CREATING AUDIO TEXTURES BY EXAMPLE: TILING AND STITCHING

J. R. Parker and B. Behm
Digital Media Laboratory
University of Calgary

Abstract

When creating multimedia constructions such as web pages, games, and animations, it is common to play some sorts of sound as a 'loop', where the sound file is played repeatedly from beginning to end. This is an easy way to extend a short audio sample, but the repetition is easily detected by a human listener. It is possible to create new sounds from examples, so that looping is not needed.

1. INTRODUCTION

Most computer users have played a modern computer game, and have probably been impressed with the quality of the graphics and the audio. A question that arises on more detailed analysis is 'where do the sounds come from?' There may be a background ocean or wind sound, the sound of engines, rain or even a crowd cheering. The background or ambient sounds continue for as long as needed - if the player is called away to answer the telephone or the door, the sounds could play for many minutes or even hours.

How does this work? In most cases, there is a simple loop, in which the same brief sound is played over and over until called upon to stop. This fact is easily detected by the human ear, and usually becomes irritating after a while. In the case of a game, there are between 10 and 100 hours of game play included on a typical game CD. There cannot, of course, be this much novel audio, even if it were compressed. Without reusing sound sequences, the games could not present a consistent audio presence, so it is clear that the sounds have to be reused somehow.

It should be possible to create realistic sounds using computer techniques, and thus create as much of any given sound as is needed. Sound synthesis is frequently not sufficiently realistic (yet); computer created wind or surf sounds often seem artificial. One way to solve the problem is to use small samples of a desired sound and to reconstitute them to for a new, longer, and non-repeating sample. This is the subject that we wish to explore.

What we are attempting to describe here is a method for creating longer sounding *sound textures* from short samples. A sound texture can be described as having a somewhat random character, but a recognizable quality. Any small sample of a sound texture should sound very much like, but not identical to, any other small sample. The dominant frequency should not change, nor should any rhythm or timbre. The sound of rain falling, surf, fire, wind, a large crowd - these are all sound textures; music and speech are not. However, our experience with synthesizing sound textures indicates that some non-texture sounds can be synthesized using the techniques to be discussed.

In this paper we will first examine the background work. Next, we will look at two ways to create sound textures from examples; these algorithms will be implemented and tested on five small audio texture samples, creating a longer sample of new sound in each case.

2. IMAGES VS SOUND

It may seem that creating sound textures from samples would be easier than creating image textures, but that is not true in practice. Just as a visual texture in image form can be thought of as a two-dimensional signal, we can treat a sound texture as a one dimensional signal. These two quite different signals share many properties, especially where the nature of a texture is concerned. Thus, it should be possible to extend some of the existing methods for generating visual textures to the generation of audio textures. In fact, there are significant complicating factors. One of these, for instance, is *scale* - a pixel frequently represents a larger fraction of an image than a single audio sample does of a typical sound image, both geometrically and logically. Thus, methods that generate texture pixels one at a time need to be modified to use audio data blocks of a significant size.

Sound frequency is usually thought to be analogous to the color of a pixel. Yet changing the color of a pixel is trivial, while the frequency of a single sound sample cannot be altered - frequency is a temporal property associated with a collection of samples. Again and again, a careful examination of the issues shows that audio data is just as complicated as image data, and in some cases more so. There is no reason to believe that audio texture generation will be faster or easier to implement than the methods currently used for images, or that those methods will adapt precisely to the audio domain.

3. RELATED WORK WITH IMAGES

There is a respectable collection of background research on texture generation in computer graphics, especially in the past five or six years. This is partly due to a burst of activity in the area of image based rendering. Graphical textures are fairly well studied and there has been some success with synthesizing them. A brief summary of some of the texture relevant synthesis methods is therefore in order.

3.1. Tiling and Stitching Based methods

One the easiest methods for building large graphical textures from smaller ones is to cut pieces from the samples and stitch them together in a new order. Of course, pieces can be reused, in whole or in part, allowing much bigger images to be built from smaller samples. There are various ways to deal with the fact that the seams between the samples are often visible.

This method is called *image quilting* by Efros [3]. The method he describes starts with square sample blocks of a fixed size, with an overlap between adjacent blocks. Instead of selecting random blocks to be adjacent, select blocks that have some significant measure of agreement between them in the overlap region. Smoothing the edges reduces the visibility of the joins between the blocks but does not eliminate it.

A second strategy in the category of tiling and stitching methods is the so-called *chaos mosaic* [7]. In this method, a simple tiling of the output image is actually created as a first step. For example, if

the existing sample is 10x10 pixels and we wish to fill a 100x100 pixel output image, then 100 copies of the sample are placed side by side into the output image. The output image is then iteratively subjected to a chaos transformation so as to destroy the tiled appearance.

There are a variety of possible transformations, but the one used by Xu maps the output image onto itself as follows:

$$x^{l+1} = (x^l + y^l) \bmod m$$
$$y^{l+1} = (x^l + 2y^l) \bmod m$$

where the image size is **m x m** and the iteration number is **l**. This is *Arnold's Cat Map* [1].

To correct for the fact that the chaos transformation does not preserve local features, the transformation is applied to a block of pixels within the image, and moves entire blocks. Still, the boundaries between the blocks can sometimes be seen, and the solution proposed is to erase a small set of pixels along the boundary and to force them to fit specifically into the region, thus 'smoothing' the edges. The need for this could be considered a flaw in the technique, as a second synthesis algorithm has to be implemented just for this purpose. Fortunately, it is possible to perform a 'fade' between the blocks across a small overlap region and achieve an acceptable visual effect.

4. RELATED WORK WITH SOUND

Nicolas Saint-Arnaud and Kris Popat [4] at the MIT Media Laboratory represent one of the very few efforts to synthesize audio textures. They describe a two-level representation of sound, where the low level consists of so-called *atomic elements* seeded through time, and the high level is a description of the distribution of those elements.

Their implementation used a binary tree structured filter to extract atoms, and at the high level they estimated the probability of each atom based on the previous ones to build a statistical model. They used this scheme to generate some simple sounds: two sine waves, photocopier noise, and applause, but not without difficulties: clicks and pops appear when not expected, and some periodicity and flavor was lost.

A relatively old technique called *granular synthesis* [5] was tried by bar Joseph [2] in combination with a statistical learning scheme. The belief that "*All sound is an integration of grains, of elementary sonic particles, of sonic quanta.*" [6] has yielded a paradigm of sound generation as the creation of many thousands of brief sound grains that can be linearly combined to form larger scale sounds, like music. Each grain is an envelope having an attack phase, a sustain phase, and a release phase, all in a time period of between 10-50 milliseconds. The method of bar Joseph attempts to learn the grain structure by decomposing the input sample audio signal into wavelets, then generating a multiple resolution tree. This tree can then be used to create new collections of sound grains that have an excellent similarity to the original sample sound.

It should be mentioned that in all discussions of texture generation that have been encountered, no effective scheme for objective evaluation of the methods has been discussed.

5. SIMILARITY IN AUDIO

How do we tell if one generated texture is better than another? We need a way to compare audio textures against each other in an objective manner. Subjective measures fail to a great extent due to the temporal nature of sound; two images can be placed side by side for comparison, whereas two sounds must be listened to one after the other. How do we tell for sure when one texture is a little better than another? This problem must be addressed before an evaluation of audio texture generation methods can be accomplished.

Looking at the problem simply, it is obvious that two different audio textures will never be identical, even if they represent the same sound - the random component of each would be unlikely to agree. The basic properties of a sound sample that are easy to measure are amplitude, represented by the value at each sample point, and the frequency, represented by the values in the Fourier transform of the sample. The overall duration of the sound is probably not relevant, but there may be a phase difference that interferes with a simple matching procedure - the sounds may include a temporal variability that is not in synchrony, causing a difference to be observed. These are the essential factors that must be considered when attempting to compare two sound textures.

The problem of similarity in audio files is too complex to be summarized here, but the difficulty of the problem should be understood. For example, we have tried three comparison methods devised specifically for the purpose of comparing sound textures, and none could perform better than about 60% success in controlled circumstances, as compared against human perceptions. Success in a comparison algorithm would provide higher similarity measures between samples that are known to be from the same source than from those from different sources. Software located on the Internet fared much worse than our 60% success rate. It was decided to rate the texture generation methods using two schemes: to have listeners subjectively rank sounds, and to make the sound files available on the Internet for anyone to access and evaluate for themselves. Until much more work has been done on similarity in audio data, this is probably the best that can be done.

6. AUDIO TEXTURE GENERATION

As a first step towards the generation of a new audio texture, rearranging and concatenating small samples of the source sounds was tried. The simplest way to do this is to repeatedly copy chunks of data from random points within the source into a new audio texture.

Using this method, audible and jarring defects are heard during the transitions between random chunks. Clearly, a method of smoothing these transitions is necessary. We use a cross-fade (*Alpha blending*) algorithm that blends the tail of the preceding chunk into the head of the new chunk. The length of the blended portion is variable, but we found that 15% of the chunk size gives subjectively good results. The cross-fade between samples helps minimize the distortion in the transition between random audio chunks, but does not eliminate it. This is an audio equivalent of the image quilting idea described above (Figure 1).

Since an audio texture is, by definition, slightly repetitive, it is not necessary to rely on randomly choosing the next chunk in the texture. There should always be some number of chunks that could logically follow the previous chunk. Taking advantage of this property of textures eliminates distortion along transitions and enforces the similarity between the source and final texture. We use a least-squares similarity measure to find the regions in the

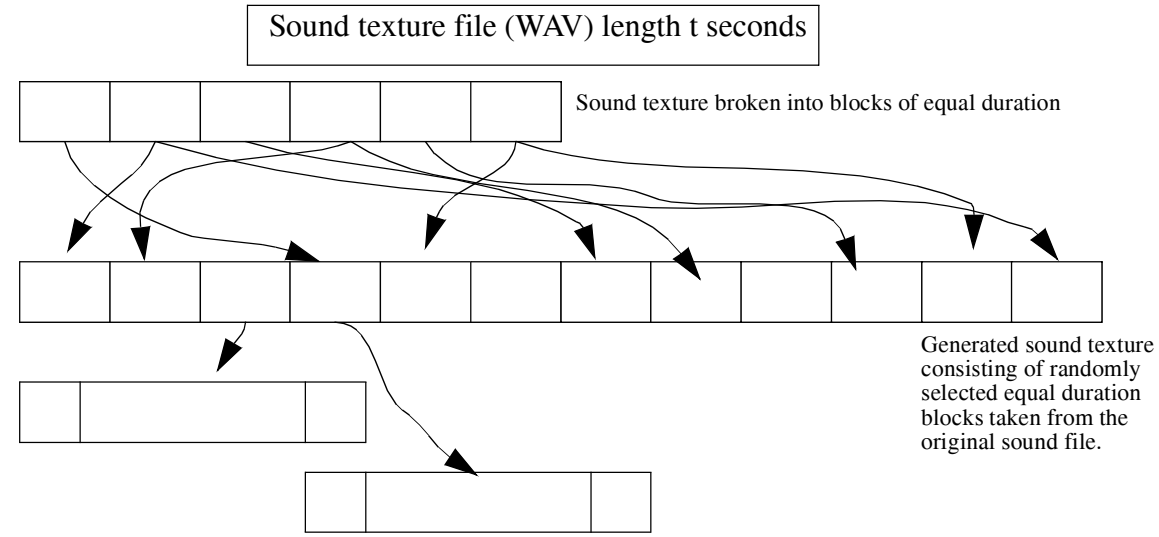

Figure 1 - Synthesis of a sound texture using equal size sample blocks.

source sample that are similar to the tail (that is, the last 15%) of the preceding chunk.

Since the chunks originate from the source sample, there is going to be one exact match, exactly where the tail occurs in the source. Allowing the copy to continue from this point results in the original sample simply being copied repetitively into the final texture, effectively tiling the source. This is undesired in a texture, so we forbid this behavior.

This method of choosing chunks provides nearly seamless transitions, but also creates some obvious tiling, or repetition, which we consider to be a defect in a texture. It is common for this algorithm to get stuck in a loop, in which it returns to a specific spot after a fixed number of chunks.

A method is needed to encourage the chunk-selection algorithm to make use of the entire source sample, and not get stuck using the same regions over and over. This is done using an accumulator vector of the same size as the source sample. At initialization, this accumulator is set to zero. When samples are copied out of the source, the corresponding bins in the accumulator are incremented by 5. After each iteration of the copy phase, all non-zero elements in the accumulator are decremented by one. When searching for the best match using the least-squares similarity measure, as above, only spaces whose accumulator 'rank' is zero are searched. If there are not enough consecutive zero-ranked elements to form the head of the new chunk, then the accumulator is decremented and the search is run again. This is very similar to the *least recently used* (LRU) algorithm used in virtual memory page replacement.

This technique forces the search algorithm to use all of the available source material. Sometimes, this results in less-than ideal matches, but we have found the transitions to be unnoticeable. It is important to note that when searching for a suitable chunk, only the rank of the lead-in segment (first 15%) is used. Because of this, it is still possible for 'popular' segments to be used several times, but their increasing rank will eventually cause the algorithm to look elsewhere. It is this behavior that prevents segments from becoming available in the same order that they were used, and thus reduces tiling.

The size of the chunks used in all of the above methods determines to a great extent the quality of the texture. The optimal chunk size depends on the specific source sample used. Less 'busy' samples, with a lower incidence of audible features, need longer chunks to avoid choppiness in the final result. Busier samples require smaller chunks to avoid audible tiling. By hand tailoring this size to a specific source sample, very good results can be obtained, but at the expense of generality.

One way to automatically determine the size of the chunks is using amplitude peaks. The entire source sample is analyzed for RMS amplitude, and peaks in amplitude more than 1.5 standard deviations from the baseline are recorded. The mean and standard deviation of the observed distance between these peaks is used to generate the size of each chunk. Hopefully, then, each chunk will contain one 'feature' that a listener can recognize. This method works reasonably well on most textures, but is still somewhat experimental. In the future, we plan to experiment with several frequency-based search strategies to determine the optimal chunk size. It is possible, however, that there is no method that will correctly determine optimal chunk size for all inputs.

It is less obvious how to use the chaos mosaic method for audio textures, since it appears to be fundamentally two dimensional in character. Still, it is possible to create a two dimensional matrix from a one dimensional sound simply by filling the matrix row by row with sampled audio data. The chaos transformation can then be applied, perhaps many times, and then the sound can be read out in row-major order.

Because audio data is one-dimensional in character, the applicability of a 2D tiling was not immediately obvious. However, if the size of a row in the 2D sound sample matrix is exactly large enough to hold one period at the dominant frequency, or an integer number of periods, then there will be a logical phase connection between the rows of the matrix. In our experiments the matrix has rows with a length that is a multiple of the dominant frequency.

In order to create the texture, rectangular regions are copied from this matrix, T, to a destination matrix, M. First, T is divided into regions. The width of these regions will affect the quality of the sound texture in the same way as the chunk size did in our method above. Fortunately, we have some additional information available to us in this case. The dominant frequency will logically be low for less busy sounds, and higher for busier sounds. We can therefore express the region width as a multiple of this frequency. Empirical evidence shows that a multiplier of 150 gives acceptable results for our textures. After the source matrix T is partitioned into such regions, they undergo a random distortion. Each corner of the region is moved by a random amount, generated using a normal function with d = 15% of the box size. After this distortion, the boxes are copied from T to M using Arnold's Cat Map [1] to determine the position of the region within the destination matrix.

Due to the nature of Arnold's Cat Map, it is unlikely that regions from the source matrix T will entirely cover the destination matrix M. In [7] this problem is solved by applying the Cat Map at the pixel level before regions are copied, creating an "even and visually stochastic" background from the source pixels to fill in the uncovered regions. Applying the Cat Map to individual digital audio samples does not result in a convincing background noise. We changed the scale to use small blocks of samples, as before, and applying the cat map to these blocks. The background sounds became more convincing as the block size grew. An acceptable background is produced with block sizes on the order of 1/2 of a second.

Despite the chaotic nature of this system, obvious defects are audible in any texture with recognizable features. Textures with few identifiable features - rainfall, for example - can be produced using this method, but the poor memory efficiency of this method make it an unattractive choice even for these simple textures.

7. EVALUATION OF GENERATED TEXTURES

Five different sound textures were used to test the synthesis methods we have discussed. These data are:

a.wav - A crowd b.wav - Flames
c.wav - Light rain falling d.wav - Ocean surf
e.wav - Water lapping on a shore

Synthesized textures of 20 seconds duration were created from each of the original samples, using each of the methods discussed: block resampling and chaos mosaic. Only a few of the possible combinations of parameters could be tested, but some basic conclusions can be drawn.

Overall, block resampling appears to yield the best results, followed by the chaos mosaic. The ranking was done using the subjective opinion of 30 students and faculty over a period of two months.

Some further criticisms of the methods we obtained by having staff experiment with them. The chaos mosaic technique is really far too slow for practical purposes. Some of the generated textures show amplitude variations not seen in the original. These can usually be smoothed by synthesizing with larger block sizes and/or overlap regions.

All of the original and synthetic sound textures are available on the web at:

www. cpsc.ucalgary.ca/~parker/AUDIO

The relatively short duration of the samples was intended to allow rapid download times. Most textures have just one synthetic result; it has been observed that the audible flaws in the synthetic output can be eliminated by adjusting the control parameters on the algorithms (E.G. block size, overlap area, etc.). Note that there is a total of eight non-texture sounds that have been evaluated, and are included on the web page for public scrutiny.

8. CONCLUSIONS

We have devised methods for synthesizing audio textures which work also for selected other sounds, and have found that the process can be done in real time, and that the textures created are of sufficiently high quality to be used in some computer games and animations. There are audible flaws that require further research and study in order to correct, but most can be eliminated by (manually) modifying the control parameters of the synthesis software. We will continue this work by attempting to set these parameters automatically based on a detailed analysis of the sound sample provided as the basis for synthesis.

9. REFERENCES

1. V.I. Arnold and A. Avez, Ergodic *Problems of Classical Mechanics*, Benjamin, 1968.
2. Z. Bar-Joseph, D. Lischinski, and M. Werman, Granular Synthesis of Sound Textures Using Statistical Learning, *Proceedings ICMC*, 1999.
3. A. Efros and W.T. Freeman, Image Quilting for Texture Synthesis and Transfer, *Proc. SIGGRAPH 2001*, Los Angeles, Aug. 12-17, 2001.
4. N. N. Saint-Arnaud and K. Popat, Analysis and Synthesis of Sound Textures, *Proc. IJCAI-95 Workshop on Computational Auditory Scene Analysis*, Montreal, August 1995, Pp. 125-131.
5. Truax, B. 1988. Real-Time Granular Synthesis with a Digital Signal Processor. *C. M. J.* Vol. 12, No. 2, pp 14-25.
6. I. Xenakis, Formalized Music, Indiana University Press, Bloomington, 1971.
7. Y. Xu, B. Guo, and H. Shum, Chaos Mosaic: Fast and Memory Efficient Texture Synthesis, *Technical report MSR-TR-2000-32*, Microsoft Research, April, 2000.