Ambient Sound with Signed Distance Fields and Gradient Fields

Tiago Boelter Mizdal, Cesar Tadeu Pozzer
Universidade Federal de Santa Maria
Programa de Pós-Graduação em Ciência da Computação
Santa Maria, Brazil
tiagomizdal@gmail.com, pozzer@inf.ufsm.br

***Abstract*— Ambient sound is an important part of any video game. They can present the player with the world that is beyond what is on the screen and can have a key role in creating an immersive experience. In this paper we propose methods to map static ambient sounds on worlds of any size in a very efficient way and we also present a realistic way to map the sound of static objects such as rivers. The first method uses signed distance field and the second uses gradient fields. This research also contains methods to save this field into files and options that can be used to reduce the size of the files.**

Keywords-ambient sound; signed distance field; gradient fields; compression; spatial sound;

I. INTRODUCTION

The Video Game industry is a big part of our world. They give us the opportunity to immerse ourselves on a different world and to experience things that we would not be able to experience in any other way.

Sounds are very important in creating an engaging experience. They can present a world to the player beyond what is presented on the screen. A certain type of music can create an atmosphere that helps to engage the player's emotions on a determined situation. For example on a scary game or on an intensive action packed shootout. Sound cues can give a feedback on what is happening to the player, they can tell when to change gear on a racing game, tell if the player hit a certain target, can make the player aware of enemies before they appear on screen, among many other things. Mapping ambient sound is an important part of sound design. It consists in playing a certain type of sound or sounds depending on the player's location and surroundings.

This paper presents a method to accurately map ambient sound on sceneries of any size and also allows different sound effects to be mapped on different parts of the scenery. The method uses signed distance fields to map every location on the environment, where different distance fields are used for different sounds. Gradient fields are used to map sounds from static sources that occupy a large portion of the environment, such as rivers or forests. We also present a way to accurately calculate the origin of the sound from static sources, so it can be implemented on a signed distance field. The research presented on this paper can be easily implemented on most video game engines, such as Unreal Engine and Unity.

To the best of our knowledge, no prior researches have considered the use of signed distance fields and gradient fields to map ambient sound. Most of the information regarding this topic was found on internet forums and talks on Game Developers Conference (GDC), but no information found resembles our research.

This paper is organized as follows: Section II presents related works on the subjects, Section III introduces a method to map the environment, Section IV describes a method to give direction to the sound, Section V discusses about saving files and how to reduce their sizes, Section VI shows how this paper can be used on modern video game engines, Section VII presents results and section VIII discusses conclusion and future works.

The contributions of this paper are as follows:

- We propose a method to map different static ambient sounds using signed distance fields and gradient fields. The precision of the audio source is given by the resolution of the fields.
- The signed distance fields and gradients field can be precomputed and saved on file. This makes finding the proper ambient audio to play on a specific situation extremely fast.
- The solution presented can also be used to map different sound effects on different parts of the environment.
- Multi-resolution gradient fields and signed distance fields can be applied to save space and accurately map critical regions, such as buildings or other enclosed spaces.

II. RELATED WORK

In [1] they use axis aligned bounding boxes and oriented bounding boxes [2] to separate the environment in different zones. Each zone has a corresponding ambient sound and a bounding box is used to check for collisions. When the player collides with a bounding box, that zone is triggered and a different ambient sound is played. They also apply different sound effects to different parts of the scenery, like reverberation or occlusion.

The use of raytracing on [3] allows for a more precise mapping of the player immediate surroundings, it also present a method to accurately produce sound effects, based on objects that are close to the player and that can interfere with the audio in some sort. The accuracy and performance of this method is given by the amount of rays casted per frame.

Mapping the sound of a river according to the player's location can be a challenging task. The player may be moving alongside the river and the sound must come from a direction that allows the player to locate the river, even when the player cannot see it. Most of the methods known revolve around the use of multiple instances of the same

audio source placed along the course of the river and then being blended together to simulate the correct direction. Our method can accurately map the direction of a river without the need of multiple audio sources.

III. MAPPING THE ENVIROMENT

A signed distance field (SDF) is a way of mapping the virtual world according to a distance relative to a specific area or object. On this paper positive distances means that we are outside the area or object, while negative distances means that we are inside.

Using a signed distance field, allows us to map different audio for different regions of our world, it also allows us to give a different volume to the sound, based on the player's position in the world. A SDF does not allow us to map the direction from where the sound is coming, but in many situations, there is no need for the sound to have a specific source. For example, when the player enters a house, there could be some ambient audio there, like a music to help put the player on a specific mood, but it does not need to come from a specific spot. Another example is wind, we can hear the wind on a determined region, but do not hear the direction it is coming from.

To map the environment and give a direction from where the sound is actually coming from, we use gradient fields. Which is similar to a SDF, but instead of only saving the distances, we save a vector that points to the sound source.

Since the SDF can only map a distance, we have to use a different SDF for every audio that we want to map. We begin by separating our environment into equally sized cells. The resolution of the cells depends on the accuracy required by the target application. The smaller the cell size the more accurate the sound is, but on the other hand, the overall size of the SDF will be larger. Fig. 1 shows an example of an environment with its cells and possible SDF.

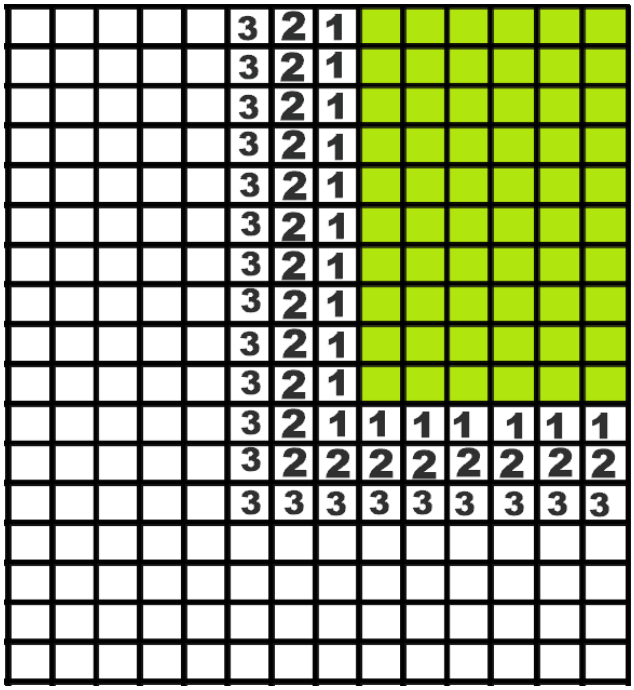

Figure 1. In green the region to be mapped. The numbers represent the distance from that cell to the region.

For each cell, we calculate the distance from the cell center position to the closest point in the area that we are evaluating. Then, based on the distance we can set the volume of the audio for that region.

To avoid playing audio when we are too far away from the area, we can set a threshold for the distance, and if the distance is larger than our threshold, no audio will be played.

IV. GIVING A DIRECTION TO THE SOUND

A SDF maps the environment by storing a distance, thus it can only map the places where a sound must be played and the intensity in which it is supposed to be played. To map the sound origin position and give it a direction we can use a gradient field.

A gradient field is similar to an SDF, but instead of saving the distances between every position on the world to a determined area, we save a vector. This vector will have a direction that points to where the sound is coming from and the volume of the sound is relative to the vector's magnitude.

To calculate this vector we developed an equation that takes in consideration the fact that the volume in which humans perceive a sound is squared relative to the intensity in which the sound is played. This vector points in the overall direction given by multiples audio sources near the player.

Given a maximum distance that we can hear the sound, we calculate a vector for those positions that are closer than the maximum distance to a river or other static object that we want to map.

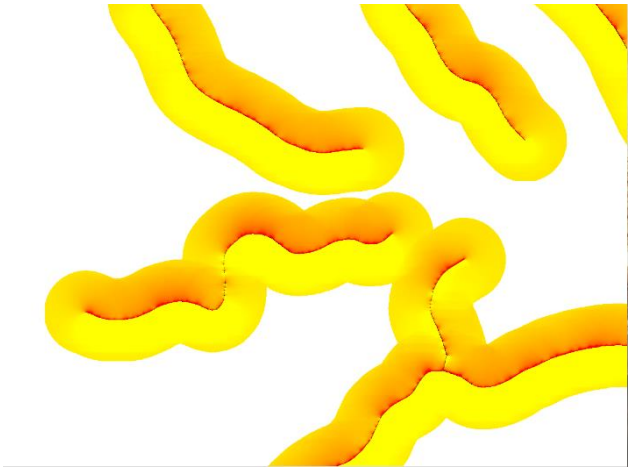

Figure 2. A resulting gradient field by mapping a set of rivers.

Given a situation where we want to map the sound of a river. By getting close to the river, the sound of the river will be coming from multiple points. We developed an equation that allows us to take in consideration every point that act as a sound source and calculate the overall direction from where the sound should be coming from.

$$V = \frac{\sum_1^n \frac{(o_n - p)}{|o_n - p|^2}}{n}$$

First, we calculate a set of vectors that are given from a defined position to all points closer than the maximum

distance from our target area or object. Then we calculate the average of the sum of all the vectors in the set divided by their respective magnitudes squared. As can be seen on equation 1. Where V is the resulting vector, p is the player position, n is the number of samples from our object and o_n is the position on our object.

Fig. 2 shows an example of a gradient field that maps the sound of multiple rivers, on this example the maximum distance to play a sound is 75 meters. Fig. 3 shows resulting vectors from the equation.

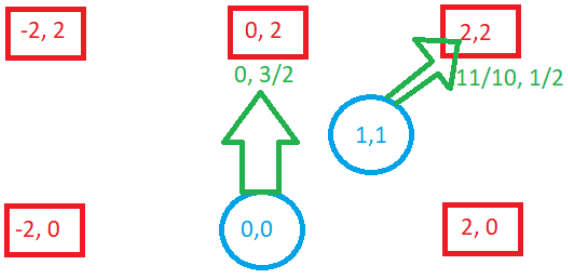

Figure 3. The number represent the Cartesian coordinates. In red the static objects to be mapped. In blue the player positions. In green the resulting vectors.

V. MAPPING DIFFERENT SOUND EFFECTS

By using something similar to a SDF and a gradient field, we can map sound effects to different areas. For example, we can map the environment with values that correspond to the amount of reverberation under a bridge or inside a tunnel.

Another way to do this is by pre-defining different sound effects, such as distortion, reverberation, echo and many others. For each sound effect we can specify a number and create a table, thus we can map different sound effects at once. Considering that a byte contains eight bits, we can set each bit to a different sound effect. Making it possible to map eight different sound effects at the same time while using a single byte of memory for each cell. However this way we cannot save the accordingly intensity of the sound effect, only if it is occurring or not.

VI. MULTI-RESOLUTION FIELDS

There are certain scenarios where a fixed resolution for the SDF and the gradient field can cause some errors. For example, given a world that contains a house where a specific audio must be played. The house may not be a perfect fit for our SDF, and so one or more edges may rest in the middle of a cell. Therefore, when the players get near the house, we may evaluate a cell and consider it to be inside the house and play the sound. But in reality the player is outside the house, but on a cell that contains it.

To solve this problem we propose the use of multiples SDF or gradient fields, each with a different resolution. One resolution for the overall world and a higher resolution for the house. We create the SDF or gradient field for the house using a resolution the closely fits the model, thus avoiding any errors that may occur. Then we define a specific value for the field, the value tells us that on that cell we must find the other SDF or gradient field. Thus sampling the higher resolution.

Fig. 4 shows an example of the error that can occur, and Fig. 5 present multiple resolution that can solve the problem.

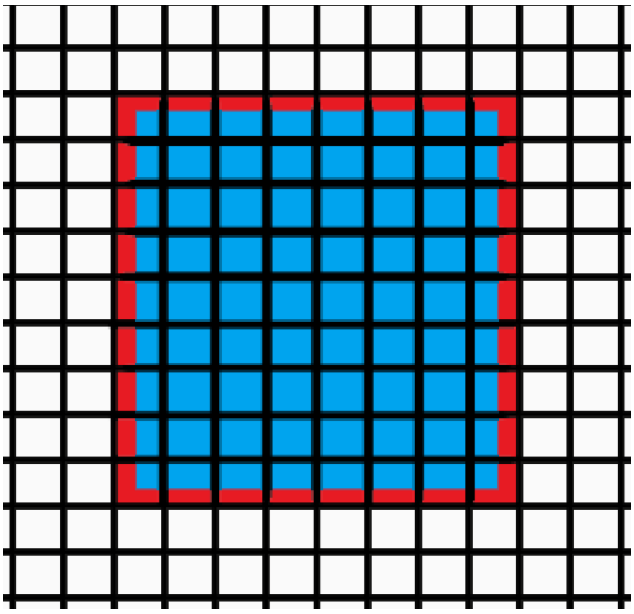

Figure 4. The blue square is the house or object we want to map and the red cells are considered to be inside the house.

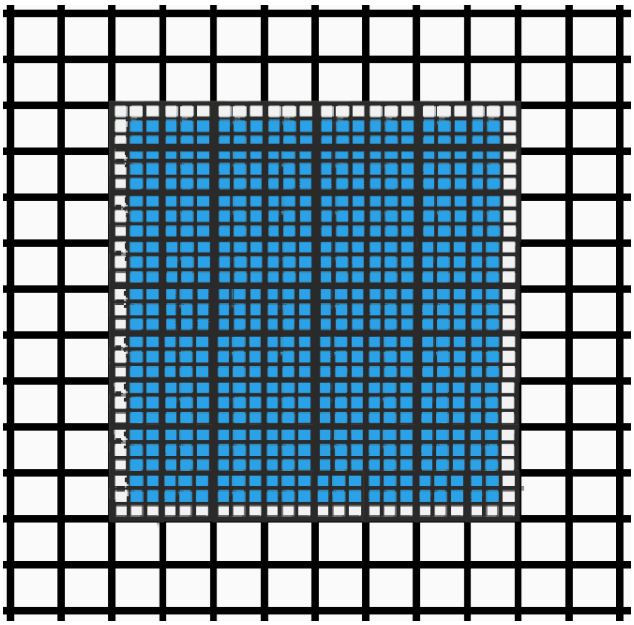

Figure 5. Multiple resolutions, the blue square is the house or object to be mapped, and the multiple resolution solves the problem from Fig. 4.

VII. SAVING TO A FILE

Both the SDF and the gradient field can be precomputed and saved into files. Since the SDF is consisted only by distances, we can save it as an integer or a float. However, the gradient field consists of vectors and should be saved as two or three floats per vector, depending on a 2D or 3D approach. For the following we will consider a 2D approach.

Considering that, a float is two bytes and an integer is one. The size of the files can be quite large in a raw state. For example, given a world of 500 by 500 meters and a resolution for the SDF and gradient field of 10 centimeters. The fields would have 25 million entries, which would be from 100 to 200 megabytes of storage for the SDF and 400 megabytes for the gradient field. For larger worlds with better resolutions, the size of the files can reach several gigabytes.

To shrink the file sizes we explored two options. The first one is based on [4], where they precompute a determined number of vectors and use those vectors as an approximation of the real ones. The second option is to use compression methods, in this paper we focused on LZ4 and deflate because they are both lossless and can greatly compress files that have a contain an single entry multiple times.

Quake 2 precomputes 162 normal vectors to be used later as an approximation to calculate the lighting of the scene. In our research, we decided that saving the entries from the gradient field as an unsigned short, which provides an excellent precision and the resulting file size is one eighth of the original one, which contain two floats per entry.

An unsigned short has two bytes, which can represent up to 65535 distinct numbers. This gives us enough information to represent up to 360 different vectors with 181 different magnitudes. We precompute all this different vectors and associate each one of them as a number from zero to 65160 on a table. For each vector on the gradient field, we find the most similar vector that was precomputed, and substitute them by the number on the table. By doing this we lose some precision, but the size of the file is greatly diminished.

To compress the files, we must first set a no data value. This no data value is a determined number, which will represent the entries, that no sound is played. For example, if the maximum distance is 100, the no data value can be 101.

Since in the majority of cases, most of our SDFs and gradient fields will represent sound only in a certain portion of the entire scenery. By setting a no data value, we ensure that a large portion of the SDFs and the gradient fields will have the exact same value, which will greatly increase the compression rate of LZ4 [5] and deflate [6].

In the most common cases LZ4 was able to compress the files an average of 900% and deflate was able to compress an average of 800%. Both compression algorithms have very similar decompression times.

On the worst cases, where we did not set a no data value and the entire SDFs and gradient fields were created without a maximum distance, we were still able to reduce the size of the files by 50% to 60%.

We can also combine both options to get even better results.

VIII. USAGE ON MODERN VIDEO GAME ENGINES

Most modern video game engines have an audio listener, which is usually on the camera, and an audio source. The audio listener is used to receive the audio from the audio sources. The audio sources are position on the world that will emit sound. The engines calculate the way that the sound is perceived based on all the audio sources available and the position of the listener.

To implement our method on most video game engines using SDF, we can set the audio source as non-spatial or 2D or we can place the audio source at the same position as the listener. Therefore, when we play the audio, the sound does not come from a specific location, it does not have a direction. Moreover, we can adjust the volume of that specific sound based on the values of the SDF.

In the case of gradient fields, we have to set the audio sources as spatial or 3D and place the audio source in a certain direction from the listener, this direction is based on the vector from the gradient field. Fig. 6 presents an example from Fig. 3. The magnitude of the vector represent the intensity of the sound played.

Figure 6. The wood boxes are the static objects and the green box represents the audio source. The player contains the listener.

IX. RESULTS

The methods presented on this paper were implemented on Unity Engine and Unreal Engine 4.22.3. In both engines we were able to precisely map ambient sounds in different environments.

Utilizing signed distance field we were able to easily map regions with wind and forests. In addition, different types of audio can be played at the same time where different regions overlap.

The use of gradient field presented the best results. The ability to precisely map static ambient sounds on the world can make a huge difference on the player's immersion. By mapping a river course through a scenery, we were able to follow the river course without noticing any kind of flaws and always having the correct direction from the sound source.

Both the SDFs and gradient fields were computed using compute shaders, in order to accelerate the process. The fields were saved into textures and then into images. Computing SDFs and gradient field, with resolutions of 8192 by 8192 took from two to five seconds to compute, depending on the amount of object that needed to be analyzed. When we mapped rivers and forests on a terrain with 60 by 90 kilometers and resolution of half a meter, the whole process took 20 minutes for the SDF and close to one hour for the gradient field.

When reading the SDF and the gradient field from files, we used a method that allows the file to be read in smaller blocks. Reading a single block based on the players location had zero impact in performance. A block contains

a small portion of the map, so another block will only be read when the player moved a certain distance.

On the most common cases, the size of the files ranged from 200 megabytes to 1 gigabyte. However, the after applying our options to shrink the files, we were able to diminish the files to one to five megabytes. On our most extreme test, where the file had 16 gigabytes, after applying both the table with precomputed vectors and compressing it with deflate the resulting file had only 80 megabytes.

Implementing the methods proposed on this paper is easy on most modern video game engines, since the majority of engines present audio listeners and audio sources.

X. Conclusion and future works

The research is new and presents a precise way to map ambient sounds on worlds of any size. The use of gradient fields present a better way to add static ambient sound to our worlds. The method proposed to map the sound of a river has a better result than any other method that we could find.

By precomputing the SDFs and gradient field and saving them into files, we have a precise way of mapping the scenery and still has zero impact on performance. This paper also presents options to reduce the final size of the files.

On procedural generated worlds, every time a new part of the world would be generated we would also need to generate all corresponding SDFs and gradient fields for that part. Depending on the case, calculating all fields can take a toll of performance, however, once the sounds are mapped the performance would increase. A solution to avoid performance loss is to calculate the SDFs and gradient field in an asynchronous way, and only using them once they are done.

Another option to improve the size of the files is by using multiples resolutions on the same SDF or gradient field. This would allow for smaller files and lower computing times.

Wind can change the distances from which we perceive a sound. Another point of future research is to apply wind when reading from a SDF or gradient field, to hear sounds that would not be possible before on determined position.

Acknowledgment

We thank the Brazilian army for the support through the SIS-ASTROS project.

References

[1] Rockstar Games, “The Sound of Grand Theft Auto V,” Game Developers Conference 2014. Accessed on July, 15, 2019. https://www.youtube.com/watch?v=L4GuM15QOFE.

[2] C. Ericson, “Real Time Collisiong Detection”, Morgan Kauffman Publishers, 2005.

[3] Platinum Games, “An Interactive Sound Dystopia: Real-Time Audio Processing in NieR:Automata.,” Game Developers Conference 2018. Accessed on July, 15, 2019. https://www.youtube.com/watch?v=BrUQdd96qzk.

[4] Id Software, “Quake 2 Source Code,” Accessed on July, 15, 2019. https://github.com/id-Software/Quake-2.

[5] LZ4, “LZ4,” Accessed on July, 15, 2019. https://github.com/lz4

[6] L. Peter Deutsch, “DEFLATE Compressed Data Format Specification version 1.3,” 1996.