Multimodal Technologies and Interaction

Article

Combining VR Visualization and Sonification for Immersive Exploration of Urban Noise Standards

Markus Berger * **and Ralf Bill**

Faculty of Agricultural and Environmental Sciences, University of Rostock, Justus-von-Liebig-Weg 6, 18051 Rostock, Germany; ralf.bill@uni-rostock.de

* Correspondence: markus.berger@uni-rostock.de; Tel.: +49-381-498-3207

Received: 28 March 2019; Accepted: 7 May 2019; Published: 13 May 2019

Abstract: Urban traffic noise situations are usually visualized as conventional 2D maps or 3D scenes. These representations are indispensable tools to inform decision makers and citizens about issues of health, safety, and quality of life but require expert knowledge in order to be properly understood and put into context. The subjectivity of how we perceive noise as well as the inaccuracies in common noise calculation standards are rarely represented. We present a virtual reality application that seeks to offer an audiovisual glimpse into the background workings of one of these standards, by employing a multisensory, immersive analytics approach that allows users to interactively explore and listen to an approximate rendering of the data in the same environment that the noise simulation occurs in. In order for this approach to be useful, it should manage complicated noise level calculations in a real time environment and run on commodity low-cost VR hardware. In a prototypical implementation, we utilized simple VR interactions common to current mobile VR headsets and combined them with techniques from data visualization and sonification to allow users to explore road traffic noise in an immersive real-time urban environment. The noise levels were calculated over CityGML LoD2 building geometries, in accordance with Common Noise Assessment Methods in Europe (CNOSSOS-EU) sound propagation methods.

Keywords: virtual reality; immersive analytics; multisensory; sonification; urban planning; environmental noise

1. Introduction

Urban noise maps today are mostly presented in a classic 2D map format, sometimes as static images, sometimes within mapping applications that allow panning, zooming, and other interactions. For most day-to-day tasks, this is appropriate and useful and will likely remain the standard for the foreseeable future. This basic approach is sometimes extended to 3D representations, in order to show data that a traditional top–down view could not visualize, like a gradient of noise extending over vertical building walls [1].

The practicality of these maps does not imply that there is no use for other representations. Classic 2D and 3D maps can by necessity only show certain configurations of parameters—a certain time of day, a specific noise measure, often at only one resolution. Deeper exploration and understanding of data usually requires some sort of interactivity. At a basic level, this can be introduced by letting the user select between multiple different layers that each represent a specific scenario. However, that solution is, at its core, still a very static one. Getting a truly dynamic view of the noise situation in an urban area would require full control over all parameters and the ability to quickly recompute results. This is not an easy challenge to overcome, as noise propagation computations involve tracing a large number of propagation paths through the environment and then require multiple complex calculation steps for each of these paths.

Even if real-time computation was achieved, the simulations methods utilized are still highly abstracted. Often, users can be confronted with noise maps that claim to represent reality without any indication about the inaccuracies and concessions that every usable urban noise model necessarily has. (See [2] for some ideas on a theoretical idea model.) Even given a hypothetical perfect calculation model, nonsubject matter experts might be misled by the results. Annoyance and health impacts are not a function of any absolute value that can be depicted on a map. Soundscape research has uncovered many psychoacoustic complexities over the years. For example, some people will indicate annoyance with a noise situation up to ten decibels later than others, people tend to be less annoyed when they are economically dependent on the source of the sound, and there are even cultural differences in what sounds are seen as annoying in the first place [3]. Purely visual representation of acoustics also has the problem that our auditory perception is very different from our visual perception: Reference [3] describes that we perceive sound as passive and emotion-rich, while visual stimuli are more active and information-rich. Thus, a visualization approach that is not only visual but also utilizes sound has the chance to represent some of these perceptual differences, even after the data have been transformed and filtered for the purposes of making them more comprehensible to the user.

These ”multisensory” visualization approaches are often enabled by immersive technologies such as virtual reality (VR) systems, which can track user movements, show a virtual environment wherever the user is looking, and play spatialized sounds according to their head position. Because of these capabilities and their recent surge in availability, the use of VR technologies is currently an active topic in soundscape research. Used correctly, they might offer great benefits, especially in terms of user engagement. Great care is taken to present believable environments in such a way that they induce a sense of presence in users while they are shown the influence of urban noise or plans to deal with that noise [4,5].

The sounds that are played back to the user are usually derived from field recordings with calibrated sensor equipment and thus have a high degree of accuracy. Reference [6] uses such recording equipment to record the noise situation in a wind park and investigates how a multisensory virtual reconstruction compares to video and audio of the real site when measuring a user’s subjective responses to wind turbines in the environment. Even though they are not using an immersive display system, the authors show that for factors like annoyance, there is significant overlap between simulation and video. For the immersive case, reference [7] also tackles a wind park scenario but tries to utilize the higher level of presence offered by current VR technology to measure not only subjective but also cognitive and affective impacts. While much of this research is preliminary, the results are promising.

Compared to some of these approaches, we want to take a step back to the earlier point of simulation accuracy and utilize VR technology to immerse a user not in an environment that is as close to reality as possible, but in the environment as the noise propagation algorithm ”sees” it. The user will be able to investigate which parts of his surroundings influence the way noise measures are calculated, showing what the result of certain changes could be, but also laying bare where these models may fail. The system will make use of the multisensory nature of current VR technology and let users listen to an approximate audio rendering of their current situation under the parameters they choose.

2. Materials And Methods

2.1. Data Preparation

We utilized a LoD2 CityGML dataset of Rostock, Germany to model our buildings. Even though software support for CityGML is sometimes lacking, there are simple ways to transform its boundary representation (B-Rep) model into a polygon mesh that is usable by modern game engines. The problem is that this step usually removes any metadata. We wanted to circumvent this to retain all the useful information that a CityGML dataset may have for accurate noise simulations, like information about building materials and absorption coefficients defined by the NoiseADE extension. To achieve this, we wrote a Python script to split the city-wide dataset into individual buildings. Then we converted each building into a COLLADA 3D model and gave it the same name as the CityGML file. We then loaded both files for each building into Unity, where we could now query for additional information simply by loading the same resource name with a different file ending.

Within Unity, we made use of the Mapbox mapping extension [8]. It supports dynamic loading of terrain height maps and a collection of basemaps and handles the accuracy problems that we would usually encounter when using georeferenced data in 32-bit floating point accuracy. The problem is that Mapbox in Unity only works with the popular Web Mercator projection instead of arbitrary reference systems. While that precision is fine for placement of buildings on a street (the noise simulation itself only runs at a limited accuracy), reprojecting our CityGML data point-by-point introduced noticeable distortions within the individual building geometries, especially in multipart buildings. To fix this, we introduced a reference point into each CityGML building file by selecting the highest corner point. If multiple points are at the same height, we take the northernmost and then the easternmost of that subset. The same Python script from earlier selects this point, reprojects it to Web Mercator (EPSG:4326), and writes it into the file. Later, in Unity, we found that same point in the mesh data and can now place the buildings according to the correct projection, while retaining all the accuracy in the geometry itself.

2.2. Hardware

Mostly driven by the gaming industry, there is a plethora of commodity VR hardware on the market today. The idea realized in this paper is in principle not limited to any one system, but the choice of hardware sets the performance and interaction constraints for the implementation. To cover as many use cases as possible, we decided to use a modern head-mounted display, which typically does not require an expert on-site to set up the system. Other options can be found in [1,9]. The former describes projector-based stereo viewing of noise data, while the latter a CAVE system with a main focus on accurate audio playback.

Once in the head-mounted display (HMD) product space, the most important choice is whether tracking is done in three or in six degrees of freedom (DOF). While a 6-DOF system might increase presence and interaction capabilities, especially because current 6-DOF systems usually feature two motion controllers for hand tracking, the 3-DOF system lowers the barrier of entry, both in terms of price and the amount of set-up and instruction required before use. The latter also corresponds well with the limitations set by the noise standard: Noise has to be recomputed for each position, and in a 3-DOF environment, this does not continuously change due to body movements; instead, it only does so at the users prompt. Furthermore, noise results in the Common Noise Assessment Methods in Europe (CNOSSOS-EU) standard are said to be only valid for observer points that are two meters or more above the ground [10]. If we want the user position to stay close to their real-world height, a 6-DOF system would allow them to duck down into a nondefined area. In a 3-DOF system, we can instead set their virtual head at two meters high and keep it there. 3-DOF systems are also more affordable, do not require the user to move around, and thus currently have the lowest barrier of entry into the VR space. Recently, they have even been utilized and validated for other VR soundscape research [11].

2.3. Input & Interactions

The problem with 3-DOF HMDs is that they are limited in what kind of interactions they allow. Some 3-DOF VR systems come with their own controller, while others rely fully on the HMD itself as an input modality. In the latter case, the common interaction methods are viewing direction combined with a single button or a touchpad. This will hardly be enough to cover all the interactions that are needed. To avoid visual clutter and distraction through unwanted audio cues, the user needs to be able to dynamically switch certain features on and off. We identified the following basic set of interactions:

1. Movement through the virtual environment (locomotion);
2. Switch the noise map on/off;
3. Switch the propagation visualization on/off;

4. Play the current noise level;
5. Allow the user to set a multitude of parameters.

It is evident that we need more than one button/touchpad to cover these use cases if we do not want to force the user to do an excessive amount of head gestures. Thus, we either have to use a system equipped with a controller or use voice recognition. While the latter is an interesting research direction, we settled on the controller-based approach, as there is already a large body of research and consumer experience that has accumulated over the years.

For the actual implementation, we utilized a 3-DOF tracked motion controller as found in common standalone VR systems like GearVR. Other types of controllers, even those without motion tracking, should also be able to handle the required interactions without much issue.

The first interaction on our list is simultaneously the most critical for any VR application: User movement and agency, ranging from simple head rotations to continuous movement in large environments. For that reason, we look at this interaction in more detail. Different techniques have been extensively researched, especially since the resurgence of consumer-level VR HMDs over the last few years. Some require the user to actually walk in the physical world, others are based on keyboard, joystick or motion controller input. Reference [12] discusses the most common of these approaches in detail and presents a point and teleport system. Here, the user does not have to move through space and is instead directly snapped to a location that is selected with a motion controller that acts as a pointer. The experimental data show that this is a low-friction method that most users enjoy working with. It also remains usable even when seated, as it does not require anything but arm movement and, depending on the specific implementation, body rotation, which can be accomodated with a swiveling chair.

For our purposes, the main contender of the point and teleport method is joystick locomotion, which continously moves the user position in the direction indicated with the joystick. Reference [12] shows this to be a viable approach. The reason we settled on teleportation is performance-related—we only have to recompute user noise levels after every snap, whereas for joystick locomotion, it would have to be continously updated in real-time. We do not believe that the advantages of joystick locomotion outweigh this advantage in performance.

Following the results from [12], we settled on a point and teleport system with a visual indicator, in which the user can decide whether they want to keep their rotation or realign their view on teleport. Because of how common this form of locomotion is in current VR applications, we did not have to implement this interaction ourselves. Almost every VR framework contains a prototypical rendition of this interaction, which also means that many users will be immediately familiar (see Figure 1).

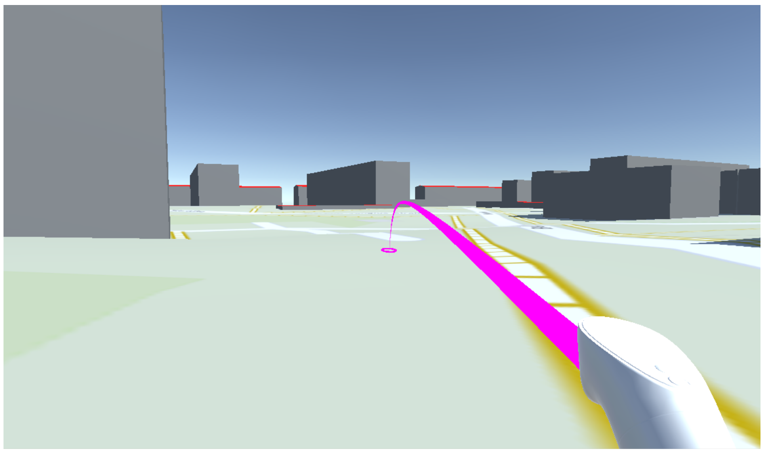

Figure 1. Standard VR point and click teleportation in our city model.

One problem with common teleportation approaches is the lack of altitude control. In joystick locomotion, the user can easily launch into flight and explore the environment at different heights.

In terms of directly experiencing the noise data, this would not offer large benefits, as noise maps are usually calulated as a 2D grid at a fixed height over the ground plane and vertically over every building surface, not in a full 3D volumetric fashion. What upward movement could enable is easier navigation through the urban environment, especially as the user could simply hop on or over buildings that are in the way. We found that users get easily lost in the highly abstracted, very uniformly rendered environments, even if they had extensive knowledge of the areas being displayed. It thus became a necessity to offer some sort of overview map to regain spatial context throughout the exploration.

To our knowledge, there are no current studies testing the effectiveness of different exploration aids for larger, city-scale immersive virtual environments from a first-person perspective. There exists some research in the Augmented Reality (AR) space concerning how city environments can be explored, but here, the navigation tasks are usually limited to wayfinding and navigating to a specific position, as seen, for example, in [13]. We require a method that helps with free exploration of the virtual urban space, while maintaining a focus on the data.

For this, we identified the following methods, most of them common interactions from non-immersive mapping appications:

1. A minimap, or any kind of overview shown in addition to the actual surroundings;
2. Upward teleportation, or any way to change player position away from the ground plane;
3. Zooming out, which in a VR context would equal a change of scale for either user or environment;
4. Environment deformation, like bending terrain upwards at a distance (see [14]).

Comparing these approaches might be an interesting direction for future research. For our purposes, option four has a relatively high implementation overhead. Option one opens up a host of other questions, such as how to place the minimap, whether and how to present the 3D data on this representation, and how to interact with it. There is no immediate disadvantage apparent in options two and three. We settled on a variant of option three, an immediate snap of the current map extent to a lower scale, because it requires only one input action and should be easy for users to understand (see Figure 2).

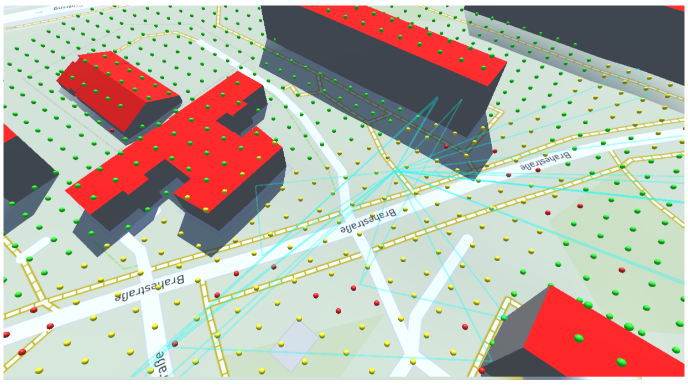

Figure 2. User perspective after scale change. Propagation lines are shown in the last real-scale position, so that propagation can be studied in a larger spatial context.

A side note that may be interesting for some readers: During the development of Google Earth VR, the designers grappled with similar problems and held a talk about their results [15]. Their interaction design contains a combination of options one, two, and three and is generally well regarded by users.

2.4. Cnossos-Eu in Unity

Now that we have an environment that the user can explore and navigate, we need to populate it with the noise data. The CNOSSOS-EU [10] standard describes a 2.5D method to compute noise propagation but mentions at several points where a 3D implementation is possible or necessary. In Unity, we have the choice between using a 2D and a 3D physics engine. In order to correctly handle different roof shapes, we settled on using the 3D physics and adapting the 2.5D method as accurately as possible.

The foremost problem in implementing these standards is the pathfinding for the noise propagation paths. Reference [10] defines four types of paths—direct paths with diffractions over horizontal edges, reflections on vertical walls, diffractions on vertical edges, and a combination of the latter two paths. Reference [16] describes how to efficiently construct noise propagation paths of the first type over LoD2 CityGML data and notes some difficulties with the other path types. Accurate handling of reflections and diffractions is indeed costly, not always solvable in the general case, and requires some knowledge about surface materials. Still, we wanted the user to have the option of including them, especially because they can potentially improve the realism of the sound simulation.

Considering that the large majority of urban buildings have a very simple shape in LoD2 data and are made out of some form of concrete or steel, we were able to implement a heuristic approach that manages to construct all four types of paths in most cases. A rendering of a common propagation situation is depicted in Figure 3. For performance reasons, we only allowed specular, first-order reflections. The way we achieve this is by casting a ray from observer to source. If the ray hits an obstacle, we fire three new rays, the target of each shifted one meter left, right, or up from the obstacle impact point. If they miss the obstacle at that point, we have a diffraction and can retry the raycast from there. If they still hit the obstacle, we shift them further in their direction. This only assumes that the user is not surrounded by geometry on all sides (to handle that case, we limit the number of steps in each direction) and usually constructs two vertical and one horizontal diffraction if no direct path is present. (We could then make another assumption about building shape and construct a simple rectangle of paths to fill out the spaces between the three that we have, though this is not currently included.) Reflections are handled by raycasting in every direction (at one degree steps) on a plane between source and observer (this is where we utilize the 2D nature of the standard—full 3D reflections would increase the complexity drastically). At the impact point, we tested whether the direction of the target and the impact angle would end up describing a specular reflection. We allowed some room for error in this test, so we do not miss too many reflections, which would have happened between two angle steps.

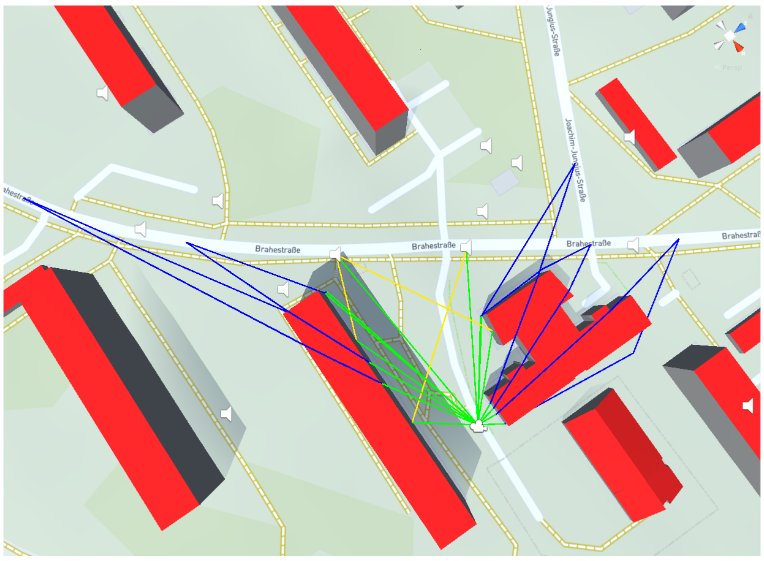

Figure 3. Noise propagation in a street with one observer point and seven source points. Green lines are direct paths and path end segments. Blue represents paths that lead to diffraction around buildings, and yellow paths that cause reflections on vertical building surfaces. The white speaker symbol shows the image sources vertically projected to the ground. (Otherwise, some would end up above the camera.) The observer is represented by a camera, the source points are located at the beginning of the line segments.

This approach finds its way around most building shapes. What it does not do is conform precisely to the roof shape—the very reason we are utilizing CityGML LoD2 data instead of the less performance-intensive LoD1. To make use of roof shapes, we moved from full 3D space to 2.5D space. We traveled along every path and raycast downwards at a regular interval. This turns every path into a 2D approximation of the slice of ground that it travelled over. Reflections and vertical diffractions are simply folded open. The result conforms exactly to the way propagation paths are described in [10]—the intersection between the terrain and a vertical plane that is spanned between observer and source points. Although we do not currently include it in our prototype, because our data do not carry this information, this is also the step where information about ground materials would be collected.

After we defined all the propagation geometry, we computed the sound pressure level (SPL) for each path. For this, we first took the sound power level of the source point (which depends on the type of vehicle, see [10] for details) and then calculated several attenuation coefficients as shown in Equation (1):

$$L_H = L_{W,0} - A_{div} - A_{atm} - A_{boundary,H}, \tag{1}$$

where L_H is the SPL in homogeneous conditions, $L_{W,0}$ is the sound power in a given frequency band (we currently exclude both favourable conditions and directionality), A_{div} is the attenuation due to geometrical divergence, A_{atm} is the attenuation due to atmospheric absorption, and $A_{boundary,H}$ is the attenuation due to ground and/or diffraction (depending on path type) in homogenous conditions.

Once the SPL for each path and octave band is calculated, everything can be summed up and A-weighted to get the total sound pressure at an observer point. Details about how to do these summations can be found in [10].

2.5. Visualization

In accordance with the previous sections, we have to visualize three sets of data:

1. The noise levels over the terrain;
2. The propagation paths used in the noise calculations;
3. The terrain and city model.

There are several approaches commonly used for mapping noise or other continuous data over 3D city environments and terrain. Reference [1] shows a system for an immersive context, which interpolates results and textures the building facades accordingly. This does not extend to the terrain. Reference [17] extends a similar visualization over the terrain but does not specifically focus on VR. Reference [18] uses 3D contour lines to cover terrain, facades, and rooftops to arrive at a similar result. It is questionable whether an environment uniformly covered with data would be easy to navigate in immersive virtual reality. Instead, we want to retain as much spatial context as possible. Ideally, the user should be able to see streets and some terrain information on the ground.

With that limitation in mind, we found it a better idea to look at AR visualizations, as they, by their nature, have to preserve some degree of spatial context. Situated Analytics is a rising research topic grappling exactly with these issues. Current research is limited but often focuses on scatterplots in a volumetric space, for example, [19,20]. This kind of visualization serves our purpose well, as the actual output of the noise standard used here is a grid of points in 3D space and thus shares some of the attributes of immersive scatterplots, e.g., depth, occlusion, point representation, and more. Visualizations like the ones shown in [1,17,18] are interpolations over this point grid and try to give an overview of a noise situation, while in our case, we want to only show exactly what is being calculated by the standard.

Showing several different techniques, reference [19] concluded that many perceptual questions remain open in how exactly to display volumetric data in an immersive environment. What an ideal visualization for our case would look like will hopefully be answered by future immersive analytics research. For now, we use simple spheres draped over the terrain that represent a grid of observer points. We colored them to represent the level of annoyance their output values indicate (see Figure 4).

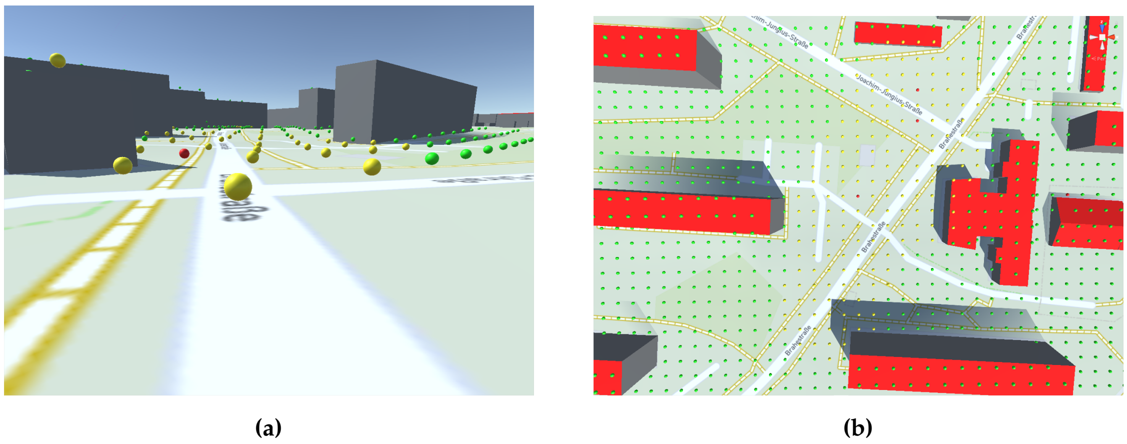

Figure 4. (**a**) View on the noise grid from the user's perspective; (**b**) top–down view on the noise grid.

For point two and point three, we remained as close to the actual geometry as possible. Propagation paths are transparent 3D lines, and the terrain is textured with a thematic street map for further geospatial context. In the current prototype, the terrain information was supplied by the Mapbox extension.

2.6. Sonification

In addition to the visualization, we wanted to make use of the multisensory nature of VR and let the noise data be audible. This process is called sonification, as defined in [21]. While in our case, the mapping from data input to sound output seems very straightforward, as we are essentially turning a virtual dB value into a real output dB value, there are a few things to consider. First, the dB levels that leave the user's speakers or headphones are influenced by many hardware and software factors. Without calibration for one specific system, we can never be sure whether input and output sound level are even close to each other. Our usage of a standalone VR headset makes this calibration process easier, as there are fewer unknown factors (like having only one systemwide volume control), and we might be able to carry over calibration results between multiple headsets of the exact same type with some reasonable accuracy. On the other hand, sonification can be useful for the user even without calibration—it allows them to compare relative values while traveling through the environment, without the need for additional visual feedback. In this case, we mapped the differences in the input dB values to differences in the output dB values.

Related to mapping accuracy is the question of spatialization. Modern game engines integrate complex 3D sound frameworks that allow factors like room reverberations, doppler effects, and attenuation through obstacles to influence the sound playback. We do not want to make use of most of these features, as they are either rough approximations of real sound propagation effects that we already simulated by taking into account the full scene geometry, or they distort our results for cinematic effect. Instead, we utilized the most basic level of audio spatialization offered in modern game engines. In Unity, this includes a nonpersonalized binaural head related transfer function (HRTF) that makes use of the sound position in relation to the listener. In the case of HTRFs, spatialization of the sound is achieved through filtering of the audio signal for each ear in such a way that important real life cues are simulated: Mentioned in Unity's manual [22] specifically are microdelays between both ears and filtering of sound waves as they travel through the human head. In order to make use of this spatialization, we need the correct positioning and directionality of both the audio source and the user's head. The latter is known to us because of the VR HMD's head tracking, while the computation of the former is already included in the noise standard in the form of image sources [10]. For each image source in the standard, we positioned one audio source in the virtual environment.

Next, we needed to set the volume for each source. One issue here is that the attenuation of the sound level over distance is part of both the noise standard and Unity's audio spatialization. The noise standard calls this *geometric divergence* and calculates it according to Equation (2), where A_{div} is the geometric divergence and d is the distance from source to receiver [10].

$$A_{div} = 20 * lg(d) + 11 \tag{2}$$

Meanwhile, Unity uses a curve that models distance to a factor that is applied to the sound volume. Fortunately for us, the divergence described in Equation (2) describes a simple logarithmic curve that can be adapted as a distance falloff curve. To utilize this, we removed the geometric divergence from the final noise calculation result as soon as it was supposed to be played back to the user. This does not result in the exact output value, especially because the divergence is subtracted from the end result, while the falloff curve is a multiplier. However, if the curve is set with care, then this should lead to a result that corresponds reasonably well with reality for the dB(A) value range we expect in urban noise environments.

If absolute precision is required, then some engines, including Unity, have the capability to implement a custom audio spatializer with customizable distance falloff functions. Such an implementation would have to go hand in hand with extensive calibration of the HMD's audio output.

Once we removed the geometric divergence and A-weighted the resulting sound pressure levels, we had the final noise that we wanted to play back to the user. To achieve this, we transformed the dB value into a linear value between zero and one, to conform to the way Unity represents volume. In our prototype, we did this by converting the SPL back to sound pressure. Sound pressure is what the A-weighted SPL that we are using is derived from. All the dB values we are using here are in reference to 20 µPa, which is the minimum sound pressure in Pascal (Pa) that a young ear can detect [23]. To get to our linear value, we converted our SPL, which if written in full has the unit ″dB re 20 µPa″, back to Pa. We then divided it by two, as shown in Equation (3). The division by two is done so that 100 dB maps to one. We do not expect much higher sound pressure levels than 100 dB in traffic noise situations, so we simply turned every value that goes above 100 back to one.

$$S_i = \frac{p_i}{2} = \frac{10^{\frac{(L_{H,i}+A_{div,i})}{20}} * 0.00002}{2}, \tag{3}$$

where S_i is the in-engine volume of the image source i and p_i is the sound pressure of the image source in Pascal.

At this point, we only need to select an appropriate audio clip (for example of a passing car) and play it with calculated volume level.

This will not output the actual physical SPL value in the user's headphones. If this is the goal, then calibration data should be taken into account in this step. The used clip should also be a recording made with specialized equipment, or entirely synthesized.

If we instead want to move further in a cinematic direction, we could now apply appropriate filters to the audio clip, as we know how many reflections and diffractions each path went through. This would of course further alter our results in ways not always predictable—in order to increase immersion and realism with what we have, and without any further distortion, we delay audio playback by the time it would take sound to travel over the whole path. This way, diffractions and reflections play later than direct hits and create reverberations and echoes without any audio filtering. Especially diffractions and reflections on vertical edges may not contribute much to the total noise level at an observer point [10], but they can drastically change what the user hears in an environment with many buildings.

3. Discussion

3.1. Performance & Usability

The basic problem of the prototype described in this paper is how it tries to bring together two worlds: The CNOSSOS-EU [10] standard, which like all environmental noise standards was not made with real-time capabilities in mind, and VR, where every choice has to respect the performance constraints. The goal here was not to explore optimizations for the algorithms of the noise models themselves. Thus, we tried to stick to a ″naive″ implementation that utilizes the calculation steps as described in the standard.

Because of this, the current prototype does not reach the framerate targets that are common wisdom in entertainment-focused VR, which are generally set between 60 and 90 frames per second. While scientific research generally adopts these measures as sensible guidelines, it shows a more complicated, inconclusive picture of VR latency and cybersickness [24,25]. At the same time, compensation routines like asynchronous time warp [26] have become very adept at managing some forms of performance issues, and the constraints we have set are (on purpose) very conducive to these methods—the user is only rotating, and the bulk of the environment remains static. In fact, in most situations, the teleportation indicator was the only object that obviously showed latency. Stronger performance drops that occur during user movement (as the noise level has to be recalculated) are mostly masked by teleportation, as the sudden snap movement covers frame drops to some degree.

Still, to inform future developments on our our prototype, we wanted to investigate its performance more closely. First, we timed how long the application needed to compute three different noise grid sizes. Then we went through two scenarios for each size: Once with the grid off and once with the grid on. We placed the user at the center point of the grid, then we made them slowly turn in place for a few seconds, before letting them randomly teleport around. The results can be found in Table 1.

Table 1. Startup time, as well as mean and standard deviations in frame rate for different sizes of the precomputed noise data grid on an Oculus Go 64GB, for ten source points. Loading the application, map, and city model takes about 20 s; the rest of the startup time is due to noise calculations.

Observer Points	Calculation Time	Frames per second Grid not Shown		Frames per second Grid Shown	
		μ	σ	μ	σ
10 × 10	~35 s	48.9	8.7	48.3	8.7
30 × 30	~140 s	48.4	8.3	40.3	5.9
50 × 50	~350 s	47.5	7.9	38.0	9.4

As expected, there were large differences in how long the precalculation takes. With the area of CityGML data used here (see Figure 5 for an overview), the startup time could be reduced to about 20 s by precomputing the grid of observer points and simply loading it at startup. Subtracting those 20 s from the startup time results in Table 1, thus also telling us how long the noise simulation for the observer grid takes in each configuration. For example, in the case of ten source points and with mobile hardware, one observer point takes roughly 150 ms to compute. This calculation has to be performed for each user movement, which means it simultaneously represents a delay in teleportation. For ten source points, this barely makes a difference, but achieving the density found in classical noise maps is currently not feasible. Some noise propagation calculations require dozens of these points for every road segment—our prototype currently cannot handle this for more than one short street.

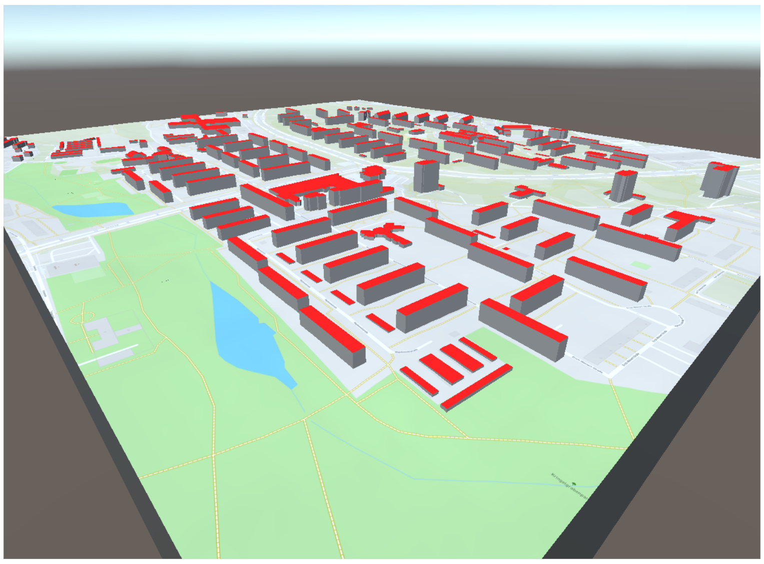

Figure 5. The segment of CityGML data used for testing this prototype.

Also expected was the behaviour of the mean frames per second—barely a change when the noise grid is not displayed (as the scenarios are the same in terms of graphics processing), and some loss of performance with increasing grid size when it is. This loss, though, was not enough to explain the often significant frame rate drops we experienced. Instead, what sticks out is the standard deviation of the frame rate over time—it stays at the same high level in all scenarios. The only factor that is present in all scenarios and experiences enough change to explain the deviation is the viewing direction. Thus, we found that the most significant contribution to performance drops comes from the number of CityGML buildings located in the viewing direction of the user.

If one is only interested in acoustic fidelity and truth to the standard is not a necessity, there is recent research covering how to do complex urban sound propagation calculations in real time, for example [27].

3.2. Limitations

In addition to the performance constraints, the current prototype has some technical limitations. While we tried to polish the experience where possible, the research nature of this prototype made it necessary to try to find an appropriate end point.

One of the more basic limitations and the largest barrier in terms of usability is our way of data preprocessing. Ideally, there would not be any reprojection and manipulation of the CityGML data through external tools. Instead, users should be able to load any arbitrary CityGML data into the application. To achieve this, reprojection and conversion from boundary representation into polygon meshes would have to happen in Unity itself. While not impossible, this would add a lot of technical complexity to the project.

Currently, the building data in the application are also entirely static. They are imported into the resources folder, where they are loaded at startup of the application. Once compiled, all the resources are part of the resulting VR app. For a more dynamic approach to CityGML data, see [28], where CityGML data are still preprocessed but can then be loaded as needed.

Another limitation is related to VR interaction design. While we settled on teleportation for various reasons, there is one argument against it that has not yet been conclusively investigated but might have large implications for its use in both immersive analytics and complex urban environments specifically: The possible loss of spatial and data context with every teleport. In joystick locomotion, the user can continuously walk around the data point and focus on it without suddenly changing position, while teleportation constantly introduces cuts into the user's perception. Should later research show this to be an important limitation in terms of usability, reference [29] offers some guidelines on how to counteract the problem, at least partly. They found that actively controlled, short-range teleportation with visual cues does not have as negative an impact on spatial awareness as long-range teleportation, while also better circumventing motion sickness in some subjects.

Lastly, while our program goes through a complete noise propagation simulation as laid out in the CNOSSOS-EU standard, the implementation itself is not complete. At various points, we assumed certain parameter configurations, usually because we lacked the data to make a more accurate assessment. One of the larger limitations is that conditons are always assumed to be homogeneous, not favorable. Favorable conditions require curved propagation paths and would add some more complexity when calculating diffractions. However, there is nothing preventing them from being implemented at a later date. The application also does not yet place source points by itself; instead, this is done by hand in the Unity editor. The source point placement method as described in [10] could be implemented by including data about roads (something the already included Mapbox extension is capable of) but would lead to a more dense distribution of points than the prototype can handle in real time (see Section 3).

3.3. Future Research

Some opportunities for future research were already noted in Sections 2.3 and 2.5: How to aid free navigation in large virtual environments and how to display volumetric data in them.

In terms of performance, there remains the goal of full real-time capability that lets the user move around continuously and at high performance. If only the user-centric noise level is updated in real time, this should be attainable. One measure would be to decouple the audio rendering completely from the geometric calculation and dynamically adjust how often the geometric situation is recalculated in the background. Some past discussion on performance in audio rendering can be found in [30]. The bigger problem is the raster of observer points. Real-time recalculation of the noise values to react to changes in the environment is not easily possible on current hardware with the current noise standards. Routines in the standard itself might have to be optimized, or calculations would have to be outsourced from the VR HMD.

In light of current soundscape research, the accuracy of calibration and sound playback can be improved—especially because currently, only the A-weighted sound pressure level (SPL) is modeled. There are many more subtleties to environmental noise, like different octave bands (which are already included in the calculations but cannot be isolated by the user yet). Sonification could be used to let the user highlight and explore noise by more than just volume.

For the research in this paper specifically, the most immediate next steps would be the expansion of the interaction space. The current foundation could be expanded so that users could manipulate the environment, actively preview the outcome of certain changes, and dive deeper into the calculations of specific propagation paths and observer points. Instead of just having a dynamic observer, there could also be dynamic sources—virtual vehicles that the user can configure and listen to as they move through the environment.

4. Conclusions

During this research, the road traffic noise propagation from the CNOSSOS-EU noise standard was successfully implemented into a mobile VR application in the Unity game engine for a certain set of parameters. In a limited, local scope, implementation works fast enough to comfortably run on a low-cost HMD like the Oculus Go. For large environments, the first components to optimize are the calculation of the observer points and the rendering of the city model.

The input side of our prototype consists of mostly tried and true interaction techniques and modalities. The user can look at their surrounding by rotating in three degrees of freedom, like in common panoramic picture or movie viewers. They can relocate themselve in the environment by teleportation facilitated through a motion controller that acts as a pointer.

Every time the user relocates, noise levels are recomputed on the fly. A way to convert CNOSSOS-EU sound pressure levels into an input representation compatible with Unity's audio spatialization was devised, so that the user can listen to a complex rendering of the soundscape around them. All four types of propagation paths defined in [10] were considered, which means that not just direct hits, but also reflections and diffractions around obstacles are rendered, something not usually handled by current game audio systems.

In order to show the user interesting locations to teleport to, the application precomputes a grid of observer points. If the current position features lots of occlusions, the user can change the scale of the environment and look at the whole city model as if it was a miniature model.

Author Contributions: Conceptualization, M.B. and R.B.; Data curation, M.B. and R.B.; Formal analysis, M.B.; Funding acquisition, R.B.; Investigation, M.B.; Methodology, M.B.; Project administration, R.B.; Resources, R.B.; Software, M.B.; Supervision, R.B.; Visualization, M.B.; Writing—original draft, M.B.; Writing—review and editing, R.B.

Funding: This research received no external funding.

Acknowledgments: The authors want to thank the Rostock cadastral office for creating and supplying the Rostock CityGML dataset. We also want to extend gratitude to Jian Kang and all the others that provided clues and discussion about soundscapes and acoustics research.

Conflicts of Interest: The authors declare no conflict of interest.

Data and Source Code Availability: The Rostock LoD2 CityGML building models used here cannot be made public. However, LoD2 CityGML data are freely available for other regions and can be prepared and used as described here. In fact, any way of loading building geometries into Unity that allows colliders can be used, as the implementation of the noise standard currently relies entirely on the physics system of the engine. All source code written for this publication is publicly available under https://github.com/Markus-Berger/ImmersiveNoise.

Abbreviations

The following abbreviations are used in this manuscript:

AR	Augmented Reality
CNOSSOS-EU	Common Noise Assessment Methods in Europe
DOF	Degrees of Freedom
HMD	Head-Mounted Display
HRTF	Head Related Transfer Function
LoD	Level of Detail
SPL	Sound Pressure Level
VR	Virtual Reality

References

1. Wing, L.C.; Kwan, L.C.; Kwong, T.M. *Visualization of Complex Noise Environment by Virtual Reality Technologies*; Environment Protection Department (EPD): Hong Kong, China, 2006.
2. Steele, C. A critical review of some traffic noise prediction models. *Appl. Acoust.* **2001**, *62*, 271–287. [CrossRef]
3. Kang, J. *Urban Sound Environment*; CRC Press: Boca Raton, FL, USA, 2006.
4. Ruotolo, F.; Maffei, L.; Di Gabriele, M.; Iachini, T.; Masullo, M.; Ruggiero, G.; Senese, V.P. Immersive virtual reality and environmental noise assessment: An innovative audio–visual approach. *Environ. Impact Assess. Rev.* **2013**, *41*, 10–20. [CrossRef]
5. Sanchez, G.M.E.; van Renterghem, T.; Sun, K.; de Coensel, B.; Botteldooren, D. Using Virtual Reality for assessing the role of noise in the audio-visual design of an urban public space. *Landsc. Urban Plan.* **2017**, *167*, 98–107. [CrossRef]
6. Manyoky, M.; Hayek, U.W.; Pieren, R.; Heutschi, K.; Grêt-Regamey, A. Evaluating a visual-acoustic simulation for wind park assessment. *Landsc. Urban Plan.* **2016**, *153*, 180–197. [CrossRef]

7. Yu, T.; Behm, H.; Bill, R.; Kang, J. Audio-visual perception of new wind parks. *Landsc. Urban Plan.* **2017**, *165*, 1–10. [CrossRef]
8. Mapbox—Maps for Unity. Available online: https://www.mapbox.com/unity/ (accessed on 10 May 2019).
9. Ejima, K.; Kashiyama, K.; Tanigawa, M.; Shimura, M. A road traffic noise evaluation system considering a stereoscopic sound field using virtual reality technology. In Proceedings of the 5th Asia Pacific Congress on Computational Mechanics and the 4th International Symposium on Computational Mechanics, Singapore, 11–14 December 2013.
10. Kephalopoulos, S.; Paviotti, M.; Ledee, F.A. *Common Noise Assessment Methods in Europe (CNOSSOS-EU)*; European Union: Brussels, Belgium, 2012.
11. Yu, T.; Behm, H.; Bill, R.; Kang, J. Validity of VR Technology on the Smartphone for the Study of Wind Park Soundscapes. *ISPRS Int. J. Geo-Inf.* **2018**, *7*, 152. [CrossRef]
12. Bozgeyikli, E.; Raij, A.; Katkoori, S.; Dubey, R. Point & teleport locomotion technique for virtual reality. In Proceedings of the 2016 Annual Symposium on Computer-Human Interaction in Play, Austin, TX, USA, 16–19 October 2016; pp. 205–216.
13. Schinke, T.; Henze, N.; Boll, S. Visualization of off-screen objects in mobile augmented reality. In Proceedings of the 12th International Conference on Human Computer Interaction with Mobile Devices and Services, Lisbon, Portugal, 7–10 September 2010; pp. 313–316.
14. Lorenz, H.; Trapp, M.; Döllner, J.; Jobst, M. Interactive multi-perspective views of virtual 3D landscape and city models. In *The European Information Society*; Springer: Berlin, Germany, 2008; pp. 301–321.
15. Käser, D.P.; Parker, E.; Glazier, A.; Podwal, M.; Seegmiller, M.; Wang, C.P.; Karlsson, P.; Ashkenazi, N.; Kim, J.; Le, A.; et al. The making of Google earth VR. In Proceedings of the ACM SIGGRAPH 2017 Talks, Los Angeles, CA, USA, 30 July–3 August 2017; p. 63.
16. Lu, L.; Becker, T.; Löwner, M.O. 3D complete traffic noise analysis based on CityGML. In *Advances in 3D Geoinformation*; Abdul-Rahman, A., Ed.; Lecture Notes in Geoinformation and Cartography; Springer International Publishing: Cham, Switzerland, 2017; pp. 265–283.
17. Park, J.W.; Yun, C.H.; Jung, H.S.; Lee, Y.W. Visualization of urban air pollution with cloud computing. In Proceedings of the 2011 IEEE World Congress on Services, Washington, DC, USA, 4–9 July 2011; pp. 578–583.
18. Stoter, J.; de Kluijver, H.; Kurakula, V. 3D noise mapping in urban areas. *Int. J. Geogr. Inf. Sci.* **2008**, *22*, 907–924. [CrossRef]
19. Luboschik, M.; Berger, P.; Staadt, O. On spatial perception issues in augmented reality based immersive analytics. In Proceedings of the 2016 ACM Companion on Interactive Surfaces and Spaces, Niagara Falls, ON, Canada, 6–9 November 2016; pp. 47–53.
20. Wagner Filho, J.A.; Rey, M.F.; Freitas, C.M.; Nedel, L. Immersive visualization of abstract information: An evaluation on dimensionally-reduced data scatterplots. In Proceedings of the 2018 IEEE Conference on Virtual Reality and 3D User Interfaces (VR), Reutlingen, Germany, 18–22 March 2018; pp. 483–490.
21. Hermann, T. Taxonomy and definitions for sonification and auditory display. In Proceedings of the 14th International Conference on Auditory Display, Paris, France, 24–27 June 2008.
22. Unity Technologies—Unity Engine Documentation for Audio Spatialization. Available online: https://docs.unity3d.com/ScriptReference/AudioSource-spatialize.html (accessed on 10 May 2019).
23. Bies, D.A.; Hansen, C.; Howard, C. *Engineering Noise Control*; CRC Press: Boca Raton, FL, USA, 2017.
24. Moss, J.D.; Muth, E.R. Characteristics of head-mounted displays and their effects on simulator sickness. *Hum. Factors* **2011**, *53*, 308–319. [CrossRef] [PubMed]
25. Davis, S.; Nesbitt, K.; Nalivaiko, E. A systematic review of cybersickness. In Proceedings of the 2014 Conference on Interactive Entertainment, Newcastle, NSW, Australia, 2–3 December 2014; pp. 1–9.
26. van Waveren, J.M. The asynchronous time warp for virtual reality on consumer hardware. In Proceedings of the 22nd ACM Conference on Virtual Reality Software and Technology, Munich, Germany, 2–4 November 2016; pp. 37–46.
27. Rungta, A.; Schissler, C.; Rewkowski, N.; Mehra, R.; Manocha, D. Diffraction Kernels for Interactive Sound Propagation in Dynamic Environments. *IEEE Trans. Vis. Comput. Graph.* **2018**, *24*, 1613–1622. [CrossRef] [PubMed]

28. Blut, C.; Blut, T.; Blankenbach, J. CityGML goes mobile: Application of large 3D CityGML models on smartphones. *Int. J. Digit. Earth* **2019**, *12*, 25–42. [CrossRef]
29. Weibetaker, T.; Kunert, A.; Frohlich, B.; Kulik, A. Spatial updating and simulator sickness during steering and jumping in immersive virtual environments. In Proceedings of the 2018 IEEE Conference on Virtual Reality and 3D User Interfaces (VR), Reutlingen, Germany, 18–22 March 2018; pp. 97–104.
30. Funkhouser, T.; Tsingos, N.; Jot, J.M. Survey of Methods for Modeling Sound Propagation in Interactive Virtual Environment Systems. 2003. Available online: https://www.cs.princeton.edu/~funk/presence03.pdf (accessed on 10 May 2019).