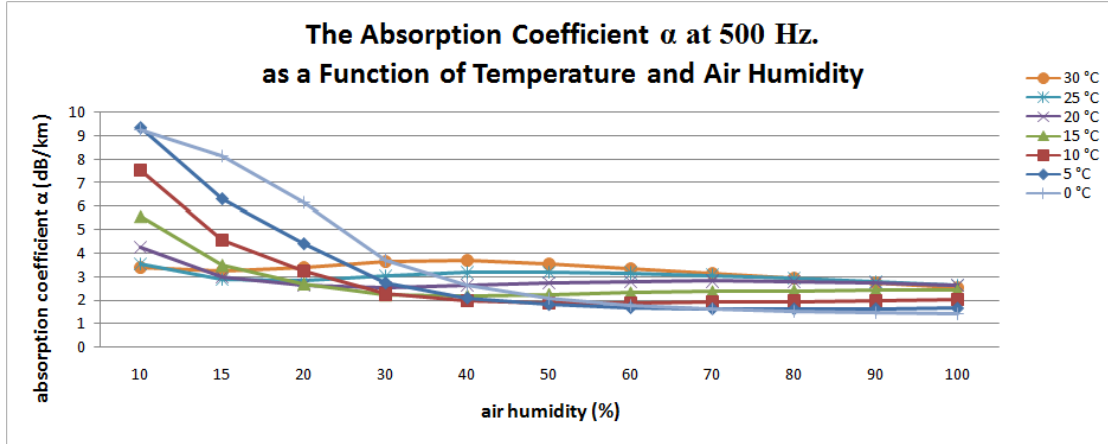
As for the calculation of these values in CUDA: while real-time calculation is certainly possible and even easy to accomplish, it is usually not necessary to do so. Temperatures and air humidity values may vary significantly in large-scale areas and over time. To accurately reflect this behaviour, a considerable amount of effort would have to be made and the usage of ISO 9613-2 would have to be abandoned in favor of physically more accurate models, which are not the subject of this thesis. For the purpose of a worst-case evaluation, however, the air absorption coefficient seems to work accurately enough (see also Chapter 6). In *Noise Simulator*, values for α are precomputed and stored as a lookup-table, which is both fast and memory-efficient. Note that ISO 9613-2 discourages interpolation and extrapolation of values for α , therefore *Noise Simulator* performs only nearest neighbor interpolation.

4.4 A_{gr} : Ground Sound Attenuation

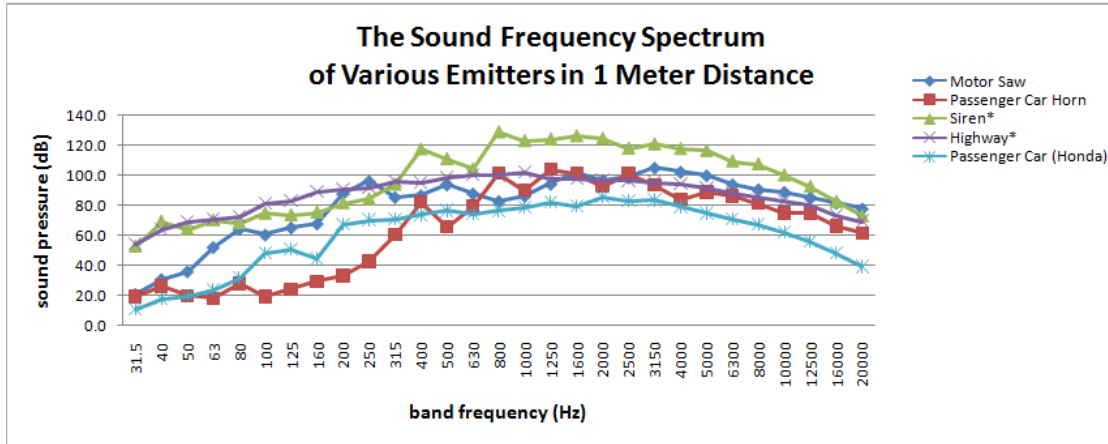
A_{gr} describes how sound is attenuated by interference with reflections of itself on the ground. It is therefore not surprising to find that this factor is a function of the type of ground and the mean propagation height of sound waves h_{mean} .

Although ISO 9613-2 describes a method to approximate this factor, Möhler et al. [24] have found that this approach is generally not suited to the problem, especially since in large-scale environments the types of ground are bound to vary. Instead, the authors advocate the use of the simplified approach also specified in ISO 9613-2, which can be used if the following is true:

1. Only A-weighted sound pressure levels are calculated
2. The ground is mostly porose or a mixture of other ground types
3. The sound is not a pure sound



(a) The absorption coefficient α for a pure tone of 500 hertz as a function of temperature and air humidity. For low values of air humidity, low temperature environments hinder sound propagation much stronger than high temperature ones, but at around 30% humidity this trend reverses.



(b) Sample sound frequencies and sound pressure levels of various emitters in 1 meter distance. Due to circumstantial constraints, entries with an asterisk (*) were measured farther away and values have been adjusted by applying Equation 4.4. For details regarding data acquisition and methodology please see Section 1.4 as well as the examples in Chapter 6.

Figure 4.2: The absorption coefficient α at 500 hertz and some sample emitters. The octave band 500 hertz is of particular interest since ISO 9613-2 assumes all emissions for which the exact frequency emissions are unknown to be in the 500 hertz band.

All of which can be assumed to be true in our case. The simplified approach is defined as:

$$A_{gr} = 4.8 - \left(\frac{2h_{mean}}{d} \right) \left[17 + \left(\frac{300}{d} \right) \right] \geq 0 \text{ dB} \quad (4.9)$$

where h_{mean} denotes the mean sound propagation height in meters and d is the euclidean distance between emitter and imitter, also in meters. The calculation of h_{mean} is outlined in ISO

Illustration of the Convex Sound Distance using the Rubber-band Method

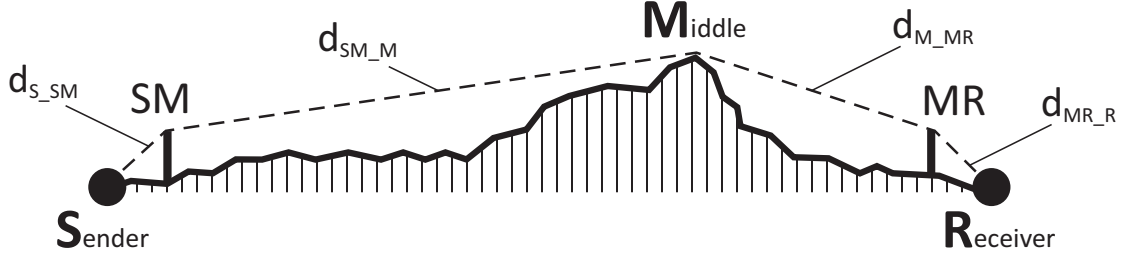


Figure 4.3: The convex sound distance between arbitrary sender (S) and receiver (R) points is the sum of all intermediate distances, i.e., $d_{S_SM} + d_{SM_M} + d_{M_MR} + d_{MR_R}$.

9613-2, but it involves solving a numerical integral, which may be a computationally expensive operation. Luckily, as can be seen in Section 4.7, the calculation of the convex sound distance d_{conv} needed for A_{bar} can be easily adapted to yield h_{mean} as a byproduct. When h_{mean} is known, the calculation of A_{gr} becomes trivial and computationally cheap.

Note that Equation 4.9 is independent of noise frequencies and does not need to be evaluated for all octave bands. It can be easily seen that Equation 4.9 converges to 4.8 dB for $\lim_{d \rightarrow \infty}$.

4.5 A_{bar} : Sound Attenuation by Barriers

A_{bar} denotes the sound attenuation by *sound barriers*. These *barriers* can be both artificial (noise screens, buildings etc.) and natural (terrain obstructions). Each of these barriers constitute a *diffraction point*, i.e., a point at which sound is being diffracted. Problematically, ISO 9613-2 does not specify which diffraction points should be chosen; it is tacitly assumed that sound is diffracted either 2 times, 1 time or not at all. However, if there are more than 2 diffraction points, ISO 9613-2 does *not* define which diffraction points should be used (it is merely stated that the two ‘most efficient’ barriers should be used). To remedy this shortcoming, we can make use of the findings in the German noise guideline ‘Schall 03’ [6], where the use of 3 diffraction points is advocated and it is explicitly stated how they should be chosen:

1. The diffraction point with highest altitude between **Sender** and **Receiver**, **M**
2. The diffraction point nearest to **S**, which will henceforth be called **SM**
3. The diffraction point nearest to **R**, which will henceforth be called **MR**

These points are also illustrated in Figure 4.3. *Noise Simulator* follows the suggestions of Möhler et al. [24] and incorporates their findings in the formulas of ISO 9613-2, the latter of which defining formulas for the calculation of single and double sound diffractions. Equations 4.13, 4.14 and 4.15 show the old formulas and how they are adapted to accommodate 3

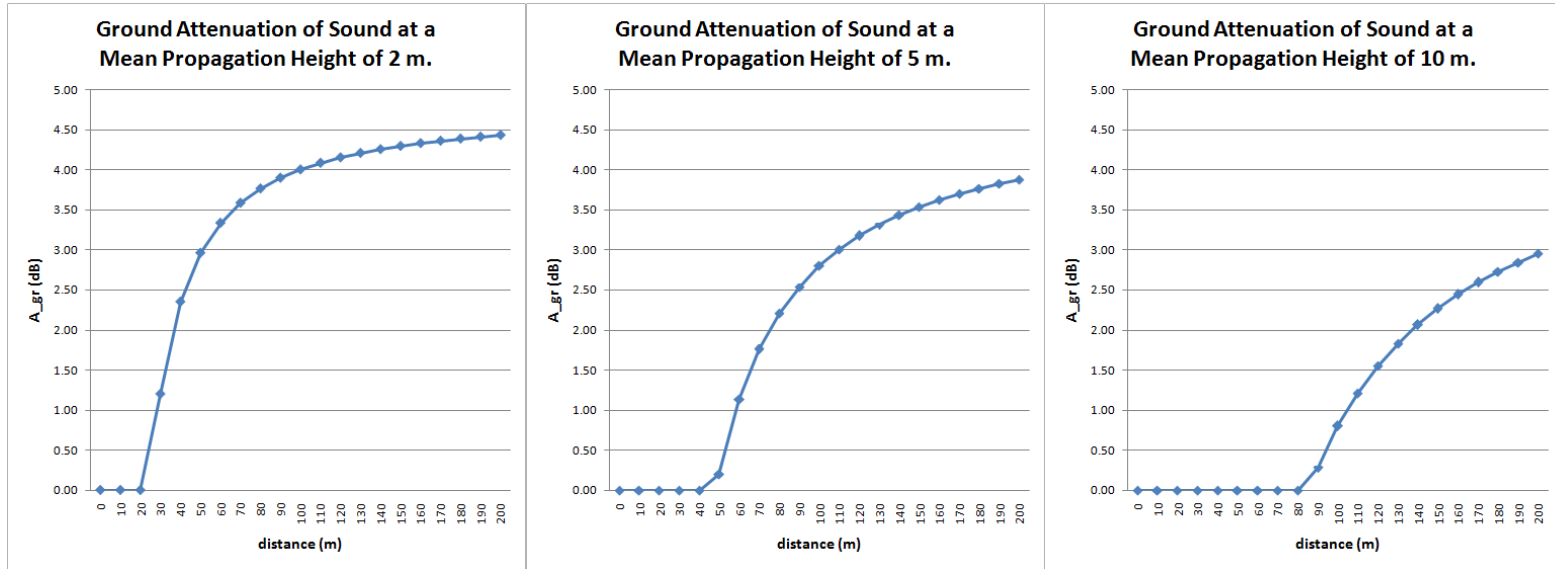


Figure 4.4: The ground attenuation of sound as a function of the distance between emitter and immitter, plotted for different mean propagation heights.

diffractions. Note that *Noise Simulator* assumes all sound diffractions to occur at vertical sound barriers, which is usually the case in outdoor areas but may prove to be incorrect in the vicinity of cities and villages.

Figure 4.3 illustrates the use of the rubber-band method advocated by Möhler et al. [24] to determine how sound propagates from a given **Sender** to a **Receiver** point across different obstacles. Henceforth this distance shall be called *Convex Sound Distance* or simply *Convex Distance*, d_{conv} . We define it as the sum of the individual euclidean noise propagation distances between **S** and **R**:

$$d_{conv} = d_{S_SM} + d_{SM_M} + d_{M_MR} + d_{MR_R} \quad (4.10)$$

According to ISO 9613-2, A_{bar} is calculated as follows:

$$A_{bar} = D_Z - A_{gr} > 0 \quad (4.11)$$

where D_Z is the *barrier attenuation factor*. Note that D_Z already incorporates the ground attenuation factor A_{gr} , thus we have to subtract it so as not to sum up A_{gr} twice in the overall Equation 4.2. For a given wavelength λ , D_Z is calculated as follows:

$$D_Z = 10 * \log \left[3 + \left(\frac{C_2}{\lambda} \right) C_3 z K_{met} \right] dB \quad (4.12)$$

Where K_{met} is the meteorological correction factor as detailed in Equation 4.16 and z is the difference between the diffracted sound length d_{conv} and the euclidean distance d between **S** and **R** (see Equations 4.15 and 4.14). According to Möhler et al. [24], C_2 is 40. For single sound diffraction, C_3 is 1. Else it is calculated as shown in Equation 4.13 (the left column showing standard ISO 9613-2 formulas, the right column depicting the adapted formulas in *Noise Simulator*):

$$C_{3,double} = \frac{1 + \left(\frac{5\lambda}{e} \right)^2}{\left(\frac{1}{3} \right) + \left(\frac{5\lambda}{e} \right)^2} \Rightarrow C_{3,double} = \frac{1 + \left(\frac{5\lambda}{d_{SM_M} + d_{M_MR}} \right)^2}{\left(\frac{1}{3} \right) + \left(\frac{5\lambda}{d_{SM_M} + d_{M_MR}} \right)^2} \quad (4.13)$$

e is the distance between the diffraction points.

In the original formula of ISO 9613-2, the calculation of z depends on whether noise is diffracted once (z_{single}) or twice (z_{double}). Let d_{SS} and d_{SR} encode the distances from the source to the first diffraction edge and from the last diffraction edge to the receiver, respectively. a is the distance between source and receiver, parallel to the barrier edge. d_{conv} is the diffracted sound length and d is the euclidean distance between **S** and **R**. z_{single} and z_{double} are then calculated as follows (the right column shows the adapted formulas in *Noise Simulator*):

$$z_{single} = \left[(d_{SS} + d_{SR})^2 + a^2 \right]^{\frac{1}{2}} - d \quad \Rightarrow \quad z_{single} = d_{conv} - d \quad (4.14)$$

$$z_{double} = \left[(d_{SS} + d_{SR} + e)^2 + a^2 \right]^{\frac{1}{2}} - d \quad \Rightarrow \quad z_{double} = d_{conv} - d \quad (4.15)$$

Finally, to calculate Equation 4.12 we need to compute the meteorological correction factor K_{met} , which is defined in ISO 9613-2 as follows:

$$K_{met} = \exp \left[-\frac{1}{2000} \sqrt{\frac{d_{ss}d_{sr}d}{2z}} \right] \quad \Rightarrow \quad \exp \left[-\frac{1}{2000} \sqrt{\frac{d_{S_SM}d_{MR_R}d}{2z}} \right] \text{ for } z > 0 \quad (4.16)$$

$$K_{met} = 1 \quad \text{for } z \leq 0$$

where d_{ss} and d_{sr} are the distances between **S** and **R** to their respective diffraction points **SM** and **MR** in meters.

We now have to calculate these terms to obtain the sound attenuation by barriers A_{bar} . It is easy to see that the computation of d_{conv} is *crucial* for this calculation; it will therefore be discussed separately in Section 4.7. Having calculated d_{conv} , evaluation of the formulas above to get the correct value for A_{bar} is computationally inexpensive and can be done in CUDA in a straightforward way.

4.6 A_{misc} : Miscellaneous Sound Attenuation

The final term of Equation 4.2, A_{misc} , encodes how sound is attenuated by vegetation A_{fol} , industrial facilities A_{site} and by housing A_{house} :

$$A_{misc} = A_{fol} + A_{site} + A_{house} \quad (4.17)$$

In ISO 9613-2, both A_{fol} and A_{site} are specified in terms of tabular values for specific octave bands. Due to the complex noise diffraction and reflection situation in a dense housing environment, we will disregard A_{house} as a separate term and will instead calculate it using the barrier attenuation A_{bar} . This will be done for A_{fol} and A_{site} as well, since real world measurements are more consistent with A_{bar} rather than A_{misc} , especially for distances > 100 m. Also, ISO 9613-2 does not explicitly specify how one should compute the mixture of A_{bar} and A_{fol} in the presence of e.g., wooded hills. We will therefore consider vegetation, industrial facilities and housing as *modifications* of the underlying terrain, effectively raising the ground level by their respective heights. This also makes the computation easier and does not introduce additional performance penalties. It does, however, introduce small simulation errors which are largest in the vicinity of the aforementioned areas. For most immission points (i.e., points not *immediately* adjacent to forests, industrial sites or housing areas), this limitation does not incur a significant loss of accuracy (see Chapter 6). It should be noted that ISO 9613-2 states that in the presence of complex diffraction and reflection environments, it is usually advisable to conduct in-situ measurements rather than to rely on simulation results.

4.7 Convex Distance Calculation in CUDA

The core part of the ISO 9613-2 driven noise attenuation computation is the convex distance calculation. There are four cases that we have to deal with, namely

1. Direct Noise Propagation
2. Single Diffraction Noise Propagation
3. Double Diffraction Noise Propagation
4. Triple Diffraction Noise Propagation

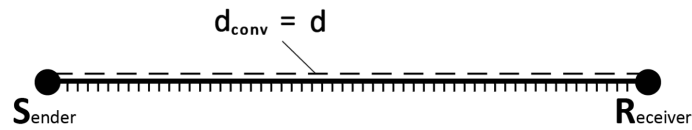
Triple Diffraction Noise Propagation also covers all cases of more than 3 diffractions, in which case we follow the suggestion of Möhler et al. [24] as outlined in Section 4.5 to choose a subset of 3 diffraction points. Figure 4.5 illustrates these 4 cases. As mentioned before, CUDA performance is highest when all threads agree on a single instruction flow. Therefore we will not differentiate between these cases, but treat all as special cases of a *Triple Diffraction Noise Propagation*.

Transforming the Problem of Convex Sound Distance Calculation to 2D While there are many ways to calculate the distance in 3-dimensional domains, most of them are unsuitable to the problem at hand, mostly because they are not fast or accurate enough (Raycasting etc.). We can, however, make use of an interesting property of the sound distance we are trying to compute, namely that the noise diffraction points along with Sender and Receiver point make up a set of points $\{S, MS, M, MR, R\}$ that is a subset of the 2-dimensional *convex hull* between S and R. Apart from S and R, these points are unknown to us but can be easily approximated (as outlined below). The basic strategy to calculate the convex distance d_{conv} between an arbitrary sender point **S** and receiver point **R** thus becomes:

1. Subsample the space between **S** and **R** and find the point of highest altitude, M .
 - If M does not exist (there is no obstruction between **S** and **R**), set M to S .
2. Sum up the noise propagation heights of all subsamples and divide by the number of subsamples to calculate h_{mean} .
3. Use M to divide the set of subsample points into a *left* and *right* set.
4. Use the QuickHull algorithm to calculate SM in the left set and MR in the right set.
 - If SM does not exist, set SM to **S**. If MR does not exist, set MR to **R**.
5. Calculate the euclidean distances between the points **S**, SM , M , MR and **R** to calculate d_{conv} .

Listing 4.1 shows the code for this process. Since this code needs to be executed by all threads simultaneously, one needs to make sure that there is enough memory available to subsample the terrain (see Listing 4.1, line 8).

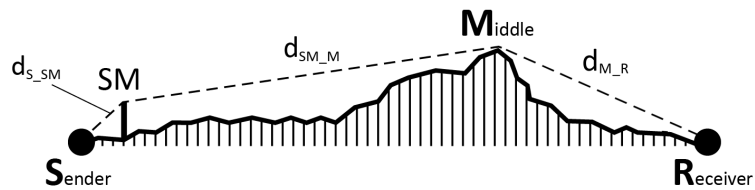
Unobstructed Noise Propagation



Single Diffraction Noise Propagation



Double Diffraction Noise Propagation



Triple Diffraction Noise Propagation

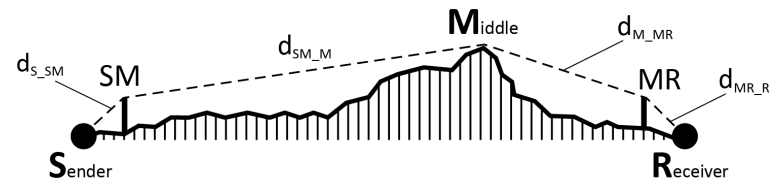


Figure 4.5: Sound propagation across arbitrary terrains with a maximum of 3 diffractions. Note that all cases can be seen as special instances of triple diffraction sound propagation (by setting **SM** and w.l.o.g. **M** = **S** and/or **MR** = **R**).

input : The sender point **S** and receiver point **R** in 3D coordinates (Y is up)
output: d_{conv} , h_{mean} and the positions of SM , M and MR in 3D

```

1 h_mean = 0;
2 MS = S;
3 M = S;
4 MR = R;
5 step_x = (R.x - S.x)/subsample_rate;
6 step_z = (R.z - S.z)/subsample_rate;
7 normalized_vector = norm(R.xyz-S.xyz);

  // Allocate some memory to store the subsampled points:
8 sample_cache[subsample_rate] ;

9 for i ← 0 to subsample_rate do
10   current_point_interpolated = S + (i/subsample_rate)*normalized_vector;
11   current_point_lookup = tex2D(terrain_grid, current_point_interpolated.x,
   current_point_interpolated.z);
12   sample_cache[i] = current_point_lookup;
13   h_mean += current_point_lookup.y;
   // current sample higher than line S to R:
14   if current_point_lookup.y > current_point_interpolated.y then
15     if current_point_lookup.y > M.height then
16       M = current_point_lookup; split = i;
17     end
18   end
19 end
20 h_mean /= sample_rate;

  // We use M to split the samples into a left and right
  subset and calculate MS and MR:
21 MS = quick_hull(sample_cache, 0, split, R, S);
22 MR = quick_hull(sample_cache, split+1, sample_rate, R, S);
  // The convex distance is the sum of all euclidean
  distances between S and R:
23 d_conv = length(MS.xyz-S.xyz) + length(M.xyz-MS.xyz) + length(MR.xyz-M.xyz) +
  length(R.xyz-MR.xyz);

```

Algorithm 4.1: Convex distance calculation of points in a 3D grid.

4.8 Noise Calculation in CUDA

As we now have specified how to calculate all the terms necessary to compute the noise attenuation as outlined in equation 4.1, we can proceed by putting these values to use. We first have to *precompute* the convex distances between all *relevant* points **S** and **R**. These point pairs are the cartesian $EP \times G$ of all emitter points EP and the terrain grid G . Therefore we set up a CUDA thread grid of dimensionality equal to $dim(G)$ and let each thread calculate the convex

distances from one point in G to all emitter points EP . The results are stored in a lookup table (i.e., in an array). This is done only once per scene and every time the terrain changes (due to the insertion of buildings, walls, forests etc.). Performance results of this precomputation are discussed in Section 3.5.

Having computed the convex distances between the relevant point pairs \mathbf{S} and \mathbf{R} (which also yields h_{mean}) we can now proceed to calculate the equivalent continuous A-weighted downwind (5 m/s) sound pressure level $L_{AT}(DW)$ as specified in equation 4.1. Once again we create a thread grid of dimensionality equal to $dim(G)$. Each thread now needs to sum up the contributions of each individual noise emitter over all 8 octave bands. The result is a noise imission value for each point in G . Listing 4.2 shows the pseudocode that each thread has to execute. This calculation has to be done for every frame, performance results are once again discussed in Section 3.5.

Choosing Imitter and Emitter Points

While we *do* attempt to generate noise imission values for each point in the terrain grid, only a small number of samples in this grid are actually *emitting* noise. These are the set of static point emissions and dynamic point emissions (i.e., vehicles), the latter moving along specified streets in the terrain. This allows us to generate a set of emission points EP that contains all static point emissions and all *possible* dynamic point emissions by sampling the noise emitting streets. In other words, we discretize the problem along the time axis and assign each vehicle an index in the emission point array. This index will be updated in every frame according to the vehicles' velocities, which is a computationally cheap operation. Figure 4.6 shows a sample session in *Noise Simulator* and the emitter sampling performed. Due to memory constraints (see beginning of this chapter) one can not sample streets with an arbitrarily high sample rate. This is usually not a big problem, since 128 samples are often more than enough to achieve a sufficient level of accuracy. However, the more and longer the streets in question, the less samples can be used for each street, thus effectively limiting the complexity of sessions in *Noise Simulator* in proportion to the amount of GPU memory available.

Long Term (one hour) Noise Calculation in CUDA

While the previous sections showed how real-time calculation of noise values may be performed, this section will deal with the problem of calculating the equivalent long term (one hour), A-weighted averaged sound pressure $L_{A,eq}$ for *individual* imission points. Thus we have to organize the thread grid differently and create one thread for every second in the timeframe. Each thread will then subdivide its timeframe into 10 units of 100 ms and perform the noise attenuation calculation as outlined in the previous sections. All threads save their average, minimum and maximum noise pressures. Calculating the overall $L_{A,eq}$ is done by the CPU in a single loop over all 3600 individual simulation results. This calculation does not need to be executed once per frame but only every time the simulation parameters change. Note that individual results need to be stored in order to use them later to generate 3D imission graphs.

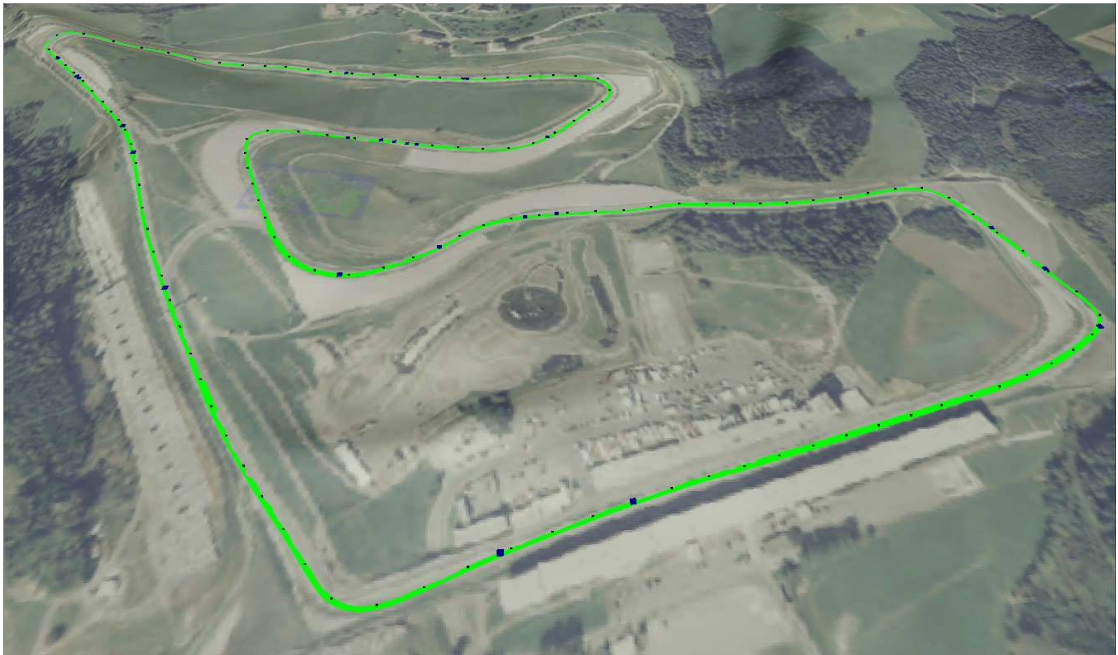


Figure 4.6: Screenshot of a *Noise Simulator* session showing the Red Bull racing track in Styria. Black dots mark possible emitter positions, the larger blue dots are vehicles moving along the track.

Real-Time Noise Visualization

Simulating noise propagation in large-scale areas (as described in Chapter 4) is only part of the problem; results also need to be *visualized* appropriately to help in the decision making process. In this thesis, we focussed on the following problem statements and created visualizations designed specifically to communicate:

1. Simulation of (building) projects *before* and *after* implementation.
2. Where can new emitters be placed in a given noise environment s.t. they are least disturbing?
3. If the estimated noise level is too high, where should noise screens be placed and how high should they be?
4. Identify problematic regions to allow both project solicitors and people in the neighborhood to reach an agreement outside court.
5. Estimate the changes to the noise environment when altering the terrain by inserting and/or deleting forest regions and buildings.

For this purpose, three noise visualizations have been developed that may be used separately or in conjunction with each other. These visualizations consist of:

- *2D Noise Overlays*, a very common and also very useful technique to communicate the overall noise propagation and to identify problematic regions (Section 5.1)
- *Noise Propagation Terrain Slices*, which are traditional terrain slices overlayed with noise propagation lines (Section 5.2)
- *3D Immission Graphs*, a new technique that uses long-term simulation results to communicate the origin of the incoming noise at certain points (Section 5.3)

input : Environment variables (Temperature, Air Humidity) and, for all valid point pairs \mathbf{S} and \mathbf{R} , \mathbf{S} , \mathbf{MS} , \mathbf{M} , \mathbf{MR} , \mathbf{R} in 3D coordinates (Y is up) as well as h_{mean}

output: The equivalent continuous A-weighted downwind (5 m/s) sound pressure level, $L_{AT}(DW)$

```

// calculation parameters for the active thread:
1 S = lookup(S_precalculated, thread_index);
2 R = lookup(R_precalculated, thread_index);
3 MS = lookup(MS_precalculated, thread_index);
4 M = lookup(M_precalculated, thread_index);
5 MR = lookup(MR_precalculated, thread_index);
6 h_mean = lookup(h_mean_precalculated, thread_index);
7 total_noise = 0;
// add up the influence of all emitters:
8 for i ← 0 to number_of_emitters-1 do
9     sum_emission_bands = 0;
10    distance = length(R.xyz - S.xyz);
11    A_div = (20.0f * log10(distance));
12    A_gr = 4.8f - ((2.0f * h_mean)/distance)* (17.0f + (300.0f / distance)); A_gr =
        max(A_gr, 0.0f);
// For all octave bands j:
13    for j ← 0 to 7 do
14        current_wavelength = lookup(wavelength_table, temperature, air_humidity,
            octave_band[j]);
15        A_atm = lookup(A_atm_table, temperature, air_humidity, current_wavelength);
16        A_bar = calculate_A_bar();
17        A_misc = calculate_A_misc();
18        A_total = A_div + A_atm + A_gr + A_bar + A_misc;
19        L_w = emitters[i].emission_values[j];
20        L_fT = L_w - A_total; L_fT = max(L_fT, 0.0f); sum_emission_bands +=
            powf(10.0f, 0.1f * (L_fT + a_weights[j]));
21    end
22    sum_emission_bands = max(sum_emission_bands, 0.0f); total_noise +=
        sum_emission_bands;
23 end
// The current thread saves the calculation result to an
    array in global memory:
24 noise_values[thread_index] = 10.0f * log10(total_noise);

```

Algorithm 4.2: Noise attenuation in CUDA.

In the following sections the implementations of these visualizations will be discussed separately. For visualization results we refer to Chapter 7.

5.1 2D Noise Overlays

The first and most basic visualization is at the same time the most intuitive technique. For every point in the terrain grid, Noise Overlays assign a color depending on the simulated noise value and the specified transfer function. The result can be interpreted as a 2D texture which can be projected on the terrain to yield the visualizations shown in Figures 5.1 and 5.2. *Noise Simulator* allows for transparency values to be seamlessly adjusted. This visualization is especially helpful to communicate the overall impact of noise emitters and to identify problematic regions. For example in Figures 5.1 and 5.2 we can immediately see:

- The racing track is situated in a basin, bordered by hills to the west and east and by a mountainous region to the north. The south of the basin, however, is open and noise ‘seeps’ out in this direction.
- The valley in the east is shielded very effectively from the racing track.
- The village southeast of the racing track may be subject to severe noise pollution, despite the hill separating it from the track.

To further analyze the situation, additional visualizations may be necessary. The combination of visualizations to analyze complex noise simulations is further discussed in Chapter 7.

Noise Overlays are a natural result of the calculation along terrain grids, discussed in Section 4.1. We create a 2D texture of these values (*Noise Simulator* uses OpenGL), perform interpolation for the values in-between sample points, define a transfer function to assign colors to noise pressure levels and finally project the resulting colors on the 2D or 3D terrain using projective texture mapping as proposed by Mark Segal et al. [33].

5.2 Noise Propagation Terrain Slices

Sometimes Noise Overlays may contain areas that are counter-intuitive in the sense that the terrain itself can not immediately explain certain parts of the overlay. Figure 5.3 shows a scene with a highway and the accompanying Noise Overlay. Note the region of relatively high sound pressure to the south-east of the highway: Even though the region in Figure 5.3a is significantly farther away from the sound emission than the area in Figure 5.3b, the sound pressure is still higher. A simple 3D rendering of the terrain does not immediately explain this behavior. To further explore the 3D data set, we can make use volume slicing, a well-known visualization technique. Volume slicing dissects 3D data by showing only one particular 2D data slice at a time. By moving the data slice across different depth values, the user can see and understand subtle data changes and/or irregularities.

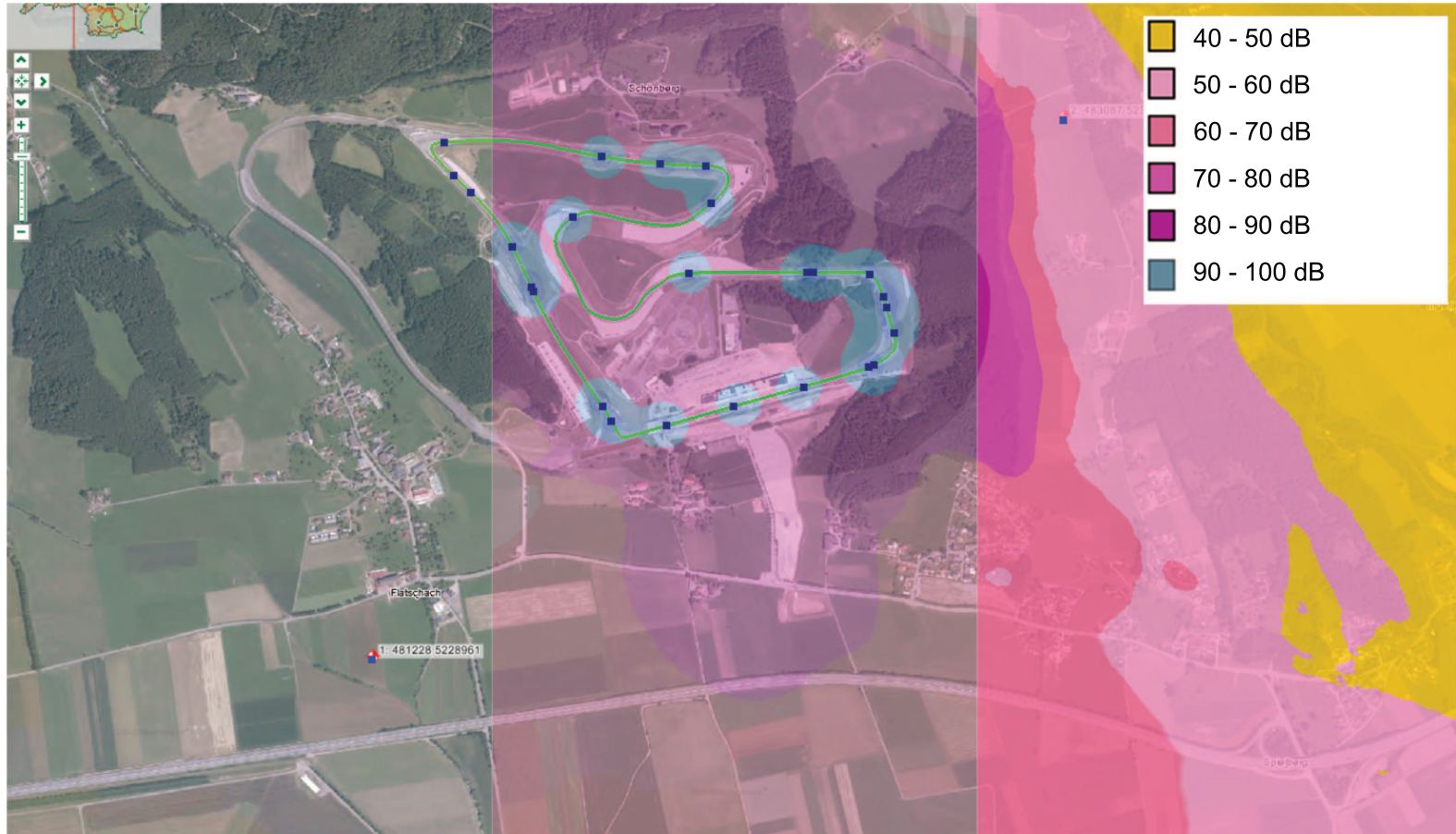


Figure 5.1: Noise Overlay on top of a 2D map with, from left to right, increasing alpha value.

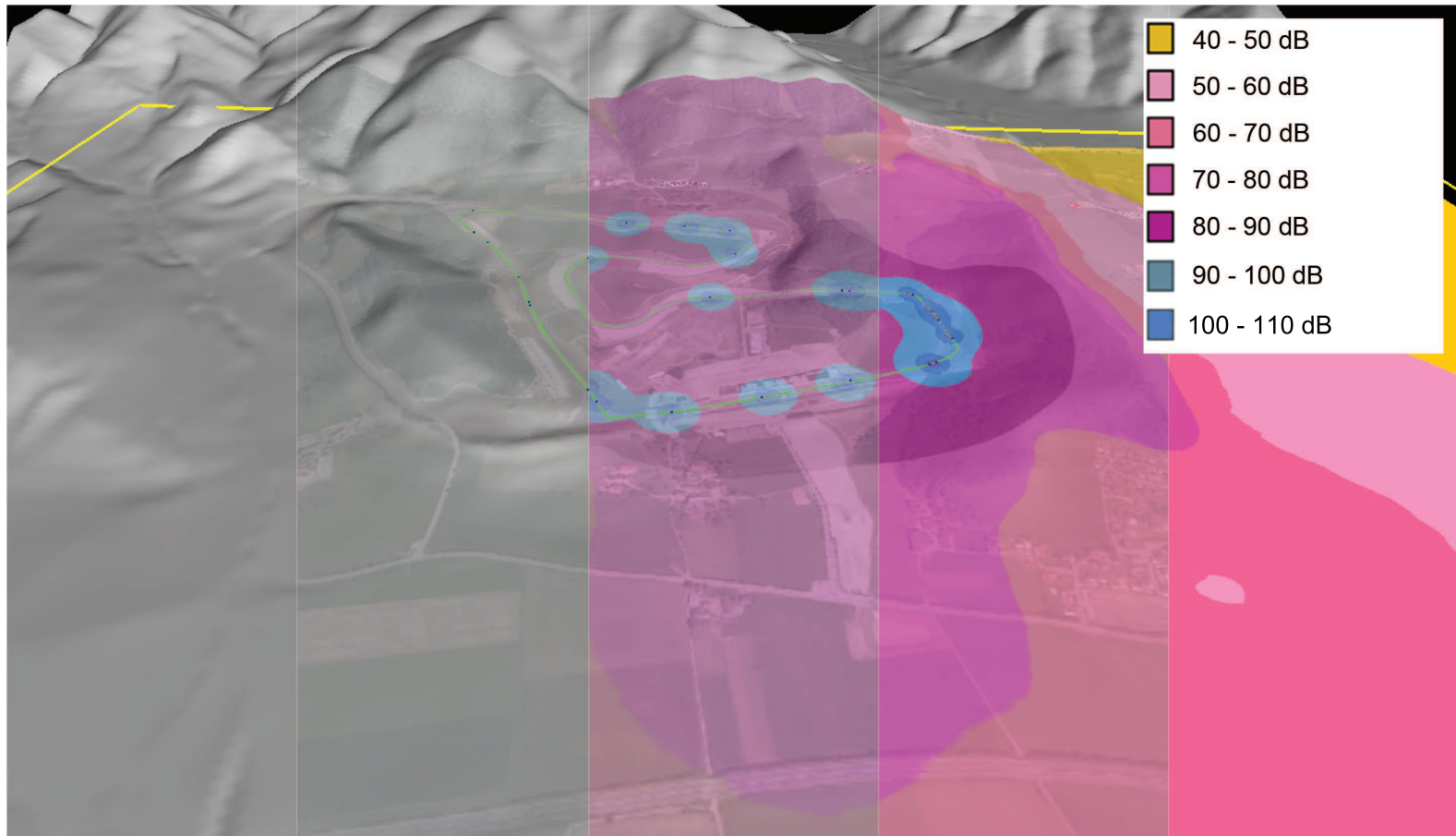


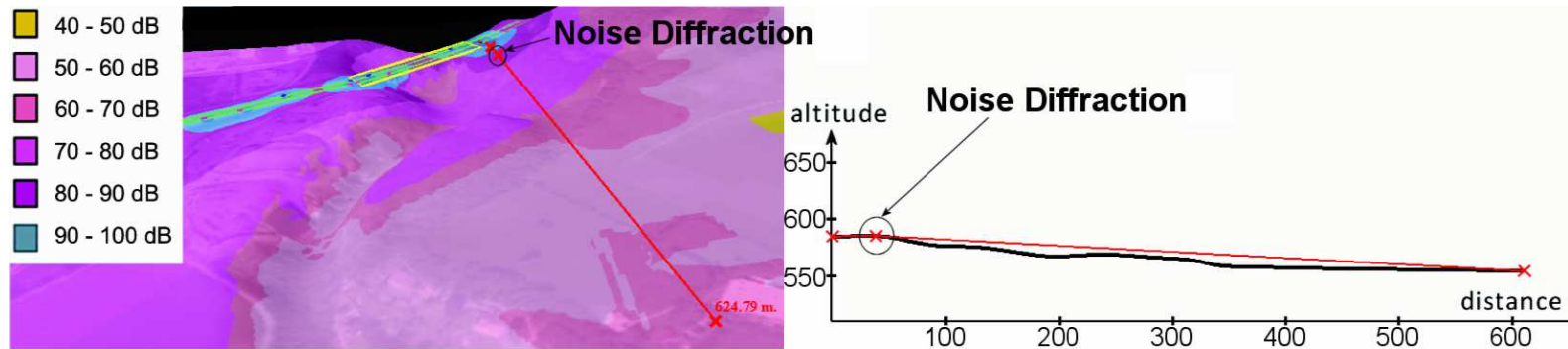
Figure 5.2: 2D Noise Overlay projected on a 3D terrain with different transparency values for the terrain and the overlay.

We can adapt this technique and modify it to provide additional information for the problem of noise propagation. We are specifically interested into *how* noise travels across a specific terrain slice. Noise propagation in *Noise Simulator*, as was previously shown in Chapter 4, is characterized by a set of five attenuation factors (see Equation 4.2)), namely A_{div} (distance attenuation), A_{atm} (atmospheric attenuation), A_{gr} (ground attenuation) as well as A_{bar} and A_{misc} (both of which interpreted as barrier attenuations in *Noise Simulator*). Of these factors, only A_{bar} and A_{misc} can hold additional information for terrain slices, since A_{div} , A_{atm} and A_{gr} are symmetrical in all directions. The barrier attenuation factor is characterized by the set of diffraction points between the sender and receiver points, as illustrated in Figure 4.3. Since the modified ISO 9613-2 formulas of *Noise Simulator* are symmetrical with respect to sender and receiver points, i.e., noise propagation from S to R is identical to noise propagation from R to S, we do not need to make additional assumptions and can easily overlay the volume slice with the calculated noise propagation path. We call the resulting volume slices *Noise Propagation Terrain Slices*.

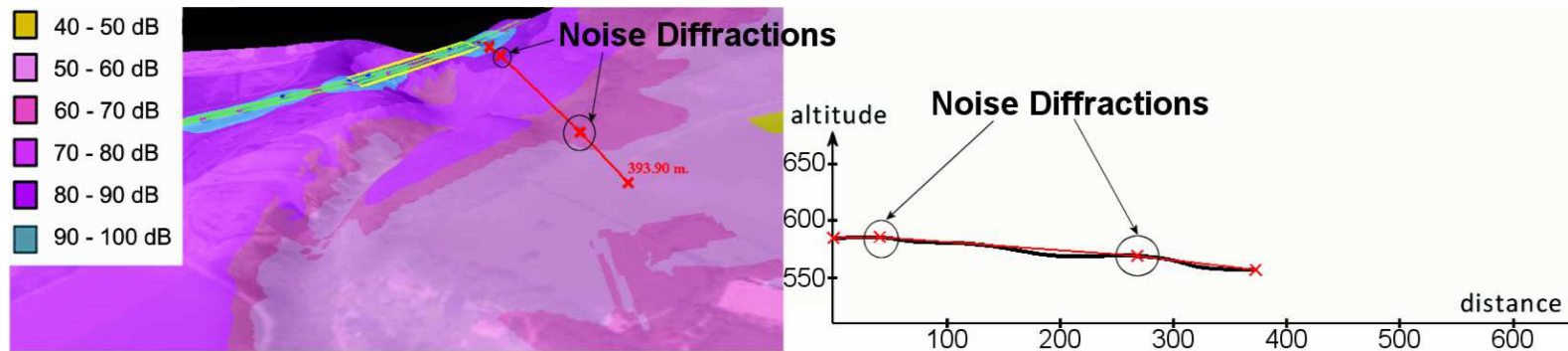
We return to the problem of the irregular patch of high sound pressure to the south-east of the highway in Figure 5.3. Using noise propagation terrain slices as illustrated in Figures 5.3a and 5.3b, we can now see that sound is diffracted only *once* in the irregular patch (Figure 5.3a), but *twice* in the areas surrounding the irregular patch (Figure 5.3b). Small as the second diffraction may be, it can still have a significant impact on noise propagation.

Terrain Propagation Slices not only show how sound propagates along the given terrain between any two points, but may also help when trying to find appropriate locations for noise screens, since terrain elevation may very well factor in these decisions. Note however, that in the sample of Figure 5.3 we can not make *certain* that the observed pattern is actually a result of the described noise diffraction differences between Figure 5.3a and Figure 5.3b. Choosing the point of the emitting street that is *closest* to the point of interest is usually a good way to guess the part of the noise emitter that has the greatest impact on the immission point, but it is *not* necessarily true. This problem can be solved by using 3D Immission Graphs, which are explained in the next section of this chapter.

The calculation of Noise Propagation Terrain Slices is, again, a natural by-product of the sound calculations in Chapter 4. Specifically, we reuse the terrain samples and the intermediate points SM , M and MR generated during the convex sound distance calculations (see Listing 4.1) to draw the slice. Since it is not feasible to actually reuse values calculated by the GPU (they are only valid for certain pairs of points anyway), and we are only interested in a single terrain slice at a time, we can let the CPU take care of the subsampling without a noticeable performance hit. Further results and evaluations are shown in Chapter 7.



(a) The terrain slice shows that in the patch of high sound pressure to the south-east of the highway sound is only diffracted *once*.



(b) This area experiences lower sound pressure even though it is closer to the highway than the area in Figure 5.3a. The terrain slice shows that sound is diffracted *twice* along the illustrated path.

Figure 5.3: Highway noise propagation visualized by a Noise Overlay (lighter colors denote lower sound pressures). The right images show sample Noise Propagation Terrain Slices drawn between the endpoints of the red lines to help explain the patch of higher sound pressure in the south-east.

5.3 3D Immission Graphs

As we have seen in the previous examples, Noise Overlays and Noise Propagation Terrain Slices are very valuable tools to communicate the overall noise situation and one specific noise propagation, respectively. However, they may fail to explain the *origin* of the sound pressure when regarding a particular point of interest. For the remainder of this thesis we will use the term immission point (IP) to refer to a special receiver point where actual noise measurements have been taken. The set of IPs constitutes the set of test points for the simulation (see Chapter 6). When regarding a specific IP, the most important questions are usually: where does the incoming noise originate from and where should noise screens therefore be placed? Noise Overlays simply convey that certain points *need* to be protected, but not where this protection should be placed. Terrain Slices only show single propagation paths, but it is not clear to which degree individual emitters contribute to the overall noise immission.

We therefore created a new visualization specifically targeted to address this problem. Since we want to convey how much of the noise emitted by a certain emission point (EP) arrives at a given receiver point, we will need to simulate the noise propagation for all EPs individually, i.e., before they are summed up. To complicate things, since we are simulating *moving* point sound sources, we can not simply evaluate the noise propagation for all EPs at any given time, but we will have to perform averaging across time. In other words, we have to sum up and average individual noise propagation *snapshots* s.t. for every possible emitter point we can calculate the average noise emission.

Fortunately, we have already performed a long-term simulation when calculating the $L_{A,eq}$ values in the previous sections. We can easily adapt this simulation to yield the desired values. Specifically, before adding up the contributions of every individual EP at any given time t , we let each thread save *where* every individual EP is located at time t and *how large* their individual noise pressure is for the given IP. To illustrate this method, we can consider the simple case of a single street and IP as in Figure 5.4. For this example we disregard octave band emissions. In a real-world example, the illustrated approach needs to be repeated for every octave band separately. Two vehicles with different velocities and emissions travel along a street which has been sampled 6 times. In this example every thread has to calculate 4 time steps, and the illustrated thread starts at time $t = 0$ and a starting configuration as shown in Figure 5.4. The sample thread then calculates the noise propagation for each vehicle separately and stores these values in a suitable data structure. It then advances the time, recalculates the positions of the vehicles and repeats the above noise propagation calculation. If two or more vehicles are located at the same street sample point at any given time - as seen in Figure 5.4, $t = 3$ - only the sum of the accumulated noise emission is saved.

After letting each thread calculate a part of the one-hour simulation, one ends up with a large table of values for each street sample point. Averaging these values for each sample then yields a good estimate for the average noise emission at every point of the street relating to a specific IP. Note that in simulating and averaging noise emission values, as described above, we tacitly simplify our simulation by assuming a constant vehicle density across the whole length of the street. While this may be true in some cases (e.g., highways), it also may introduce an additional error in other situations. To properly handle this problem we would need to implement

a sophisticated traffic simulation, which is outside the scope of this thesis.

Having calculated the IP-specific average noise immission for each emitter point, we are now able to visualize the data. Arrow plots and quiver plots - in fact any kind of 2D plots drawn *on* the terrain are unsuitable for our purposes, for the following reasons:

- Noise Overlays already ascribe semantic values to the terrain by colorizing it, additional information on the same plane makes it harder to understand the visualization
- Traditionally, visualizations with arrows or glyphs often make use of larger icons when the signified values are larger. However, in our case, large values often occur where the distance to the EPs are smallest; in other words, where there is not enough space to draw sufficiently large symbols

Instead, we constructed a 3D Graph on top of the noise emitters in the scene with color and size proportional to the calculated values, as illustrated in Figure 5.5. From a technical point of view, we create 3D square prisms centered around the emission sample points and placed orthogonal to the terrain, on top of the street sample points (which are essentially the set of EPs). We employ the same transfer function values used in the Noise Overlay to color these prisms. To avoid biasing in the presence of multiple streets and different altitudes along these streets, we add an offset equal to the highest occurring altitude to all prisms. Single emission points are visualized by single prisms. To emphasize the section of the street that each prism covers, we draw black outlines. Similar to noise overlays, these graphs can be made (semi)transparent so as not to obstruct visibility of the street and the vehicles travelling along it.

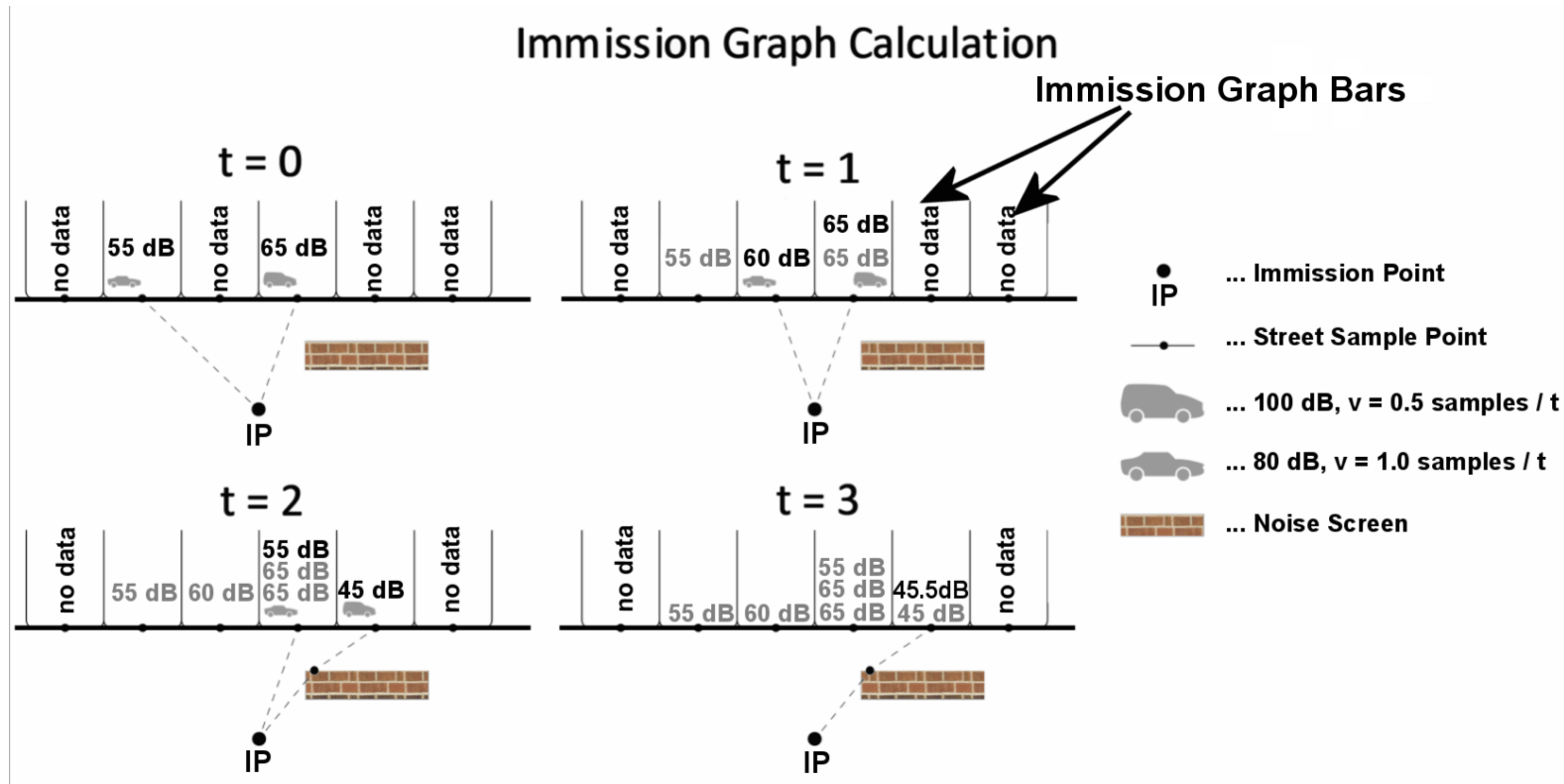


Figure 5.4: Construction of a 3D Immission Graph for a specific street. At every time step t , we calculate the current traffic situation and simulate the noise propagation from each street sample point to the given immission point separately. Values for the current time step are printed in black, past values are shown in grey. All values are accumulated and averaged across all threads. This yields a measure of how noisy each sample point is for the given immission point, averaged over time. These values are subsequently visualized using 3D square prisms centered around the street sample points. Note that all emission and immission values have been chosen for illustrative purposes only and therefore do not reflect actual simulation results.

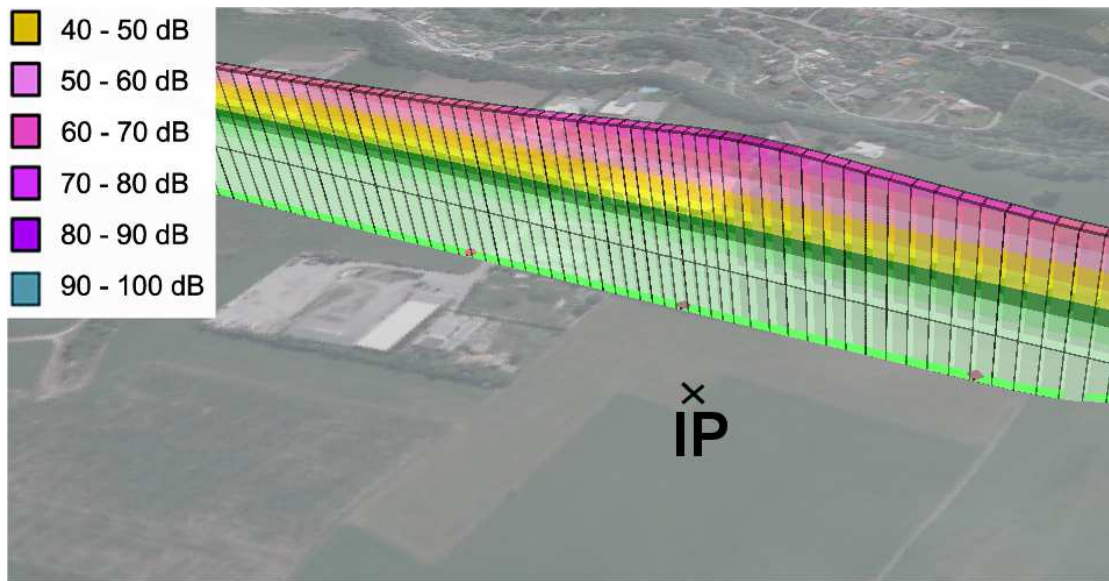
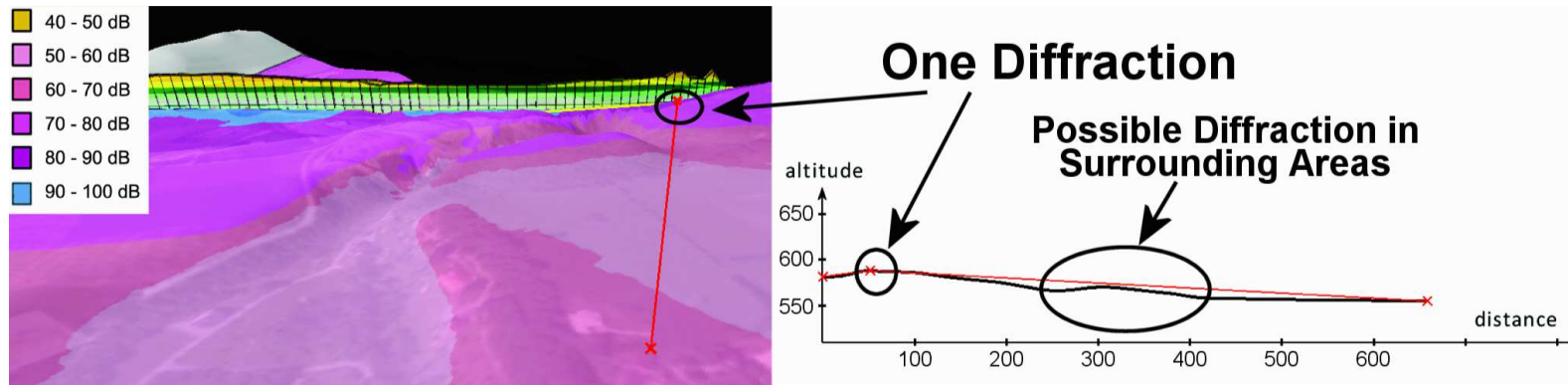
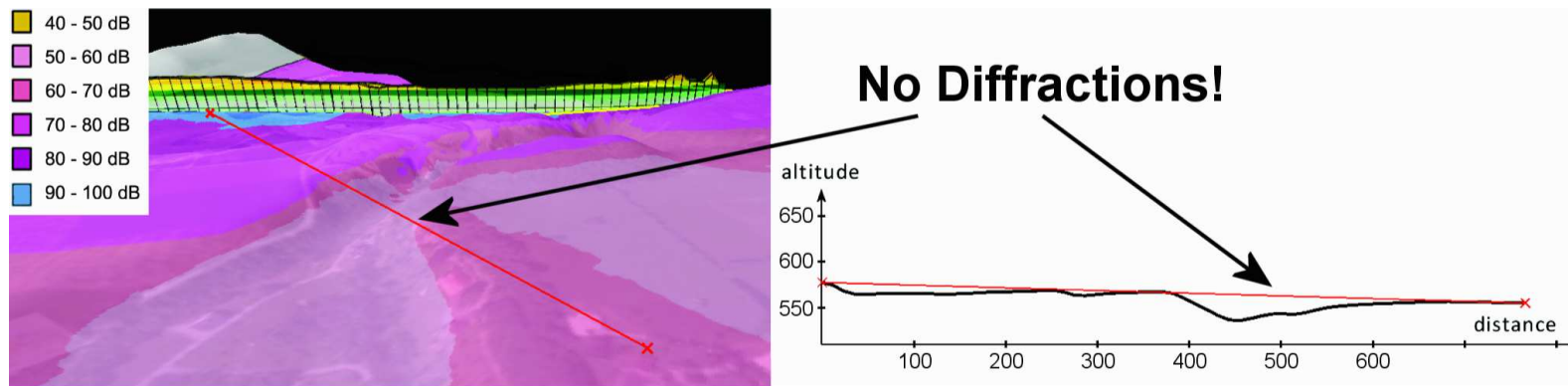


Figure 5.5: A sample immission graph for an IP next to a country road. Higher noise immissions are visualized with taller prisms and darker colors. The terrain does not obstruct noise propagation, therefore, unsurprisingly, we see that the street is loudest where it is closest to the IP.

Figures 5.6a and 5.6b show the same example scene as previously introduced in Figure 5.3. We can see that the patch of high sound pressure in the lower part of the scene is partly due to the left and the right part of the street, as the immission graph in these sections is largest. Noise Propagation Terrain Slices allow us to understand the reasons behind this phenomenon: Sound from the left part of the street can propagate freely (without diffractions) to the IP and sound from the right-hand side of the street is only diffracted *once*, whereas the shape of the terrain suggests that surrounding areas can benefit from an additional sound diffraction point (see Figure 5.6a). Protecting an IP situated inside the patch of elevated noise pressure thus requires the construction of two noisecreens (or similar measures) to cover all relevant noise propagation paths. In this way we can combine visualizations to analyze and understand complex situations not accessible by individual visualizations. Again, for further examples and an evaluation of 3D Immission Graphs, we refer to Chapter 7.



(a) The right peak of the immission graph can be explained by examining the corresponding terrain slice. Sound is only diffracted *once*, but the shape of the sound propagation suggests that in surrounding areas, sound may be diffracted *twice* (which has already been confirmed in Figure 5.3).



(b) High values on the left side of the immission graph seem to stem from the fact that sound can freely propagate along the illustrated path (no diffractions).

Figure 5.6: 3D Immission Graphs help analyzing complex sound situations. In this example, the immission graph shows two especially noisy sections of the street, one on the left side, the other on the right side of the canyon. Using Noise Propagation Terrain Slices we confirm that noise can propagate almost freely along the highlighted lines (in red). Protecting an IP situated inside the area of elevated noise exposure requires the construction of *two* noise screens (or similar measures) to cover all relevant noise propagation paths.

Noise Simulation Results

In this chapter the simulation results by *Noise Simulator* are presented and evaluated by comparing them to real-world measurement results (which we interpret as ground truth data). All scenarios described in this chapter are real-world examples and correlate with specific projects which are briefly outlined in their respective sections (Sections 6.2, 6.3 and 6.4). Some names and locations (specifically the locations of immission points) have been changed or omitted to protect the privacy of the involved parties.

6.1 Ground Truth Data and Expected Simulation Errors

To test the simulation accuracy of *Noise Simulator*, real-world noise measurements of some common noise situations are compared to results obtained by using *Noise Simulator*. Specifically, this process entails:

1. Locating and identifying noise measurement sites, i.e., immission points (which will be used in *Noise Simulator*)
2. Obtaining and filtering noise measurements of these points. Specifically, we only consider noise measurements that satisfy the following conditions:
 - a) Temperature and humidity values stay constant for at least one hour at a time
 - b) Favorable wind situation (either calm or in the direction of the noise propagation)
 - c) Other noise emissions (railway, animal sounds etc.) have been manually removed where applicable
 - d) For long-term measurements (e.g., highway) we use the loudest hour to compare simulation results with
3. Obtaining 2D and 3D map data of the given scene to load in *Noise Simulator*
4. Adding emission points (streets and vehicles), immission points and all relevant additional 3D structures like houses, noise screens and forests
5. Comparing *measured* with *simulated* results

This section describes to what extent simulation results deviate from real-world measurements and discusses the various error sources. To this end, three real-world scenarios in Austria are presented and evaluated in turn, namely the Red Bull Ring in Styria, the street bypass project in Canyon Village¹ and the ‘Reduction of Highway Noise Emission’ project in Green Valley². To provide some context, each scenario is first briefly outlined. Then, the simulation and its parameters are discussed. Also, since emission values for simulated vehicles are based on additional measurements, they too are presented and justified. Finally, noise measurements are compared with simulation results and the deviations are discussed.

Expected Errors

ISO 9613-2 only defines expected accuracy values for the simple case of free sound propagation. Table 6.1 summarizes these values. This situation occurs very rarely in real-world scenarios.

For the three scenarios which are evaluated hereafter, emitter-receiver distances are always between 100 and 1000 meters and the mean sound propagation heights are lower than 30 meters. We can therefore expect simulation accuracy to be ± 3 dB *or worse*, since more complex scenarios with possible multiple noise diffractions are likely to yield more inaccurate results.

Nevertheless, it should be noted that simulation results performing worse than the above (for whatever reasons) are very undesirable and are usually discarded as being wrong altogether. This is all the more true in the case of comparing $L_{A,eq}$ values instead of individual noise measurements. We will therefore interpret these accuracy values as accuracy *goals* for *Noise Simulator* and consider results outside these boundaries to be *errors*.

Table 6.1: Estimated Accuracy of ISO 9613-2

Mean Height	0 < distance < 100 m	100 m < distance < 1000 m
0 m < h < 5 m	± 3 dB	± 3 dB
5 m < h < 30 m	± 1 dB	± 3 dB

6.2 The Red Bull Ring in Styria, Austria

The Austrian racing track in Styria, formerly known as *Österreichring* and now called the *Red Bull Ring*, has recently become an object of public interest and controversy. Red Bull, the new owner as of 2003, decided - perhaps prematurely - to demolish some buildings and disrupt the racing track in 2003, even before the approval procedure of the newly planned large-scale Project “Projekt Spielberg” had been successfully completed. Though the project was approved at the first legal instance in June 2004, a subsequent appeal by some citizens in the neighborhood was allowed by the next legal instance, which led to the reversal of the first instance’ judgment on December 4th, 2004. At this point, since the racing track had already been disrupted, *no* racing or driving could be conducted on the racing track since both building and operating rights had been voided.

¹Name changed to protect the privacy of the involved parties.

²See footnote 1.

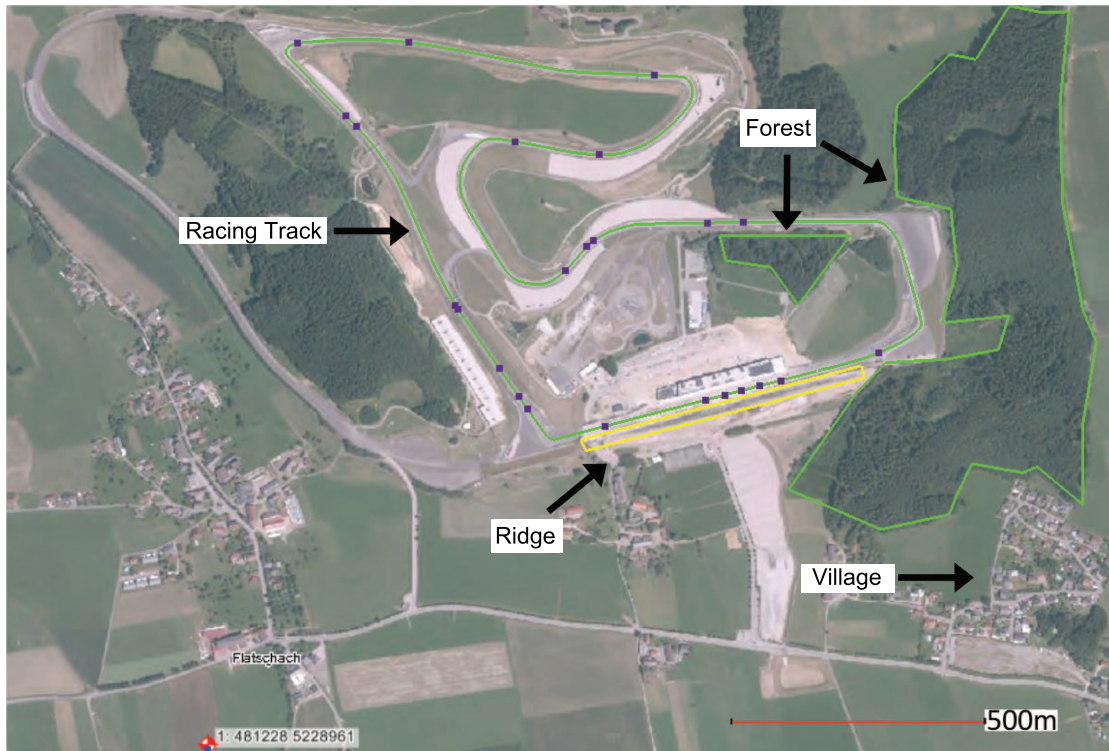


Figure 6.1: 2D view of the Red Bull Ring while simulating the DTM training that took place on June 1st, 2011. The green polygons outline the (relevant) forest regions, the yellow quad shows the location of an artificial ridge due south of the homestretch.

In an attempt to revive the project, the Styrian government began planning of a rescaled Project “Projekt Spielberg Neu” in 2005. In cooperation with the aforementioned neighbors this project was finally approved in September 2007 and given over to Red Bull. After the necessary constructions had been completed, the new Red Bull Ring began operations in May 2011.

To ensure that noise levels in the neighborhood would not exceed the agreed-upon values specified in the operating permit, several noise measuring stations were established by both Red Bull and the neighbors. The measuring results of these stations provide the ground truth to test *Noise Simulator*. For the purpose of this evaluation we will focus on one specific racing event, namely the DTM training which took place on the 1st of June, 2012. Figures 6.1 and 6.2 show the corresponding simulation in *Noise Simulator* in 2D and 3D. The village depicted in the south-east corner is the nearest settlement and thus the area of most interest. Noise measurements have been taken in and around this village³, the simulation used these same locations. The village is separated from the racing track by a wooded hill. On-site inspection yielded an average forest height of 20 meters (disregarding tree tops). In the south of the racing track, an artificial ridge has been constructed to act as a noise screen. The height varies between 8 and 9 meters. For the

³The exact immission point locations will not be disclosed in this thesis to protect the privacy of the involved parties.

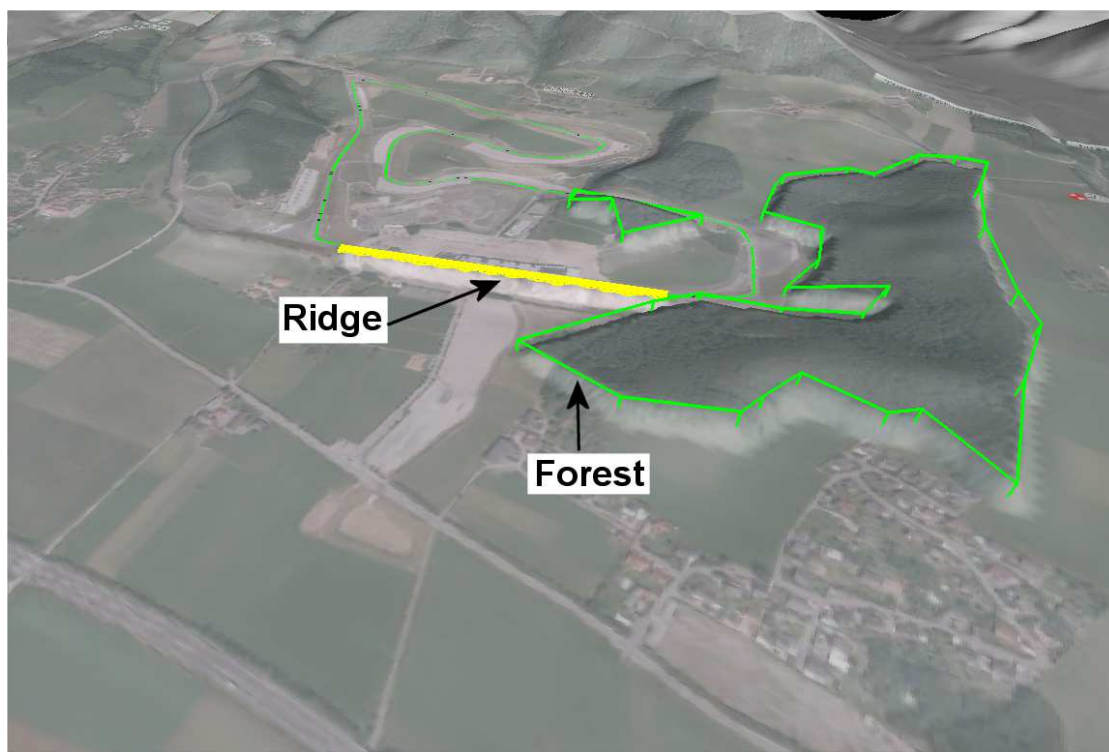


Figure 6.2: 3D view of the Red Bull Ring while simulating the DTM training that took place on June 1st, 2011. The green polygons outline the (relevant) forest regions, the yellow quad shows the location of an artificial ridge due south of the homestretch. Forest regions and the ridge are elevated by their respective heights.

purposes of this thesis we will assume its height to be uniformly 9 meters.

Simulation Parameters Noise emissions of DTM racing cars have been evaluated separately to be approximately 143 dB at a distance of one meter. This value reflects the mean emission of DTM racing cars and already takes into account that emission values differ in-between various sections of the track (as a function of vehicle speed). 24 DTM racing cars participated in this training. During the loudest hour (14 till 15 o'clock) temperature and humidity values were approximately 20° C and 40 %, respectively. Wind direction changed multiple times between 14 and 15 o'clock, but was generally favorable (between west and north-west), i.e., in direction of the village. Since all immission points are in relatively close proximity to each other, these values were used for all of them.

Noise Measurements and Comparison Figure 6.3 depicts the noise measurements taken at immission point 2 (see also Table 6.2). There were two separate training sessions and the loudest hour - which we will subsequently use as ground truth - took place between 14 and 15 o'clock. This is also the loudest hour at the other immission points (IP 1 and IP 3).

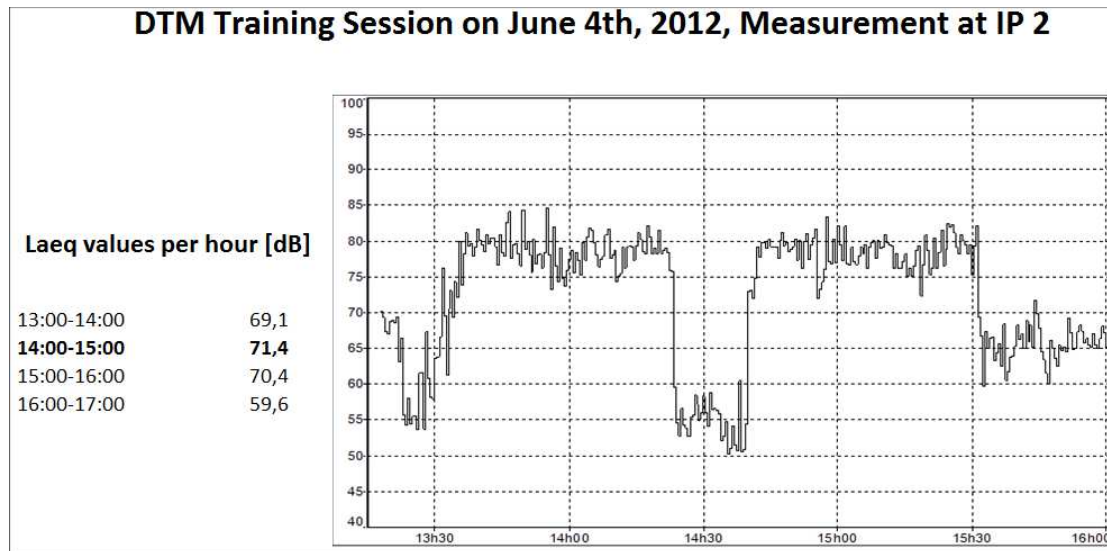


Figure 6.3: Noise immission values for a DTM training session on June 4th, 2012, taken at IP 2. The table on the left shows the equivalent A-weighted sound pressure levels for each hour. The loudest hour is highlighted. Measurements were taken using the sonar 01dB Type Symphonie as shown in Figure 1.1a, the graph was created with the software dbTrait by the same company.

Table 6.2 shows measurement and simulation results for all evaluated immission points. Since individual measurements cannot be compared directly (since the simulated race does not reflect the actual car movement during the DTM training) we compare $L_{A,eq}$ values instead. The closest convex sound distances of the immission points to the racing track range from 345 meters (IP 1) to 400 and 431 meters (IP 3 and IP 2, respectively). The differences between measured and simulated noise immission values are quite small with a maximum deviation of 1.77 dB, which is well within expected boundaries. Simulated values are consistently higher than measured ones. This is very likely due to the implicit simplification that racing cars emit their full noise pressure in all directions equally. This is of course not true, in reality maximum emission values are usually in a cone behind the vehicle. To accurately simulate this behavior we would need to perform a 360 degrees noise measurements of a typical vehicle. However, since we are only interested in a worst-case simulation, we can skip this step. We can conclude that *Noise Simulator* accurately calculates the worst-case A-weighted sound pressure levels for this scenario and the given immission points.

Table 6.2: DTM Training (June 4th, 2012), Evaluation Results ($L_{A,eq}$)

Immission Point	Measurement [dB]	Simulation [dB]	Difference [dB]
IP 1	78.8	80.57	1.77
IP 2	71.4	72.87	1.47
IP 3	72.1	73.29	1.19

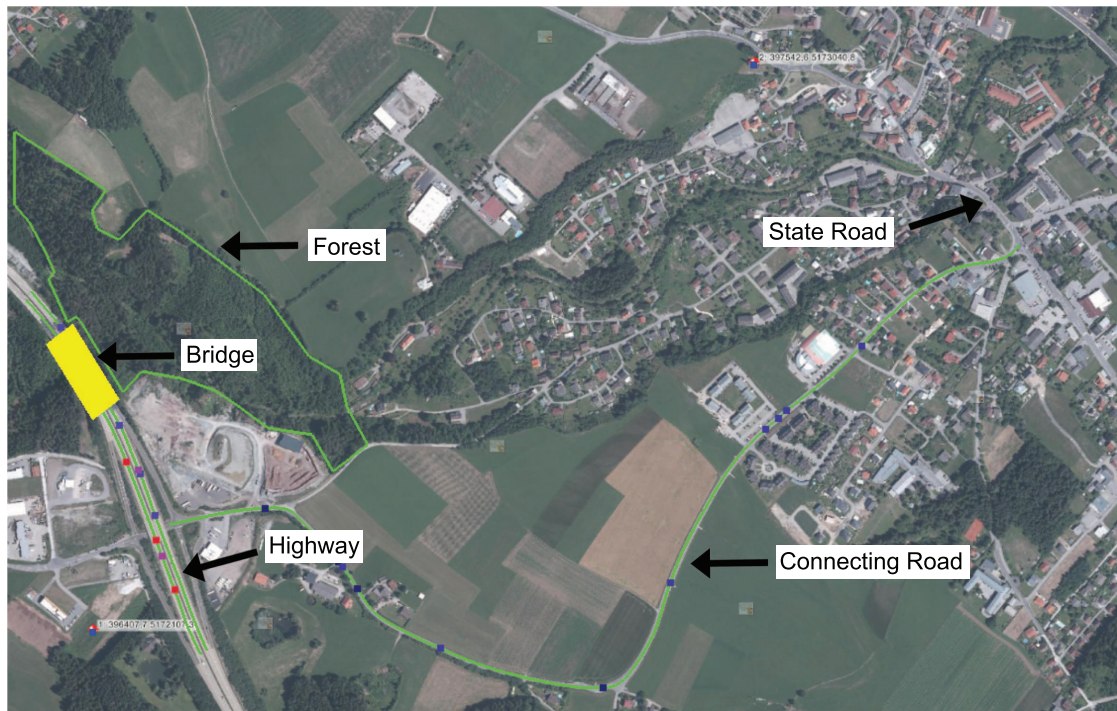


Figure 6.4: 2D view of the current traffic situation in Canyon Village. Since for the immission point of this scenario background noise is dominated by the highway traffic, other streets and emitters can be safely ignored and are not shown or simulated separately. The green polygon outlines the relevant part of the forest. The yellow rectangle depicts the location of the bridge across the canyon.

6.3 Construction of a Street Bypass to lower Traffic Noise in Canyon Village

Canyon Village lies in a canyon and is situated east of a highway which is responsible for most of the environmental noise (see Figure 6.4). Problematically, many vehicles exit the highway to reach the state road by way of moving through the village.

In 2011 the local government of Canyon Village decided to offset some of the traffic noise by constructing a street bypass to the west of the village (see Figure 6.5). This plan, however, caused some frictions with several inhabitants of Canyon Village who are situated in the vicinity of the newly planned road. The provincial government raised some concerns as well, specifically pertaining to the necessity and the cost of this project. As of this date (April 2013) the future of this project is politically debated and uncertain.

Nevertheless and in preparation of defending their rights as citizens of Canyon Village, several inhabitants employed external consultants and experts to measure their current noise situation in late 2011. These measurements capture the status-quo at one particular immission point and are used as ground-truth data for this simulation scenario. Subsequent analyses of the

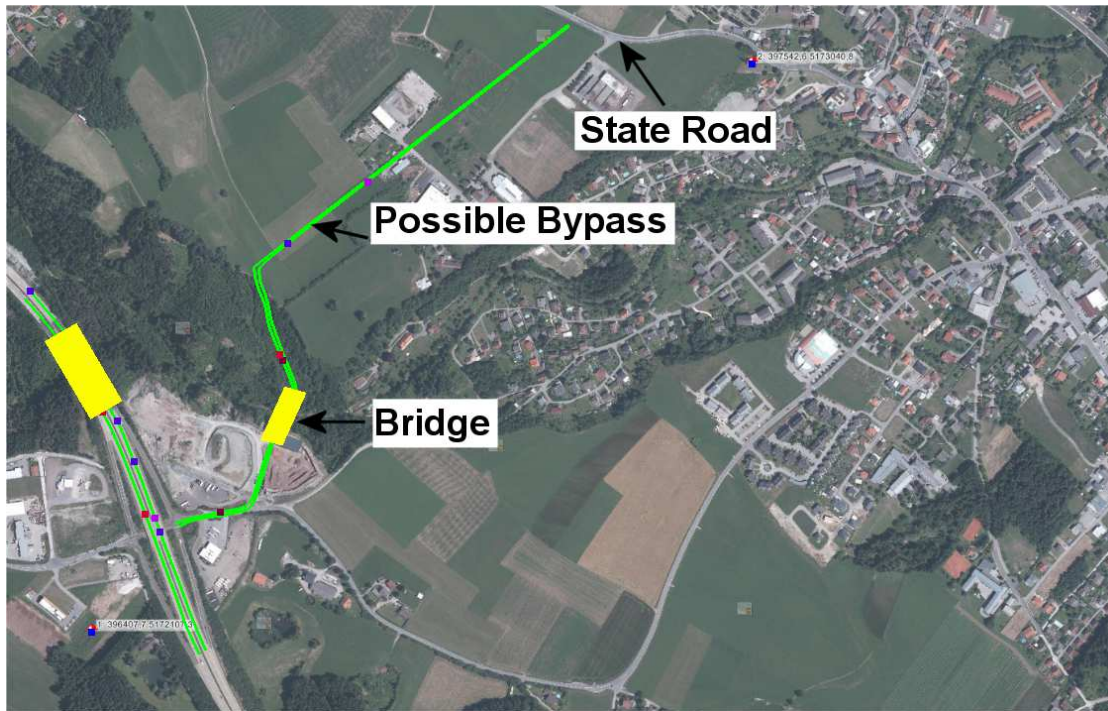


Figure 6.5: 2D view of the planned street bypass. For better visibility the forest is omitted in this plot.

measured data reveals that environmental noise at the given immission point is very strongly influenced by the highway to the west of Canyon Village and irregular helicopter noise, as well as environmental noise of a nearby commercial center. Other traffic noises (including the current connection from highway to state road through the village) are *inaudible* and are consequently ignored in the simulation. Since ISO 9613-2 cannot be used to evaluate the noise propagation of helicopters or planes, we will instead focus on the simulation of the highway. The 3D view reveals a much more diverse terrain as was the case for the Red Bull Ring (Section 6.2). A large part of the village lies below the highway and within a canyon that it eventually crosses. Some inhabitants, however, live alongside the ridge of the canyon (see Figure 6.5, next to the planned street bypass).

It should be reiterated that noise measurements were not intended to solely capture the influence of highway traffic, but environmental noise in general. Therefore we can not expect simulation results to accurately reflect real-world measurements. However, we *can* make educated guesses as to the expected traffic density during and after peak hours, simulate the highway for both situations and analyze whether *relative* immission changes are accurately reflected or not.

Simulation Parameters Unfortunately no traffic counts were conducted at the time when the noise measurements took place. We therefore have to rely on statistical data. According to the

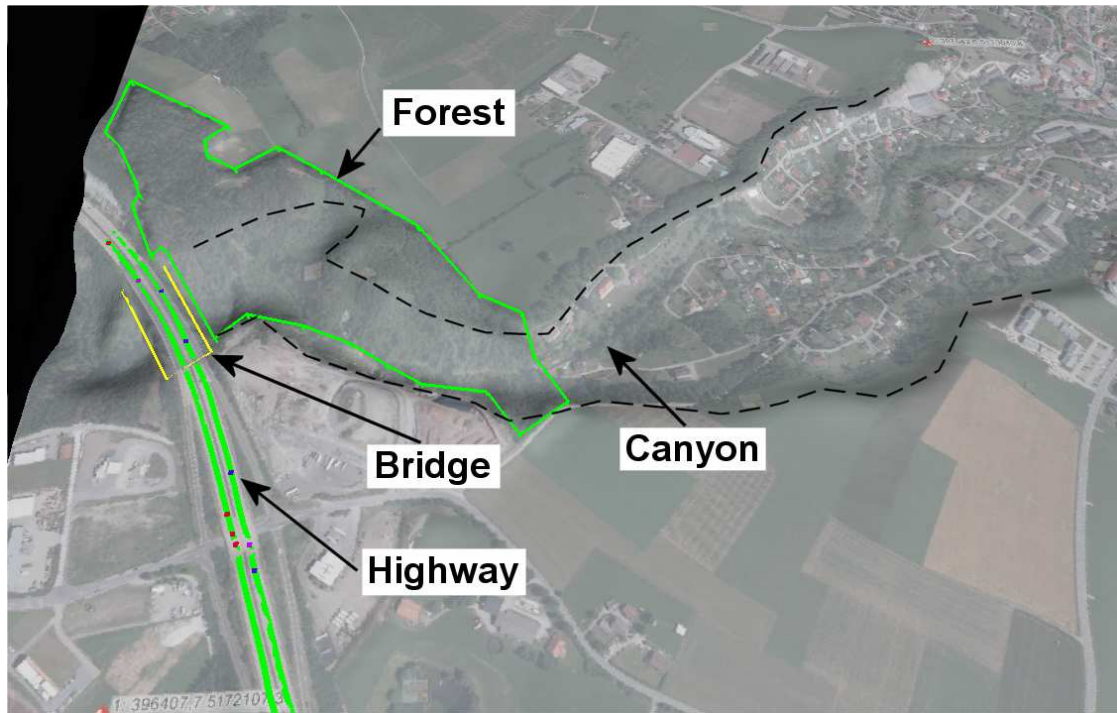


Figure 6.6: 3D view of the highway west of Canyon Village and its environments.

Austrian ministry for traffic, innovation and technology, the yearly traffic at working days at this section of the highway averages to 12775 passenger cars and 4094 trucks per 24 hours (see BVIT 2012 [7]), which would average to 532 passenger cars and 171 trucks per hour. We can further assume the traffic frequency to change depending on the time of day, with peak values in the morning and in the afternoon roughly coinciding with the daily commuter traffic. This is further corroborated by traffic counts conducted in Green Valley which exhibit this very behavior (see Section 6.4). If we assume that the relation between peak hour and non-peak hour traffic frequency in Canyon Village is roughly the same as measured in Green Valley - a coefficient of 1.5 to 2 - we can make an educated guess as to the amount of traffic during peak hours in Canyon Village. We multiply the value for non-peak hour traffic by this coefficient, resulting in 1064 passenger cars and 342 trucks in the loudest hour and, by normalizing the rest of the day, an average of 484 passenger cars and 155 trucks during non-peak hours.

Apart from the number of vehicles to simulate we also need to measure the noise emission of individual passenger cars and trucks for use in the simulation. Using a portable sonar as described in Section 1.4, individual car and truck emissions on highways were measured and averaged, resulting in a total emission of 100 dB for passenger cars and 102 dB for trucks (see Figure 6.7). Finally, separating the region of interest from said highway is a small forest (mostly saplings with approx. 10 m height) which will be modelled as well.

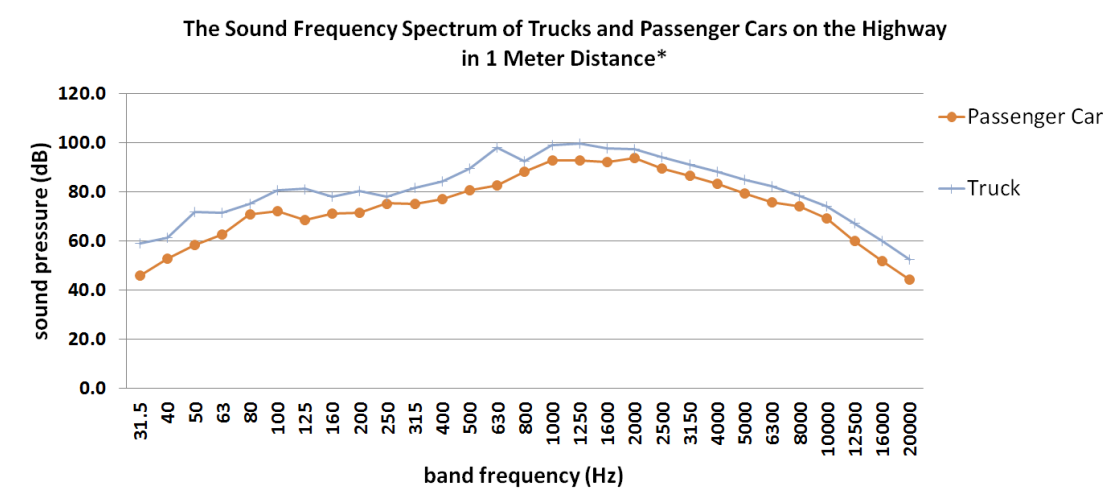


Figure 6.7: The sound frequency spectrum of a sample truck and passenger car on an Austrian highway (see also Figure 1.1d). Values have been adjusted to reflect the sound emission at a distance of one meter.

Noise Measurements and Comparison Figure 6.8 depicts the equivalent A-weighted sound pressure levels per hour for five subsequent workdays (from October 12th to October 18th, 2011) at an immission point in Canyon Village. Irregular values on October 13th and 17th are circumstantial, mainly due to other noise emissions next to the immission point (noises caused by neighbors, including lawn-mowers and other gardening devices) and have been ignored when computing the average immission values. As expected, we can observe two peak hours during work days, namely 07:00 to 08:00 (47.58 dB) and 17:00 to 18:00 (47.45 dB). Averaging the remaining hours yields a value of 44.33 dB. Peak hours and non-peak hours therefore show an immission difference of roughly 3 dB.

If we simulate this session with the above parameters, once for non-peak hours and once for peak hours, we get values of 39 and 42 dB, respectively. These values are quite significantly lower than the measured ones, but this is - as previously mentioned - due to additional sound sources in the vicinity of the immission points which could not be modelled with *Noise Simulator*. However, we can still observe an immission difference of roughly 3 decibels between peak and non-peak hours, which very nicely coincides with our measurements. We can conclude that - apart from the bias - our simulation is consistent with the real world measurements.

6.4 Green Valley, Elevation of Highway Noise Screens

Green Village is an Austrian municipality situated close to a very frequented highway, as depicted in Figure 6.9. Since the terrain is flat and does not exhibit any interesting particularities, the 3D plot will not be shown here. Until 2008 noise screens were only 2.5 meters high and the municipality had already been fighting many years to ameliorate the noise situation for their citizens by means of additional noise screens. Finally, in summer 2008 and as part of a general

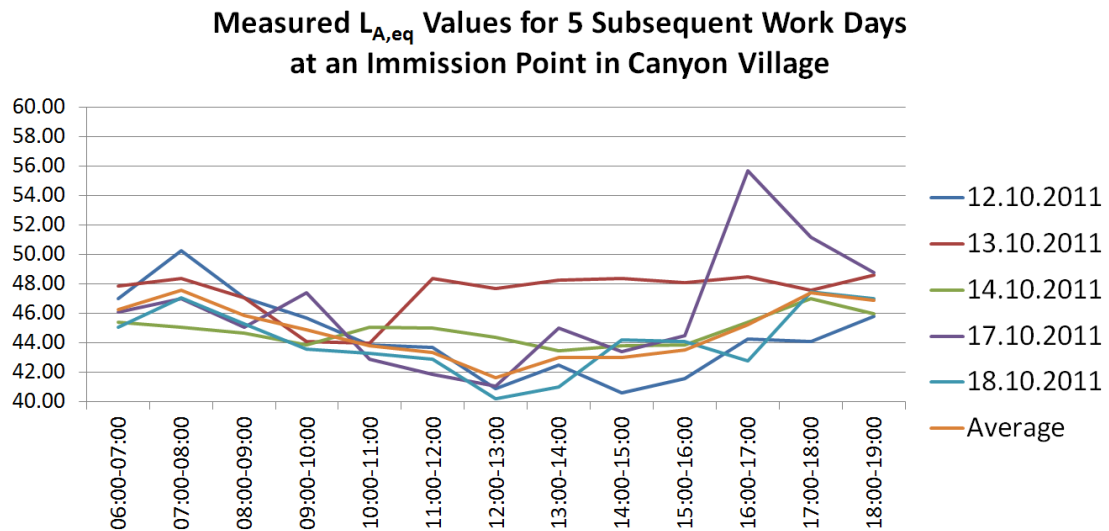
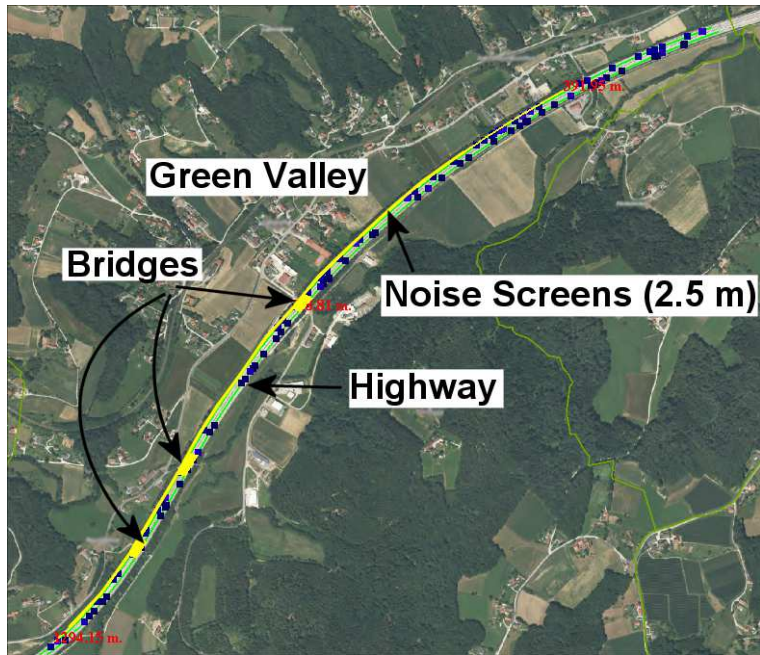


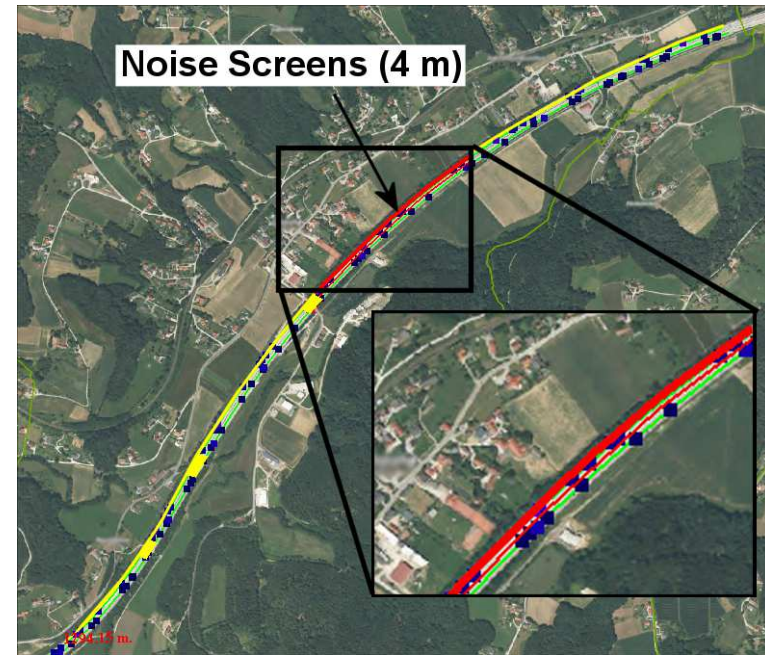
Figure 6.8: Average measured per-hour immission values for an immission point in Canyon Village during five subsequent work days from October 12th to October 18th, 2011. Irregular values occurring on October 13th and 17th are due to circumstantial environmental noise in the vicinity of the microphone and have been omitted when calculating average immission values.

refurbishment of all noise screens and streets in the area, at least part of them were elevated to 4 meters and some additional screens in-between highway lanes (also 4 meters high) were introduced (see red lines in Figure 6.9b). At several points of the municipality (both benefitting and not benefitting from the additional noise screens), noise immission values were measured *before*, *during* and *after* said alterations were made. Measurements *during* construction are especially interesting since at the time of measurement all noise screens had been effectively decommissioned and a speed limit of 80 km/h (instead of 130 km/h) was *enforced* by the use of a *Section Control*, an Austrian law-enforcement system used to track (and, if applicable, fine) drivers by calculating their average velocities within a street section. This allows us to observe the noise reduction as a function of vehicle speed and compare it with noise reduction as a function of noise screen height. Vehicle counts during and after construction of the noise screens are available (we estimate the value before the noise screen alterations to be roughly the same as afterwards) which should make simulating this scenario a lot easier and accurate than in the Canyon Village scenario (see previous section).

As we will see below, noise immissions could be lowered, but are still higher than stipulated in the directive of the Austrian ministry for traffic, innovation and technology (maximum 60 dB during the day and 50 dB at night, see BVIT 2011 [8]). At the time of writing (April 2013) a civil court case to enforce adherence to said immission directive is being prepared.



(a) Before 2008 noise screens (yellow lines) were only 2.5 meters high.



(b) In summer 2008 the northern part of the noise screens was elevated to a height of 4 meters and additional 4 meter screens in-between highway lanes were introduced (changes are highlighted in red).

Figure 6.9: The highway noise situation in Green Valley before and after the elevation of noise screens in summer 2008. Since Green Valley is situated north of the highway the southern noise screens have been omitted in this plot. Measurements confirm that this does not effect simulation accuracy for this scenario.

Simulation Parameters A traffic count conducted between April 22nd and April 24th 2009 (all working days) shows that both directions of the highway were similarly frequented. At April 22nd, a total of 25678 vehicles (3732 of which were trucks) travelled along the lanes to the north and 28299 vehicles (3245 trucks) travelled along the lanes to the south (vehicle counts at other dates are similar). At the peak hours (between 18:00 and 19:00 o'clock for the northern lanes and between 07:00 and 08:00 o'clock for the southern lanes) 2470 vehicles (320 trucks) drove to the north and 2279 vehicles (282 trucks) drove to the south. During the noise screen refurbishment northbound traffic had been counted as well (16.10.2008) and values were found to be 2250 passenger cars and 492 trucks per hour. Southbound traffic was not counted; we assume the relative traffic density of northbound and southbound traffic to be roughly constant in order to estimate these values. The high number of trucks observed may be due to specific circumstances but has little impact on the noise emission anyway, since at 80 km/h, passenger cars and trucks exhibit very similar noise pressure levels and the total number of vehicles per hour is consistent with the other evaluation. For April 2008, only rough vehicle counts are available. We will therefore populate the simulated lanes with the number of vehicles per hour shown in table 6.3.

Table 6.3: Maximum Vehicle Counts for Both Green Valley Highway Directions Before, During and After Noise Screen Refurbishment

Date	Northbound		Southbound	
	Passenger Cars	Trucks	Passenger Cars	Trucks
28.04.2008	2300	230	1800	230
16.10.2008	2250	492	2090 (est)	434 (est)
22.04.2009	2150	320	1997	282

Wind directions as well as humidity and temperature values were determined for all three measurements and all immission points individually. We disregard small temperature [$< 2.5^{\circ}\text{C}$] and humidity differences [$< 10\%$] since they do not factor in the overall result in a measurable way. Wind directions were favorable during all measurements (south - southeast), Table 6.4 shows the rounded environmental values used in the simulation.

Table 6.4: Rounded Temperature and Humidity Values in Green Valley

Date	Temperature [$^{\circ}\text{C}$]	Humidity [%]
28.04.2008	15	40
16.10.2008	20	50
22.04.2009	15	50

The terrain is mostly flat and does not obstruct noise propagation. Three simulation runs were conducted to cover the highway development between April 2008 and April 2009. Typical highway emission values for trucks and passenger cars were used as outlined in Figures 1.1d and 6.7. Since velocities (and therefore emissions) due to an enforced speed limit in August 2008 had been lower, an additional noise measurement was conducted during that time. Average sound pressure levels of 95.13 dB (as opposed to 101.9 dB) show a significant noise reduction of almost 7 decibels.

Noise Measurements and Comparison Tables 6.5 , 6.6 and 6.7 show the simulation results and a comparison with the ground truth. The outcome is very satisfying and all values are within acceptable error tolerances.

Table 6.5: Green Valley (**April 2008**), Evaluation Results ($L_{A,eq}$)

Immission Point	Measurement [dB]	Simulation [dB]	Difference [dB]
IP 1	52.8	52.8	0.0
IP 2	59.9	58.2	1.7
IP 3	62.0	61.7	0.3

Table 6.6: Green Valley (**August 2008**), Evaluation Results ($L_{A,eq}$)

Immission Point	Measurement [dB]	Simulation [dB]	Difference [dB]
IP 1	44.8	47.7	2.9
IP 2	n.a.	53.2	n.a.
IP 3	56.3	56.75	0.45

Table 6.7: Green Valley (**April 2009**), Evaluation Results ($L_{A,eq}$)

Immission Point	Measurement [dB]	Simulation [dB]	Difference [dB]
IP 1	52.9	51.74	1.16
IP 2	57.9	58.02	0.12
IP 3	61.2	61.61	0.41

Unfortunately, the noise screen refurbishment seems to have had little impact on the observed (and simulated) immission values. Only IP 2 and 3 show a moderate reduction (2.0 dB and 0.8 dB, as measured). Interestingly, the temporary enforced speed limit in August 2008 resulted in reductions by as much as 8 dB (IP 1) and 5.7 dB (IP 3). Due to an equipment failure, no ground truth data for IP 2 in August 2008 is available.

We can conclude that despite governmental efforts, the noise situation in Green Valley is still critical and the municipality prepares to take this case to the civil court (at the time of writing, April 2013). *Noise Simulator* performs accurately in all three scenarios with the input values discussed above.

Noise Visualization Results

In this chapter visualizations for the sample scenarios are presented and analyzed. These scenarios have already been described in Chapter 6. Please refer to their respective sections for a short project summary, namely Section 6.2: The Red Bull Ring in Styria, Austria; Section 6.3: Construction of a Street Bypass to lower Traffic Noise in Canyon Village and Section 6.4: Green Valley, Elevation of Highway Noise Screens.

After having evaluated the simulation results and confirmed the accuracy of *Noise Simulator*, we can now turn to analyze the presented scenarios using various visualization techniques. Visualizations are always employed to attend very specific needs of the user. In our case, we will focus on one research question per scenario, which will be discussed in the following order:

1. Red Bull Ring: How large is the noise dampening effect of the noise barrier south of the home stretch for different sections of the racing track?
2. Canyon Village: What is the reason for the stretch of high noise immissions east-southeast of the highway and how might this problem be mitigated?
3. Green Valley: Noise screen refurbishment versus speed reduction: which one is more effective and why?

Keep in mind that this thesis is mainly a *technical* one, focussing on various aspects of noise simulation and visualization but less on noise analysis. Though the findings presented hereafter have been confirmed with noise experts, they *can not* replace a formal noise evaluation by certified assessors or experts. As already mentioned in Chapter 1, this work is intended to be used in conjunction with existing simulation toolkits and not on its own. With that being said, we will now attempt to put the various previously mentioned visualizations (see Chapter 5) to good use and solve the above questions.

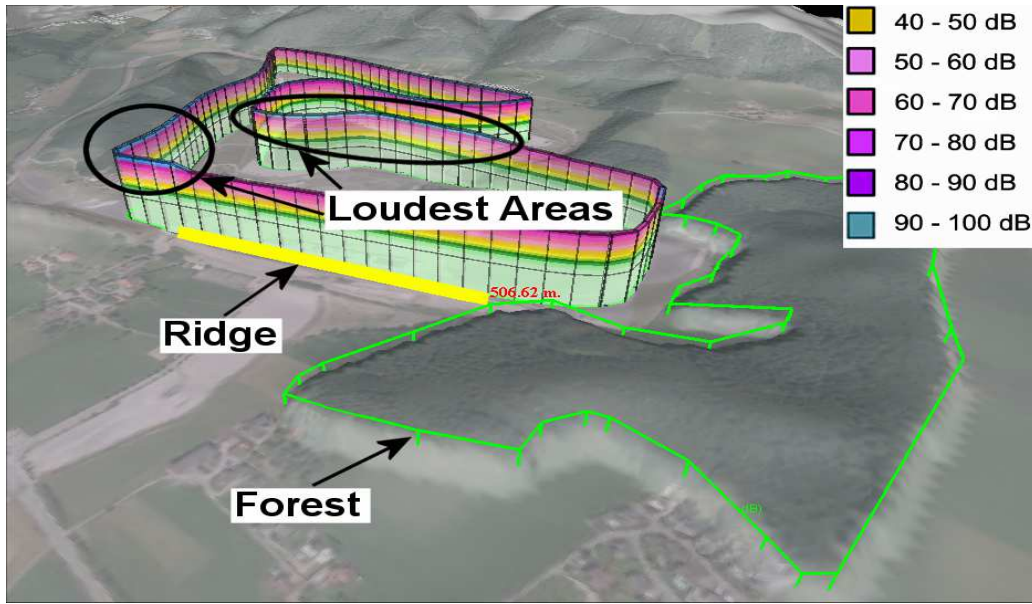
7.1 The Red Bull Ring in Styria, Noise Barrier Evaluation

To protect affected neighbors, an 8 to 9 meters high ridge was constructed along the southern stretch of the racing track. Since we have already established the accuracy of *Noise Simulator* for this scenario and the given immission points, we are now able to alter simulation parameters and observe the changes. Figures 7.2a and 7.2b show the noise overlay for both the original and modified scenario. Though the presence of the ridge leads to a large immission reduction in a lane due south of the racing track, large parts of the village to the south-east (which, as mentioned before, is the area of most interest) are completely unaffected by the presence of this ridge. This is also reflected in the simulation results shown in Table 7.1. IP 1, situated in this lane of 'noise leakage', is directly affected and experiences a very large immission increase of almost 8 dB. Immission values at IP 2 and 3, on the other hand, do not change at all. Note that in reality, immission values *would* change at IP 2 and 3 as well due to reflection effects (not simulated by *Noise Simulator*) and also due to the fact that, contrary to ISO 9613-2's simplification, sound does not travel along geometric *lines* (see also Section 1.2). Even so, the benefit of the ridge on the majority of the village is sure to be very small. In this example, noise overlays illustrate the area of effect for the given noise barrier and thus allow to infer better noise barrier placements (i.e., further east).

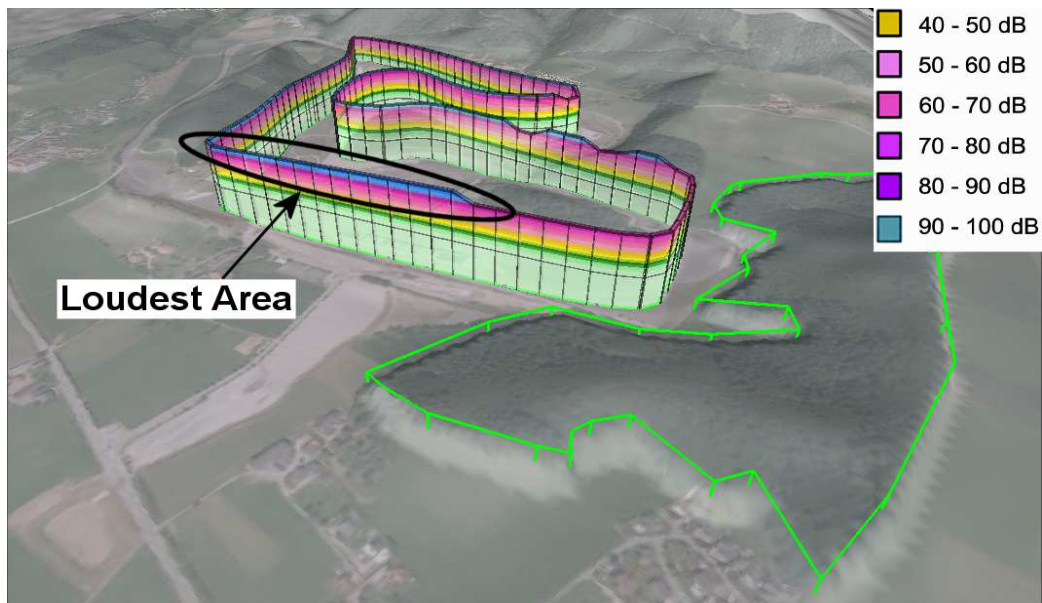
Table 7.1: DTM Training Simulation with and without Noise Screen

Immission Point	Simulation w/ Ridge [dB]	Simulation w/o Ridge[dB]	Difference [dB]
IP 1	78.8	86.58	7.78
IP 2	72.87	72.87	0.0
IP 3	73.29	73.29	0.0

One can also make use of immission graphs to analyze both current and hypothetical scenarios with respect to one specific immission point. For IP 1, these graphs are shown in Figures 7.1a and 7.1b. Noise from the southern part of the racing track is blocked very efficiently, to the point that most of the incoming sound at IP 1 now originates from other parts of the ring, namely the western and mid-northern sections (where the immission graph is highest). Note that the immission graph shows some farther sections of the racing track being responsible for more of the incoming noise at IP 1 than other, closer, sections. The most likely reason is that these parts are - compared for example to the home stretch in the south - elevated, which allows sound emitted from these points to more easily bypass the noise ridge.

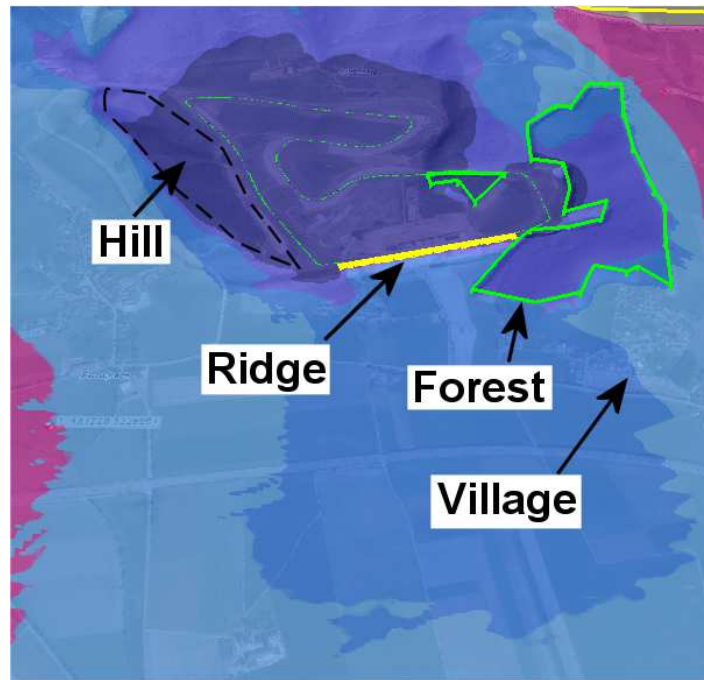


(a) Immission graph for a sample immission point *with* the noise barrier.

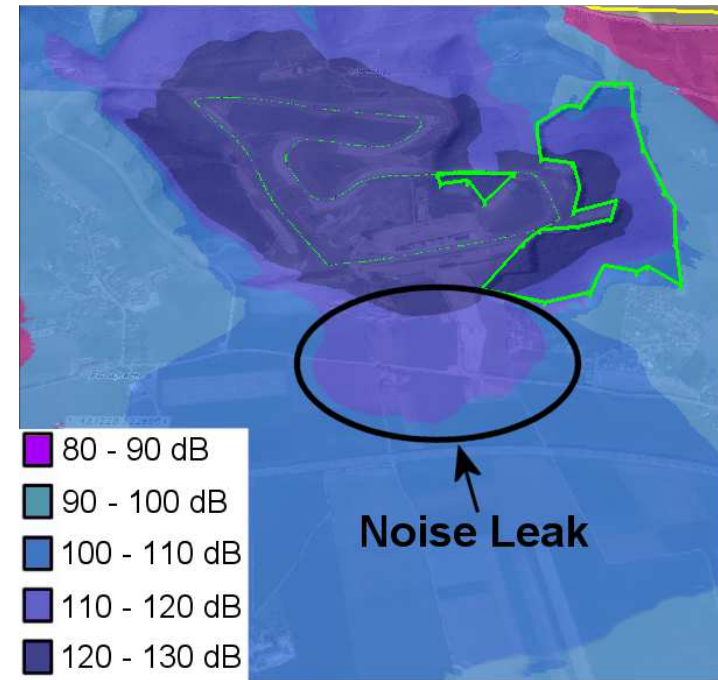


(b) Immission graph for a sample immission point *without* the noise barrier.

Figure 7.1: 3D immission graphs for IP 1 (south-southwest of the racing track), *with* and *without* the noise barrier south of the racing track. The barrier's influence can be clearly seen along the southern part of the immission graph. With the noise screen in place we can see that parts of the northern and western sections of the track are largely unaffected and are now responsible for most of the incoming noise at IP 1. This is most likely a result of these parts' higher altitudes, allowing sound to bypass the noise barrier more easily.



(a) The noise overlay for the original, i.e. 'real', scenario.

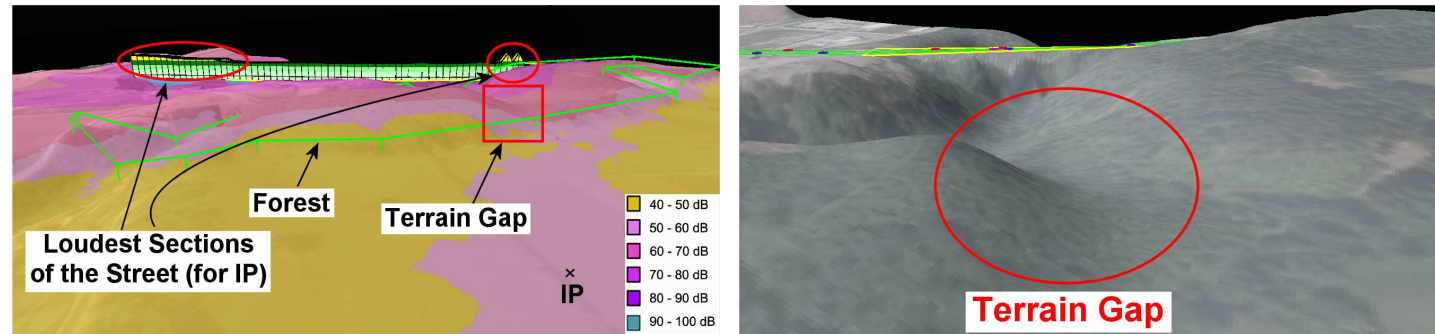


(b) Removing the 9 meter high ridge results in a lot of noise 'leaking' to the south. Note, however, that a large part of the village south-east of the racing track is completely unaffected.

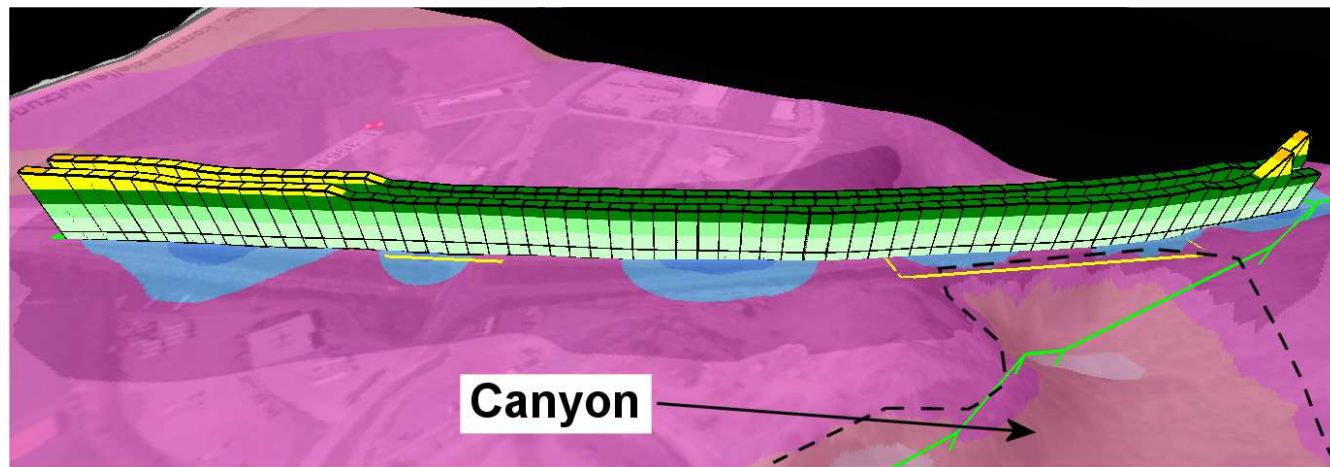
Figure 7.2: Red Bull Ring Noise Overlay for DTM training simulations with and without the noise dampening ridge along the south section of the racing track. Small visualization changes are due to slightly different simulation timestamps and are safe to ignore. See Table 7.1 for the quantitative changes in the simulated immission values.

7.2 Canyon Village, Highway Noise Visualization

Large parts of this evaluation were already previously shown and discussed to illustrate the construction and usage of 3D immission graphs and other visualization techniques (see Chapter 5 and especially Section 5.2 therein). To recount the problem: east of the highway, a small section in and around the area of interest shows increased immission values. This section can be easily identified using noise overlays, but needs additional analysis to be explained. Terrain Slices, as seen in Figure 5.3, show that for this particular terrain section, sound is diffracted only *once*, whereas in surrounding areas noise is diffracted twice or even thrice, a strong indicator for the reason behind this phenomenon. A more readily available explanation can be found using immission graphs. Figure 7.3 immediately reveals that the section of high immission values results from two different sources; first, as mentioned above, by sound travelling through a narrow gap in the hills (see Figure 7.3b for a close-up), second, by sound from the other side of the valley. Protecting the neighbors living in this particular stretch of the terrain is not an easy and/or cheap task. The simplest solution would probably be the erection of noise screens along the east side of the highway. Since current immissions are within allowed immission values for villages in Austria, it seems unlikely that the local government of Canyon Village will implement this step in the foreseeable future.



(a) Immission Graph and noise overlay of the Canyon Village highway and its environments. (b) A gap in the terrain allows traffic noise from the highway to bypass some of Canyon Village's natural noise screens.



(c) Close-up of the immission graph for a particular immission point in Canyon Village.

Figure 7.3: 3D immission graphs for both traffic lanes and an immission point situated in the relatively noisy area in the lower part of the image. We can observe two peaks along the highway, one in the south (left part of the picture) and one in the north (right part of the picture). Noise from the south travels across the canyon, noise from the north can pass through a gap in the two hills east of that section of the highway (see Figure 7.3b for a close-up).

7.3 Green Valley, Noise Screen Refurbishment versus Speed Reduction

In Section 6.4 it has been shown that the effects of the highway noise screen refurbishment to help protect affected neighbors in Green Valley are somewhat limited. Most immission points only exhibited a reduction (both simulated and measured) of about 1 - 2 decibels. On the other hand, the temporary speed limit during refurbishment had a large effect on emitted noise pressures and resulted in reductions by as much as 5 - 8 decibels at various immission points. Figures 7.4a and 7.4b show a comparison of the noise overlays for April and August 2008, i.e., before and during noise screen refurbishment. We can observe that even in the absence of noise screens, traffic noise was significantly lower in August 2008. We can also apply the reduced vehicle immissions to the current scenario, i.e., with noise screen refurbishments already in place, and evaluate the possible immission reduction by (re)introducing a speed limit of 80 km/h for this section of the highway (see Figure 7.4c). Table 7.2 shows a comparison between current noise measurements and simulated values for this hypothetical scenario. Immission reductions vary between 4 and almost 6 decibels, a very large improvement that would very likely satisfy most affected neighbors. We can conclude that a permanent speed reduction on the section of the highway closest to Green Valley would effectively solve the present dilemma. Since a court case is now (April 2013) being prepared, further developments remain yet to be seen.

Table 7.2: Green Valley Highway Simulation Results, Hypothetical Scenario with Noise Screens and enforced speed limit of 80 km/h

Immission Point	Current Measurement [dB]	Simulation [dB]	Difference [dB]
IP 1	52.9	47.16	5.74
IP 2	57.9	53.26	4.64
IP 3	61.2	56.83	4.37

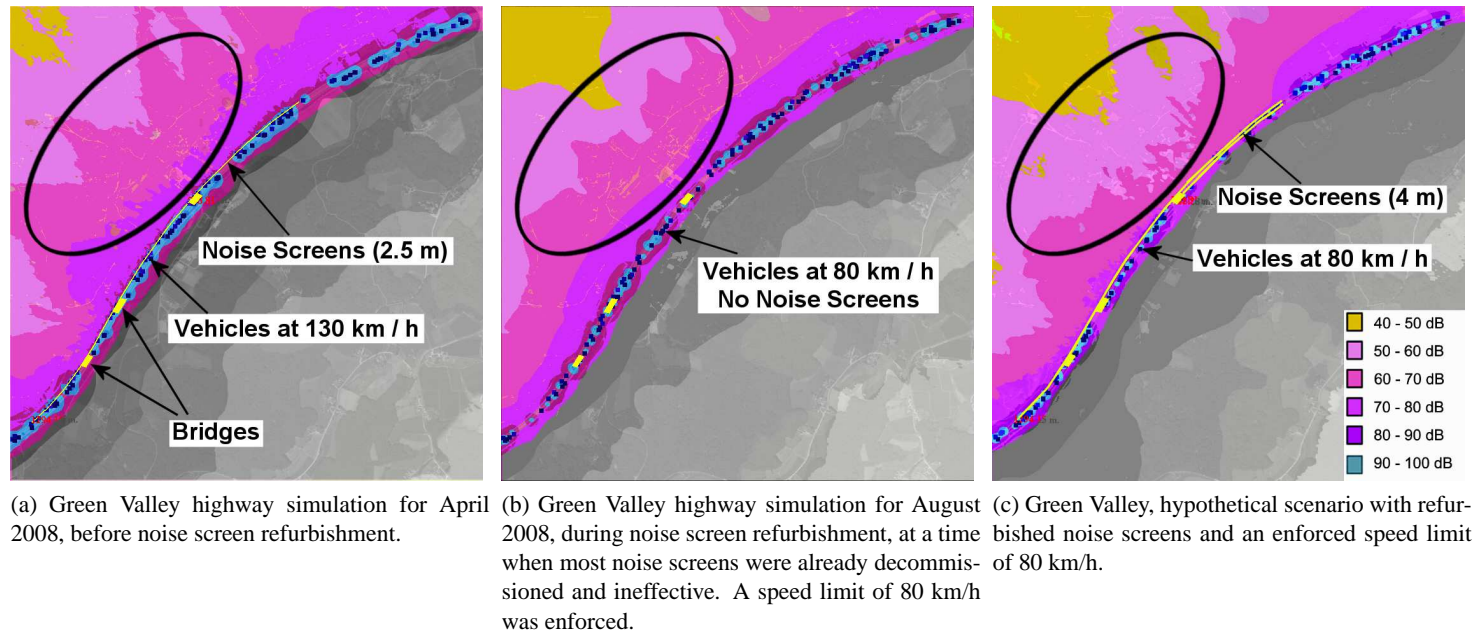
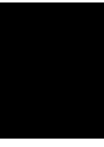


Figure 7.4: Noise overlays for Green Valley, April 2008 (7.4a, vehicles at 130 km/h and intact noise screens) and August 2008 (7.4b, vehicles at 80 km/h, noise screens largely decommissioned and ineffective). Note that even without noise screens, the overall noise situation in August 2008 is still better than in April 2008. Simulation parameters are covered in Section 6.4. A speed reduction at the current time, i.e., with refurbished noise screens, results in a significant improvement of the noise situation (7.4c).



Conclusion and Future Work

In this chapter the findings presented in the previous chapters are summarized (Section 8.1) and some potential future work is discussed (Section 8.2).

8.1 Conclusion

The simulation and visualization of noise is technically challenging and must also take into account other concerns such as privacy and the sensitivity of the matter. Arguably the most important aspect of this research is the communication of the tradeoff between peoples' living conditions and economical factors. Affected neighbors, (local) governments and project solicitors each pursue contrasting interests and have different concerns about noise. Noise assessors are charged with writing assessments, often to promote their employers' interests and not in order to reach a consensus with their employers' 'opponents'. Noise visualizations can help in the mediating process but require a lot of time and effort to create - and the time of a noise assessor can be pricey indeed. Existing simulation software packages like Immi [39] and CadnaA [11] are very expensive and arguably pursue a different goal than *Noise Simulator*, the tool presented in this thesis. The former being mainly interested in delivering the most accurate predictions, the latter focussing on rapid computation and visualization of different scenarios.

With that being said, *Noise Simulator* does not obviate the need for said assessors or noise experts. It does, however, enable them to efficiently evaluate a number of different scenarios and present the results in an intuitive way that is understandable by laymen as well. To this end, qualitative feedback of both laymen and domain experts suggests that both consider *Noise Simulator* a very helpful tool to facilitate communication with each other. Even so, further and more in-depth evaluations are necessary to establish this software's applicability to a variety of scenarios and users.

The use of point sound sources allows the simulations and visualizations to capture the *dynamics* of noise propagation, which state-of-the-art simulation software like Immi or CadnaA,

at the time of writing, were unable to. This was made possible by moving the computation process to the GPU using NVIDIA's CUDA architecture [26]. The resulting speed-up is more than sufficient to allow for real-time rendering times of even moderately complex noise scenarios, involving hundreds of point emitters, buildings, noise screens and other obstacles in large-scale areas. Immi or CadnaA perform several orders of magnitude slower, but a direct comparison is, due to formula simplifications on our part, not fair and has thus been left out. These simplifications and adaptations were necessary to make a GPU port of ISO 9613-2's formulas feasible to begin with, the most prominent being the complete disregard of noise reflections. To ensure the validity of the modified formulae, simulations of three very different real-world scenarios have been evaluated and compared to available ground truth data. The results are very promising and consistent across all scenarios. Note, however, that ISO 9613-2 does not make any accuracy claims for 'complex' scenes involving sound diffractions and the like. It is very likely that scenes can be constructed where ISO 9613-2 - and consequently *Noise Simulator* - fails. This, however, is a problem of the underlying noise propagation model and not *Noise Simulator* itself. Rapid computation on the GPU as shown in Section 2.3 relies on the problem being dividable into a grid of (independent) computations. This tacitly assumes that noise propagates along geometric lines, a simplification that is made by many noise models but nonetheless may be subject to change in future revisions.

The most time-consuming part of noise simulations is undoubtedly the acquirement of the input data. Noise measurements as well as supplemental information such as traffic counts and environmental data like temperature and humidity values require long-term observations and the use of expensive equipment. A simulation can only be as good as its input data and as such, greatest care must be taken to provide correct and valid simulation parameters.

In Chapter 7 we have discussed how noise visualizations can be used to analyze and evaluate both real-world and hypothetical scenarios. These visualizations may heavily factor in the decision making process and can also help in the development phase of projects. Although creating visualizations requires the use of domain knowledge, the results may be communicated to the general public. It should be noted that visualization as well as simulation results created by *Noise Simulator* are not intended to be used by either side of a conflict between affected neighbors and project solicitors and/or local governments to enforce their point of view. The aim is to provide a tool to mediate these conflicts and effectively reduce the number of legal cases arising as a result of these disputes. Naturally, *Noise Simulator* only provides the *technical* prerequisites of mediation and not the *social* ones. The latter are and always will be the responsibility of the conflicting parties.

8.2 Future Work

Although *Noise Simulator* can already be applied to a large variety of different scenarios, there are still many areas to improve upon. Probably the most prominent limitation and subject of future work is the incorporation of reflection sound sources as specified in ISO 9613-2. As an intermediary step (and an additional feature) the implementation of oriented sound sources will

be necessary. The application also very heavily relies on large amounts of GPU memory, which effectively limits the size of possible scenarios. However, since nowadays many GPUs provide one GB of memory or more and typical scenario sizes rarely exceed 4 km², we did not find this to be a serious problem. Of greater interest is probably the adaptation of more complex traffic models which would allow *Noise Simulator* to not only simulate specific time frames, but rather the average immission over extended periods of time.

Bibliography

- [1] IEC 61672-1. Electroacoustics - sound level meters - part 1: Specifications. 2002.
- [2] Arthur Appel. Some techniques for shading machine renderings of solids. In *Proceedings of the April 30–May 2, 1968, spring joint computer conference*, pages 37–45. ACM, 1968.
- [3] Jens Bellmann, Frank Michel, Eduard Deines, Martin Hering-Bertram, Jan Mohring, and Hans Hagen. Sound tracing: Rendering listener specific acoustic room properties. *Computer Graphics Forum*, 27(3):943–950, 2008.
- [4] Martin Bertram, Eduard Deines, Jan Mohring, Jevgenij Jegorovs, and Hans Hagen. Phonon tracing for auralization and visualization of sound. In *Proceedings of the Conference on Visualization, VIS '05*, pages 151–158. IEEE Computer Society Press, Oct. 23-28 2005.
- [5] Frank Brittain and Marlund Hale. Issues in using ray-tracing software for predicting community noise from power plants. In *INTER-NOISE and NOISE-CON Congress and Conference Proceedings*, volume 1, pages 208–217. Institute of Noise Control Engineering, 2003.
- [6] Deutsche Bundesbahn. Akustik 03: Richtlinie zur Berechnung der Schallimmissionen von Schienenwegen (Schall 03). *Bundesbahn Zentralamt München, München*, 1990.
- [7] Bundesministerium für Verkehr, Innovation und Technologie. Abteilung II/Infra 5. Verkehr in Zahlen. Österreich. Ausgabe 2011, 2012.
- [8] Bundesrepublik Österreich, Bundesministerium für Verkehr, Innovation und Technologie. Dienstanweisung 'Lärmschutz an Bundesstraßen (Autobahnen und Schnellstraßen)', GZ. BMVIT-300.040/0003-II/ST-ALG/2011, 2011.
- [9] Dae Seung Cho, Jin Hyeong Kim, Tae Muk Choi, Byung Hee Kim, and Douglas Manvell. Highway traffic noise prediction using method fully compliant with iso 9613: comparison with measurements. *Applied Acoustics*, 65(9):883 – 892, 2004.
- [10] Owen Cramer. The variation of the specific heat ratio and the speed of sound in air with temperature, pressure, humidity, and co concentration. *The Journal of the Acoustical Society of America*, 93:2510, 1993.

- [11] DataKustik. CadnaA - state-of-the-art noise prediction software. <http://www.datakustik.com/en/products/cadnaa>. Accessed: 2013-01-06.
- [12] Randolph W. Franklin. Pnpoly - point inclusion in polygon test. http://www.ecse.rpi.edu/Homepages/wrf/Research/Short_Notes/pnpoly.html. Accessed: 2013-01-06.
- [13] W. James Jr. Hadden and Allan D Pierce. Sound diffraction around screens and wedges for arbitrary point source locations. *The Journal of the Acoustical Society of America*, 69:1266, 1981.
- [14] Laiq Hasan, Marijn Kientie, and Zaid Al-Ars. Dopa: Gpu-based protein alignment using database and memory access optimizations. *BMC Research Notes*, 4(1):261, 2011.
- [15] International Standards Organization. *ISO 9613-1: Acoustics - Attenuation of sound during propagation outdoors - Part 1: Calculation of the absorption of sound by the atmosphere*. ISO, Geneva, Switzerland, 1993.
- [16] International Standards Organization. *ISO 9613-2: Acoustics - attenuation of sound during propagation outdoors - Part 2: General methods of calculation*. ISO, Geneva, Switzerland, 1996.
- [17] Henrik Wann Jensen. Global illumination using photon maps. In *Rendering Techniques' 96*, pages 21–30. Springer, 1996.
- [18] Sami Khoury, Adrian Freed, and David Wessel. Volumetric modeling of acoustic fields in cnmat's sound spatialization theatre. In *Proceedings of the Conference on Visualization, VIS '98*, pages 439–442. IEEE Computer Society Press, Oct. 18-23 1998.
- [19] Khronos Group. Opencl - the open standard for parallel programming of heterogeneous systems. <http://www.khronos.org/opencl/>. Accessed: 2013-01-06.
- [20] Jørgen Kraugh, Svein Storeheier, and Hans Jonasson. Nord2000, comprehensive outdoor sound propagation model-part 2: Propagation in an atmosphere with refraction. *Delta Acoustics for Nordic Noise Group, Report AV*, 1851.
- [21] Land Steiermark - Amt der Steiermärkischen Landesregierung. GIS Steiermark. <http://www.gis.steiermark.at/>. Accessed: 2013-01-06.
- [22] Chi-Wing Law, Chee Kwan Lee, and Moon Kwong Tai. Visualization of complex noise environment by virtual reality technologies. Technical report, Environment Protection Department (EPD), Hong Kong, 2006.
- [23] Frank Michel. *Simulation and visualization of in-and outdoor sound*. PhD thesis, Technical University Kaiserslautern, 2008.
- [24] Ulrich Möhler and Manfred Liepert. Die Ermittlung der Schallausbreitung nach der neuen Schall 03. *Fortschritte der Akustik*, 31(2):527, 2005.

- [25] NVidia Corporation. Cuda C programming guide. <http://docs.nvidia.com/cuda/cuda-c-programming-guide/>. Accessed: 2013-01-06.
- [26] NVidia Corporation. Cuda parallel computing platform. http://www.nvidia.com/object/cuda_home_new.html. Accessed: 2013-01-06.
- [27] NVidia Corporation. Cuda samples. <http://docs.nvidia.com/cuda/cuda-samples/>. Accessed: 2013-01-06.
- [28] NVidia Corporation. Nvidia cuda compiler driver nvcc. <http://docs.nvidia.com/cuda/cuda-compiler-driver-nvcc>. Accessed: 2013-01-06.
- [29] Hyun Kyu Park, Yang Woo Lee, Sea Il Lang, and Im Pyoung Lee. Online visualization of urban noise in ubiquitous-city middleware. In *Proceedings of the 12th International Conference on Advanced Communication Technology, ICACT '10*, pages 268–271. IEEE Press, Feb. 7-10 2010.
- [30] David Parzych. Handling of barriers in iso 9613-2. In *Proceedings of Noise-Con*, 2004.
- [31] Berger Plovsing and Jorgen Kragh. Nord2000. comprehensive outdoor sound propagation model. part 1: Propagation in an atmosphere without significant refraction. *DELTA Acoustics & Vibration Report AV*, page 2001, 1849.
- [32] Wolfgang Probst, Heinrich A Metzen, and Ingo Rabe. Noise prediction for industrial facilities. *Journal of the Acoustical Society of America*, 127(3):1766, 2010.
- [33] Mark Segal, Carl Korobkin, Rolf Van Widenfelt, Jim Foran, and Paul Haeberli. Fast shadows and lighting effects using texture mapping. In *ACM SIGGRAPH Computer Graphics*, volume 26, pages 249–252. ACM, 1992.
- [34] Svein Å Storeheier. Nord2000: Sound scattering outdoors. revised simple models. Technical report, SINTEF Memo 40-N0 990003, Trondheim, 1999.
- [35] Jantien Stoter, Henk De Kluijver, and Vinaykumar Kurakula. 3d noise mapping in urban areas. *International Journal of Geographical Information Science*, 22(8):907–924, 2008.
- [36] The FLTK Project. The Fast Light Toolkit. <http://www.fltk.org/index.php>. Accessed: 2013-01-06.
- [37] Lee Thomason. TinyXML. <http://sourceforge.net/projects/tinyxml/>. Accessed: 2013-01-06.
- [38] Nathan Whitehead and Alex Fit-Florea. Precision & performance: Floating point and iee754 compliance for nvidia gpus. *NVidia Technical White Paper*, 2011.
- [39] Woelfel Group. Immi - the noise mapping software. <http://www.woelfel.de/en/products/modelling-software/immi-noise-mapping.html>. Accessed: 2013-01-06.

- [40] Denton Woods. Devil - a full featured cross-platform image library. <http://openil.sourceforge.net/>. Accessed: 2013-01-06.
- [41] Diange Yang, Bing Li, Ziteng Wang, Sifa Zheng, and Xiaomin Lian. Video visualization for moving sound sources based on binocular vision and short-time beamforming. *Noise Control Engineering Journal*, 58(4):382–388, 2010.
- [42] Takatoshi Yokota, Shinichi Sakamoto, and Hideki Tachibana. Visualization of sound propagation and scattering in rooms. *Acoustical Science and Technology*, 23(1):40–46, 2002.