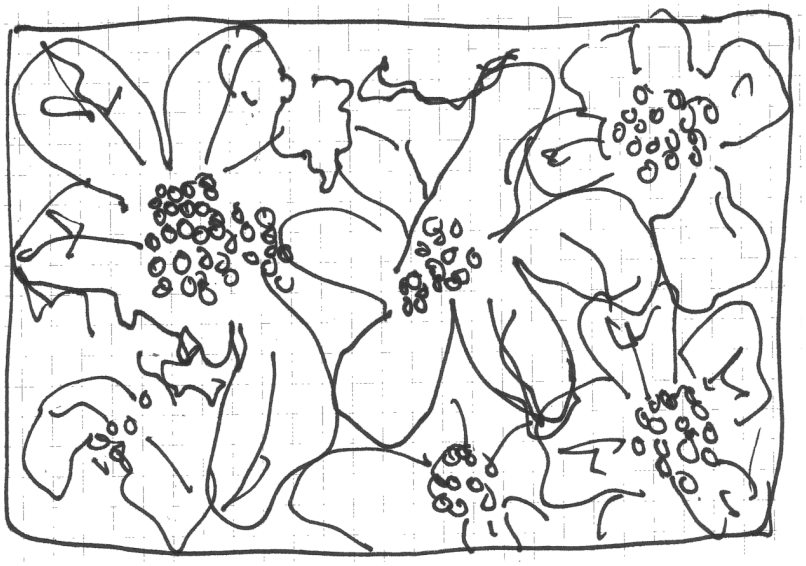
Audiovisual Granular Synthesis for Composition and Performance

Michael R. Bernstein

Friday, April 16, 2004

Instructors: J. Sharp, L. Wolozin,
G. Levin & M. Doyle

Table Of Contents



0. Abstract

This thesis is an investigation of the atomic elements of digital sound and graphics, and how they can be related and expressed through performance. My approach draws on the fields of texture synthesis, stochastic composition, and a history of texture as a composition and performance subject. Audiovisual Granular Synthesis for Composition and Performance is comprised of two software components, one that allows for the creation and composition of audiovisual grains, small bits of sound and image that border on the imperceptible, and one that includes an environment for performing these grains in real time. Once audio and visual grains are composed separately, they are linked together. The final manifestation of these paired grains are viewable as a full screen animation for the visual component and a stereo sound component. The graphical explorations in this thesis have led to an augmentation of the texture synthesis algorithm to include controls for temporal coherence during synthesized textural animations.

1. Introduction

"The original presentation included two lasers, 92 spotlights, and bonfires and processions of torches on the neighboring hillsides. The music was diffused throughout the site over 59 loudspeakers. In the middle of the desert, in the middle of the summer, it would have been, and by all accounts was, an awesome experience." - A description of the performance of Iannis Xenakis' Persepolis, 1971. (James 2001)

1.1. Motivation

The reduction of perceptual material such as audio and visual stimuli to digital signals raises many questions about the nature of the signals themselves, and how they may be related to each other. It is possible and almost standard to execute the same mathematical functions on digital sound and digital image files, because their signals are reducible to the same types of data, often in the same ranges. When the results of these functions resemble each other, what does that mean? The investigations in this thesis are driven by the desire to make these connections clear and to understand the fundamental similarities between signals that may seem insignificant on the level of analysis, but take on new and important meanings when these connections are made perceivable.

Figure 1. A still from Stan Brakhage's "Moth Light" 1963. Brakhage glued moth wings and other natural textures to film and animated them to tell the story of a Moth's life.

In researching the connections between these signal types, I have become particularly interested in the concept of visual and audio textures and how they relate to the signals that compose them. The abstract film work of Stan Brakhage, pioneering American avant-garde filmmaker, has been a particular influence in this realm, where texture is composed and arrange in order to tell stories as described in the above quote. Additionally the compositions of Iannis Xenakis, Greek composer and theoretician, extensively used the concept of texture in his music. I am explicitly motivated by the idea that audiovisual textures, as composed of audiovisual grains, can have the same profound expressive qualities as the work of these two masters.

1.2. Overview of this Thesis

In an attempt to make connections between audio and visual signals, I have chosen to focus my attention on the small elements that, when combined, create rich

textures capable of representing dynamic content. This has lead me to consider small patches of image texture and small "blips" or "grains" of sound that border on the imperceivable yet represent significant information when combined using various computational methods. In this way I can “knit” an audiovisually rich “fabric” out of the “threads” of these small grains. This focus, on particulate matter of digital images and sounds, has led me to investigate whether or not connections on this "micro" or atomic scale can produce more meaningful connections than those made on a "higher level." Connecting these atomic elements using metrics that require signal analysis in addition to intuitive user input, I have composed a set of audiovisual grains and a software interface to perform with these tightly connected combinatory signals.

I have established the research contained in this thesis on the intersection of representative works in the fields of audio granular synthesis, graphical texture synthesis, and artistic work dealing with the composition and performance of textures, both in the fields of motion graphics and experimental music. Within these fields I have devised six categories that I will explicate and evaluate for effectiveness in the respective field in that it is contained. These six categories include The Science of Audio / Visual Grains; The Composition of Audio / Visual Grains; and The Performance of Audio /Visual Grains/Textures. From these six categories I intend to demonstrate that Audiovisual Granular Synthesis answers the questions that I have set out to answer and that the examples I provide cover the issues raised by the investigation contained herein.

Once the background for this thesis has been established, I will give an overview of the solution from a conceptual and technical stand point, and explain the decisions I have made for which areas of research to pursue and how to best embody this research with a set of software tools. This section, that encompasses methodology and implementation strategies, will revamp and describe the questions this thesis raises and how these methodologies and implementations speak to these questions.

1.3. Contributions of this Thesis

This thesis introduces an approach, Audiovisual Granular Synthesis, to answer long-asked questions about the connections between audio and visual stimuli. Focusing on digital signal representations of these perceptual elements, I have shown that pairings made on the micro scale, where individual components border on the imperceptible, are fruitful in new and interesting ways with respect to the overall question about the connections between divergent signal types.

In addition to answering questions about the connections between audio and visual stimuli on a particulate level, this thesis deals extensively with the ideas of audio and visual textures and how they apply to the construction of more complex and dynamic perceptual signals. Grains of sound and graphical patches are inherently linked with sound textures and graphical textures respectively, and I have shown that utilizing a new medium of audiovisual grains can lead to novel explorations in the field of audiovisual texture.

The realm of audiovisual texture offers many possibilities, and the focus of the performance component of this thesis is real-time and improvised as opposed to pre-composed and rendered. The final form of the performance software offers a software interface that uses the metaphor of an ecosystem for its primary control mechanisms. The user is capable of choosing between several textures on two audiovisual channels, and to mix back and forth between the channels. The selection of textures is based on a deformed grid map, which limits the connections between textures, enforcing constraints on the movement from one texture to the other during the overall performance.

2. Background

2.1. Background Overview

The background component of this thesis is interested in the need to investigate the scientific and perceptual connections between audio and visual media as a case for the importance of Audiovisual Granular Synthesis, and it will assert that audio and visual textures represent fruitful territory for these investigations.

Keeping in mind the overall question of whether or not audio and visual signals can be paired based on the atomic, micro scale elements on which they are built, I have divided this background section into smaller subsections designed to deal with the involved questions. The first section, "The Science of Textures," will explain the scientific and computational background for audio and visual textures, while pointing out historical and contemporary examples of scientists, engineers, and artists who have dealt with these questions directly during the course of their work. The second section, "Textural Composition," will deal with artists who have utilized, in visual and musical fields, the ideas of texture and granularity in the compositional elements of their work. Finally, "Textural Performance" will highlight contemporary examples of artists who have used texture for purposes

beyond composition and have brought these issues into the realm of real-time performance.

These three categories represent the thrust of this thesis and should be seen as a large part of the argument being established, namely that the micro scale is the most appropriate scale on which to "pair" audio and visual components, and that pairings made on this level will produce intuitive and meaningful connections.

2.2. The Science of Textures

In order to explain the techniques I am using to create animate sonic and visual textures, it is essential to explain the concept of textures and how they are generated from a scientific standpoint. The examples in this section demonstrate the origins of digital implementations of sonic and visual texture generation through advanced forms of synthesis.

Figure 2. Four Common textures for computer graphics. From the upper-left image going clockwise, these textures are scanned, scanned, synthesized poorly, and synthesized well.

2.2.1. Graphical Textures and Texture Synthesis

Visual textures are defined as images that have a local structure but no global structure. (Efros 2001) The types of textures commonly being referred to in this thesis are of the type usually taken from photographs or other digital representations of images. However this has more to do with the conventions of texture synthesis than the directions of this thesis. I have synthesized textures with synthetic examples in addition to sampled textures, and have tested the boundaries of my performance software by inscribing dynamic image ability into the language of the grains themselves. This point is where the synthesis algorithm I have chosen to use differs from others: its ability to handle varying types, but not all types of textures, allows for performance improvements which make it available for real-time usage.

Considering the normal fare for textures, such as stones, or bricks, or other stochastic and irregular patterns, such as those shown in Figure 2, it is difficult to conceive of the individual elements that make up the image. When granular synthesis was chosen to handle the audio component of the audiovisual pairing, texture synthesis presented itself as an obvious choice, due to its reliance on small patches of input textures in order to synthesize larger novel textures. Therefore the micro scale *equivalent* of audio grains could be said to be these small patches which are organized around the local content-based structures of the sample texture.

I was introduced to texture synthesis through the work of A.A. Efros, a computer

scientist at UC Berkeley who has done some recent revolutionary work in the field. Through Efros I have encountered other implementations of texture synthesis including Liang 2001, Zhu et al 1998 and the early work of Ken Perlin. These examples represent a wide array of implementations of texture synthesis, including perceptually based filter models, mathematically based function generators, and those relying on probability density functions (PDF). (Liang 2001)

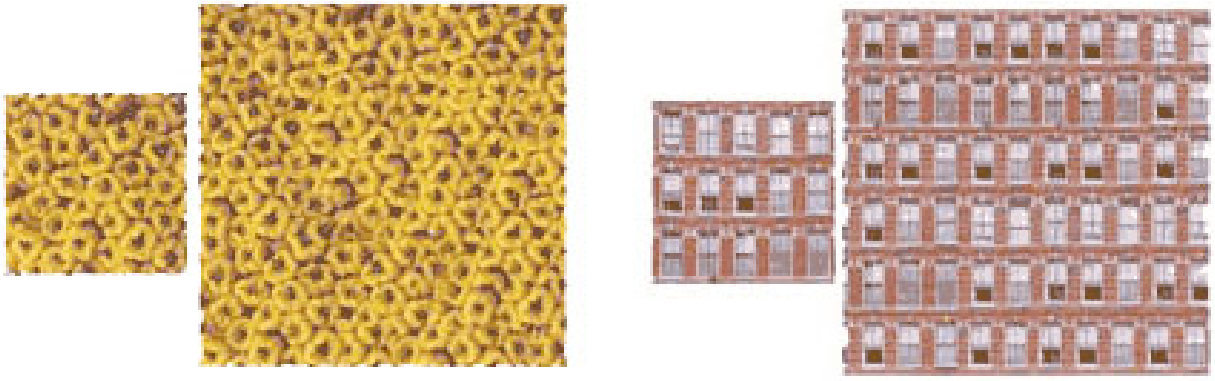


Figure 3. Two examples of Efros' texture synthesis. The smaller texture on the left is the sample image, and the larger image to the right is the synthesized texture. Notice the new patterns forming in the building image and the problems with the pepper image.

Texture synthesis is defined as a process by which larger novel textures can be generated from smaller input images. In real-world computing applications, implementations of texture synthesis are extremely helpful in any situation where real time texturing is necessary and repetition in texture is not desirable. Take for example a 3D simulation where a texture to be placed on a virtual wall is taken from a scanned photograph. (Liang 2001) If the texture contains any stochastic elements, that is if it is not a repetitive pattern but contains elements that bind it together locally, it is not acceptable to simply tile the image. Tiling the image produces undesirable effects such as repetition in the overall image even if the image is not repetitive, and obvious visible seems throughout the texture, as shown in Figure 4. (Liang 2001) As an alternative to tiling input textures, Texture Synthesis offers the ability to synthesize, or create "from scratch," graphical elements that maintain the local structure of a texture and thus preserve the perceptual elements without detrimentally harming content.

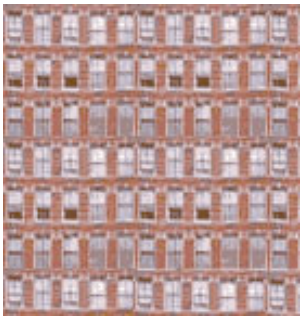


Figure 4. The results of texture tiling. Note the repetition of the

Liang 2001 utilizes the concept of probability density functions by relying on

Markov Random Fields (MRF), a mathematical concept which characterizes the values in a set in terms of the connections between values. This speeds Efros 2001 up by several orders of magnitude. (Liang 2001) MRF are useful because they are known to model a large number of textures and they preserve local structures. The concept of Patch Based Sampling is key to real-time texture synthesis, and its similarities with the signals of audio grains make this technique highly desirable.

The texture synthesis algorithms presented above, including Liang and Efros', are all flawed in one major and similar way: their commitment to synthesizing all types and classifications of textures preclude them from being successful in real time performance. Efros synthesis takes seconds, while the other methods do not even conceive that such an operation could operate in the real time domain.

In October 2003, Steve Zelinka and Michael Garland published a paper entitled "Jump Map-Based Interactive Texture Synthesis" which boasted real time performance for static texture synthesis. Textures 256 x 256 and smaller were reported to take six hundreths of a second on an Athlon 1800+ PC. (Zelinka 2003) This algorithm utilizes the idea of splitting the texture synthesis operation into two distinct parts: an analysis stage and a synthesis stage. (Zelinka 2003) This way, the operations which are the most time consuming, of comparing pixels or groups of pixels to each other, is only done once, and the light operations, of assigning colors to pixels, are the only operations which need to be executed repeatedly.

2.2.1.1. Real Time Texture Synthesis

Understanding that texture synthesis was possible in real time prompted me to investigate the possibility of animated, temporally enhanced texture synthesis, which could be capable of a dynamic range of styles, nuances, behaviors, and aesthetic styles and approaches.

I envisioned a living Brakhage frame, where dirt could swarm around inside of itself, the dead skin of the Moth could crawl, and the lines of the film strip would live out their endless journey out of the top of the frame and back into the bottom.

2.2.2. Audio Textures and Granular Synthesis

Audio textures exist in opposition to traditional methods of conveying content and emotion through music. While traditional composition techniques emphasize larger time scales, textural techniques focus on the micro scale. (Roads 2001) For example, a Bach Concerto refers to "notes" that a performer should play, varying in

exact time but always capable of being heard, while the Micro time scale is on the order of 1/44,100 second. Working on the micro time scale, or the time scale in which the components verge on the imperceptible, connotes certain compositional techniques that are necessarily different than those for larger, more inclusive time-scales. The results of operations on the micro time scale tend to be organic and fluid sounding, as they tend to model "natural" sonic phenomena such as birds chirping, gravel crunching beneath our feat, water flowing, etc. (Roads 2001) The performance component of this thesis involves many manipulations of operations which occur on the micro time scale.

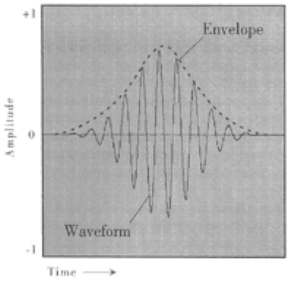


Figure 5. A visual depiction of an audio grain, shaped by a Hanning Function. The contents of this grain are sinusoidal in nature.

Since the creation of the devices necessary to synthesize sound, there have been technological advancements that have changed the face of electronic music production. Namely the invention and popularization of all-purpose microcontrollers like Pentium and Apple chips from the 1980's until the present have created possibilities in the field that simply did not exist before. (Roads 2001) Because this thesis is attempting to answer questions about the atomic elements of digital sound, I have chosen to deal almost exclusively with a particular type of synthesis known as Granular Synthesis.

Granular Synthesis is the creation of sound utilizing short pieces or "grains" of sound that border on the imperceptible. Typically lasting anywhere from 1 to 100 milliseconds, these grains are combined in huge numbers to create "animated sonic atmospheres." (Roads 2001) The grains themselves consist of frequency and time domain information that, when combined and played back in large numbers, are capable of sonifying natural and particle based phenomena. A component of Granular Synthesis is its ability to create sonic atmospheres or textures by first creating the grains and then altering their composition in real time. This allows precise control over the sound texture being created without having to deal with many of the undesirable pitfalls of improvising in real time with acoustic instruments that are incapable of being manipulated on the Microsound time scale. (Roads 2001) Microsound is a term developed by Curtis Roads, the first person to implement Granular Synthesis in a digital environment, to refer to these sounds that border on the imperceptible. (Roads '96) Granular Synthesis allows for the novel creation of sounds and sound textures by controlling properties of the grains themselves in addition to the manner in which these grains are "played" or perceived.

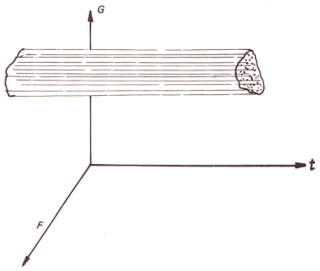


Figure 6. Xenakis' conception of grains in space, along the axes of frequency, intensity and time. Here we see the beginning of the

Audio Granular Synthesis is a technique that was prompted by the intellectual

curiosity of a long line of composers, computer scientists, and musicians. Most notably, Iannis Xenakis composed *Analogique A&B* for tape and strings that used principles of granular synthesis that he discussed as composition on the microsound scale. (Roads 2001) After taking a course with Iannis Xenakis in Music and Mathematics at Indiana University in 1972, Roads went on to implement and develop digital granular synthesis for the next thirty years. (Roads '01) This course was a distillation of Xenakis' book *Formalized Music* from one year earlier. (Xenakis '71) Roads, in his two tomes on digital sound *The Computer Music Tutorial* (Roads '96) and *Microsound* (Roads '01), outlines the history and implementation of every common and most uncommon techniques for sound synthesis.

Roads lays out the conceptualization of granular synthesis in terms of "clouds," that are controllable and parameterized to an amazing degree. During the creation of grain clouds, Roads outlines how you can control the density of the grains at any point during the cloud, and also control the duration, amplitude, and content of the grains in real time as well.

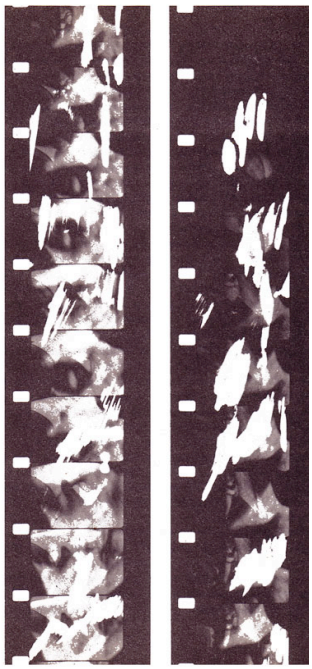


Figure 7. Stills from Brakhage's *Dog Star Man: Part 3*. Sitney points out in his book *Visionary Films* the similarities between this work and Jackson Pollock's Abstract Expressionist Paintings. This thesis is interested in this idea insofar as this work can be extended to the idea of texture composition.

2.3. Textural Composition

2.3.1. Composition of Visual Textures

Composing Visual Grains and Textures has deep roots in experimental film, but is perhaps most excellently represented in the work of American Filmmaker Stan Brakhage. Over four decades Brakhage uncompromisingly explored many issues with film that examined the nature of filmmaking and the limitations of the medium. By applying paint and other objects directly to film, Brakhage was able to explore composition for texture in ways that were not possible using a camera, filming "reality" and abstracting it to a level of texture. Instead Brakhage chooses to synthesize texture as he composes, using these non-camera and non-developed film methods as ways to circumvent literalness of representation.

Brakhage's methods were explicit in their desire to explore texture in many different examples. Moth Light (1963) is a short film where moth wings, grass, leaves, and other natural objects are glued to the film in order to show the natural textures up close. In "Eye Myth" (1972), hand-painted textures are applied to a still image (a photograph) until the image is obscured, and then it is partially re-revealed. "The Garden of Earthly Delights" (1981) and "The Dante Quartet" (1987) explore both methods of attaching objects and hand painting textures to

film. In the notes to the Criterion DVD Collection "by Brakhage," Fred Camper, long time Brakhage Historian, notes that Brakhage would not put music to most of his films because the music would "dominate the image's rhythms" that were fundamental to the appreciation and understanding of Brakhage's films. (Camper '03)

I have explored the relationship between these visual textures and sound, and using Brakhage's content as an inspiration, I have ecplored the synthesis of these textures on a smaller scale (with a finer granularity) with digital, non-natural means. In the creation of my performance software I had the desireto treat the *substance* of Brakhage's films as a fluid surface that I could manipulate and change with simple physical gestures. With each performance, I hope to compliment the types of visuals that Brakhage conceptualized without ruining the feelings evoked by silence that make his films so remarkable.

The types of visuals this thesis demonstrates includes various visual textures being animated using texture synthesis from jump maps. The types of textures used in my software include I did on my skin, along with high resolution scans of flowers and plant matter, and digital file format images of rocks, rusty metal, animal hair, and more.

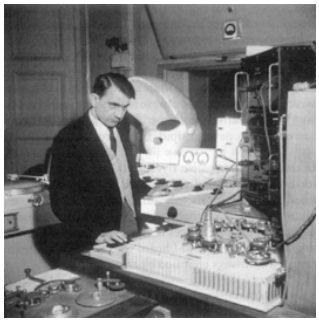


Figure 8. Pierre Schaeffer at the helm of one of his many machines for manipulating magnetic tape at his state-sponsored studio in France. Schaeffer is often credited with coining the term Musique Concrete, and the work he did in the early 20th century exemplified textural audio explorations.

2.3.2. Composition of Audio Textures

Historically, achieving audio textures required the use of fluctuations in amplitude and pitch in normal instrumentation settings. Many composers utilized concepts of texture in their orchestral works, such as Xenakis, Stockhausen, and Harry Partch. The interesting textures, however, developed with the advent of electroacoustic music, or music that explored alternate electronic methods of sound production alongside traditional acoustic methods.

The idea of sampling in the analog sound realm was used by Pierre Schaeffer and Pierre Henry, amongst others, in a movement in composition in the early 20th century called "Musique Concrete," translated into english as "Concrete Music" or "Actual Music." The idea behind Musique Concrete was that utilizing instruments alone to make sounds is to leave out a large part of what actually makes up sound, and that using "concrete" or "actual" sounds to make music is somehow more honest and worthwhile.

Schaeffer and Henry composed works for magnetic tape, where pre-recorded

sounds from natural or industrial sources were manipulated to make unrecognizable and dynamic sounds with new timbres and unique time scales. Famously, Henry composed a work for magnetic tape entitled "Variations for a Door and a Sigh," where the sound of a door slamming and a person sighing is manipulated and repeated into myriad textures and abstract soundscapes. (Henry 1965) The concept of sampling has been extended and utilized to the present day, taking many forms from Hip-Hop to Noise Music.

Synthesis, on the other hand, utilizes the concept of the creation of the elements of a sound "from scratch," where the individual elements are perceptually recognizable but not necessarily "meaningful." Hardware sound synthesis toward the ends of creating music was popularized and executed during the 1960's, with synthesizer companies such as Moog Music and Buchla creating synthesizers for the professional and home markets. (Trocco 2002) These synthesizers used analog circuitry to build larger sounds with a modular framework that had the potential to be connected in many different ways. By linking sound generators with filters, and other modulators, the analog systems were capable of producing strange sounds never heard before that time. In this way, the creation of music became intrinsically tied in with the technology being developed. The history of instruments that utilize electricity or electronic sound creation methods are precisely detailed in a chart in Curtis Roads' *Microsound* entitled "Electric and electronic music instruments: 1899-1950," with over eighty examples provided.

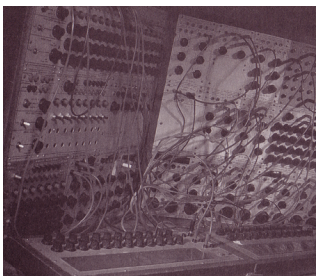


Figure 9. Don Buchla's Buchla 100 Music Box, one of the first Modular Synthesizers for the composition of electronic music. The music created with these synthesizers has inherently textural qualities due to the focus on timbre .

Iannis Xenakis was responsible for the progeny of Composing Audio Grains, when he conceptualized, composed, and executed *Analogique A and B* in 1958-59. (Roads '01) Xenakis here composed sounds that are beautiful and natural sounding, very dynamic, and fresh sounding. The outcome, less than five minutes of synthesized sound, was remarkable for the time and shows how dedicated Xenakis was to seeing his concept through to fruition. I will clearly not be emulating Xenakis' methods, of using magnetic tape and generators to achieve granular synthesis. Instead I will use his example as an inspiration for the conceptual underpinnings of composition with grains. As excerpted in *Microsound* (Roads '01), *Formalized Music* (Xenakis '71) defines the concept of dealing with sound on a micro scale, how this could be implemented, and what kind of impact this would have on the outcome of the composition and the very practice of composing. I have tried to extend this to the audiovisual, where I have considered clouds as consisting of Audiovisual grains instead of solely Audio grains as Roads and Xenakis espoused. I have often considered Xenakis' music (in Persepolis,

Analogique, and other Electronic works, for example) to be very visual in terms of its use of space and tonal coloring, and his early work as an architect has often been cited as an influence to his later work. Perhaps due to the conceptual aspects of the visual, the synthesis with the sonic will not be hard to muster.

2.4. Textural Performance

2.4.1. Visual Textural Performance

Performing visual texture is well supported by the conveniences of modern computational systems. Programming suites like Max/Jitter, a commercial product for graphically programming matrix based data streams (optimized for use with video frames) offer direct access to real-time video information with little computational or programmatic overhead.

Figure 10. Sue Costible performing with visual textures. Using Jitter, a projected surface, and pre-composed textures, Costible creates live and idiosyncratic textures that repeat themselves and reveal their technical underpinnings. This work is significant in its reliance on texture, and resembles Brakhage's work, to some extent, in its oblique desire to represent narrative.

Sue Costible, a San Francisco based artist, uses Jitter in her own work to create real time collages out of pre-composed textures that she makes with transparencies and vellum. A time delay system and manipulation of the video frames makes for an entrancing visual experience that points directly to the concepts I am trying to

outline in the genre of texture performance. While there are moments where the subject matter is representational, it is not these moments, but rather the transitional points between abstraction and representation that are emphasized. Costible is able to move between textures with ease due to the ease of the interface.
This has been an inspirational component for me of Costible's work, that it is an excellent example of how to take advantage of the power of the computer without having to deal with its admittedly arcane interface methods (mouse, keyboard, etc.). Costible is often accompanied by music when she plays, and in this respect my work will differ from hers: she performs along with music, using presumably perceptual metrics (rhythm, pitch, etc.) to influence how she performs. My system is composed of a set of united audio and visual textures, with a selection method that can hopefully approach the natural and intuitive method that Cositble has developed.

2.4.2. Audio Textural Performance

Conceptualizing music performance in terms of texture is an idea that predates granular synthesis, in the work of Xenakis and beyond. Instead of focusing on "contemporary classical" composers who wrote pieces to be performed that are textural in nature, I would prefer to focus on contemporary electronic musicians who use principles of texture and improvisation to explore sound. Mego Records, an electronic music label based in Vienna, Austria, is an example of a collection of artists who are principally involved in the idea of musical texture. Amongst these artists are Pita (Peter Rehberg, one of the founders of the label) and Christian Fennesz (also known as simply "Fennesz"). These two musicians utilize in their music an intense, moment-to-moment focus on the idea of texture and use it in extreme forms to enormously dynamic ends. While Pita's music tends to be harsh and in some respects more textural, Fennesz is interested in the intersection between the analog and the digital, as is evidenced on his album "Endless Summer," that finds him playing guitar through a computer, processing the sound, then running the digital sound through analog circuitry to an analog recording device. The slowly shifting textures of "Endless Summer" show a dedication to texture in well crafted passages that move from melodies to noise and back again. Pita's textures are more raw and less processed, and tend to be evocative of the extreme energy present in the sound particles he is manipulating. While to my knowledge neither of these performers uses granular synthesis directly, their sounds are indebted to the concept of Microsound and the work of Xenakis and Roads certainly had a subconscious if not direct effect on their sounds.

2.5. Evaluation of How Background Material Contributed to This Thesis

Each of the examples I have chosen to represent the background material for this thesis have components I borrowed from, pitfalls I have avoided, and concepts I have utilized. As a systematic method of explaining these various evaluations, this subsection will follow the section structures above but will emphasize what I have learned from approaches taken by precedent work.

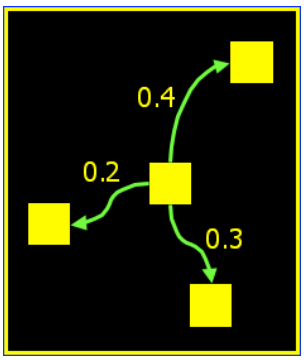


Figure 11. Mr. Zelinka's diagram explaining the jump map. The centermost yellow square represents a neighborhood of the input texture which is being compared against the other yellow squares. The numbers represent how similar, according, to Zelinka's algorithm, the neighborhoods are to the centermost area.

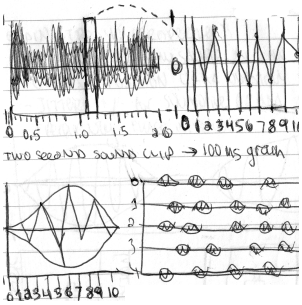


Figure 12. An early sketch from one of my notebooks about how to granulate sound files in real time. The sound file is converted into a simplified 8-bit version, and a grain (shown lower-left) is shown operating over several streams.

2.5.1. Texture Synthesis

I have learned scores from researching texture synthesis, in terms of how textures are created, and what the definition of a texture is. Most of the work I have looked into is excessively scientific and not artistically based, and it is nearly all completed by rendering as opposed to being capable of performing in real-time. The scientific component of the research papers I have read is harmful in the respect that there are certain "benchmark" textures that these algorithms need to process in order to be considered "appropriate" or "functional." In the case of the work in this thesis, I have composed visual textures to synthesize larger textures from, and in a sense I have designed my system according to the textures I wished to synthesize.

It is because I have tailored my system to accommodate the natural textures I am interested in synthesizing that I have chosen to use Zelinka and Garland's jump map algorithm, which has been successful in terms of its ability to synthesize textures fast enough to perform real time texture animations. The brilliant stroke in their method, of dividing the process of analysis and synthesis, has broken through so many barriers in terms of high quality and fast synthesis that in many respects this thesis could not have been executed without it.

2.5.2. Granular Synthesis

The work of Curtis Roads has been a great help in my conceptualization and realization of programmatic granular synthesis. *Microsound* as an overview is extremely thorough and outlines several possibilities that I have used in various combinations to execute the audio component of my performance software. Roads does not discuss the visual components to the creation of grains in *Microsound*, but does go into how they were linked in the field of experimental animation. The visual metaphors he uses to describe grains, such as clouds and other natural phenomena, are extremely helpful in conceptualizing the technical components of the method. These metaphors have also come into play with my interface design process, which utilizes similar concepts in the audiovisual realm. Roads'

metaphoric representations have made it easier for me to understand granular synthesis, because, as a designer, visual explanations are particularly resonant. Finally, Roads' discussions of pushing granular synthesis into the real time domain have been inspiring: he sees no limit to the computational possibilities of modern computers, and erased any doubt I had that the technical accompaniment of visuals to sound is possible.

2.5.3. Composing Visual Textures

Stan Brakhage's visual textures are extremely inspirational to me, and pointing out where this work "lacks," after reading his essays, interviews, and letters, seems a bit contradictory. His refusal to add sound to many of his films is inspirational to me rather than inhibitory; he points out that in order for this coupling to be worthwhile, it must be tight and natural. Above all, it is Brakhage's penchant for narrative through abstraction that I would like to borrow from, and his unflinching desire to communicate using alternative media. Brakhage made films while he starved, and created in conditions with little to no technology, working in a medium considered amateur, and yet produced some of the most breathtaking examples of abstract cinema. This thesis is inspired by the notions Brakhage has put forth about the purity of visual expression, and therefore the links I have made with the audio components are that much more well thought out.

2.5.4. Composing Audio Textures

Many musicians consider their work "textural," and the examples I have provided in previous sections merely represent, to a certain extent, current tastes and aesthetic triumphs. Musical composition is something that I have not studied formally but have learned a great deal about by reading the essays and ideas of many great modern composers, from Henry Cowell to John Cage, Iannis Xenakis, Stockhausen, Harry Partch, and more. I am interested in the real time components of much of the work of these composers, and how the textural qualities of their work can be extended to improvisational space in the textural domain. The visual possibilities posited by these great composers are endless, and often the scores, essays, and other ephemera surrounding their work offers many clues to their personal visual inspirations. I have continued to investigate these visual components, in the hope of extending my own work into the domain of sublimity that many of these composers are capable of achieving.

2.5.5. Performing Visual Textures

Visual texture performance is a much-maligned practice in current computer

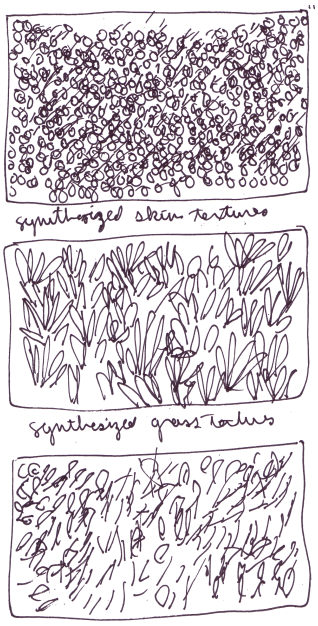


Figure 13. Sketches from my notebook depicting possible textures to synthesize. From top to bottom: human skin, grass, and dirt.

graphics trends, often showing up in over-saturated, palette-rotated, feedbackey video signals that lack content and are severely damaged in scope by their blatant penchant for psychedelia. While I am not opposed to "mind expanding" graphics, I am not interested in pure style, and my convictions for texture should not be confused with a love for complete abstraction. Sue Costible offers an interesting insight into this arena, shying away from bright colors and tending toward more muted tones, using gestural components to form visual textures, and allowing the tools behind her work to speak for themselves in many ways. Costible's work often accompanies music, but is not by any means audiovisual. In this manner she falls more in line with the "VJ" or Video-Jockey set, who compose live visuals to music. The audiovisual substance I will be creating and manipulating will be outside of this context, where the two will be inseparable. It will be interesting to see, in relation to Costible's work, how well my own is capable of faring.

3. Methodology

3.1. Process / Method Overview

3.1.1. Solution Details: Audiovisual Granular Synthesis

Audiovisual Granular Synthesis is a combination of techniques from the fields of Audio Granular Synthesis and Graphical Texture Synthesis. Breaking the components down into their respective media is helpful for explaining the methods used in their combination. The methodology presented here should be seen as a part of a larger group of questions, namely: "Can the manipulation of the atomic elements of digital graphics and sound aid in their combination in new and meaningful ways?" and "Can these atomic audiovisual pairings be used to compose and perform audiovisual textures?"

Audiovisual granular synthesis as an approach is different than its implementation. In addition to conceiving of audiovisual grains as units of this type of synthesis, I have composed a set of these grains. This set of grains is accessible within the context of the performance software I have written, which allows me to select and manipulate these textures in real time.

3.1.2. Synthesis and This Thesis

This thesis is inherently interested in the connections between the audio and the visual, and the combination of these media in novel ways and in both scientific and aesthetic terms. This combination has prompted me to research and understand

the manner in which these media are constructed, and understanding these phenomena is impossible without understand the atomic or particulate matter which compose them. It is for this reason that the principles of this thesis rely on the principles of synthesis.

Synthesis in sound is a term utilized in opposition to the idea of "sampling" or using pre-existing analog or digital media to compose "new" works. A form of synthesis that straddles the boundary between sampling and synthesis is Granular Synthesis, that I have relied on for the audio component of this thesis.

The idea of synthesis in terms of graphics is perhaps less intuitive and obvious than its parallels in the realm of sound. Graphical synthesis has many applications and covers many areas that are outside of the scope of this thesis, but one particular area, known as Texture Synthesis, is particularly relevant to the areas of research that this thesis deals with.

3.1.3. The Visual: Google, Brakhage, and Beyond

In attempting to conceptualize aesthetic directions that this thesis would explore, I was tempted to move away from vector-based graphics or graphics that were entirely abstract, formless and based on drawing commands. Instead I wanted to investigate graphical textures that while maintaining some of the abstracted components of much vector or drawing based work, could, if necessary, call upon the representational and take these types of forms. This is based on discussions I had with Golan Levin during the summer of 2003, in that he asked me to enumerate for him my sonic and visual inspirations. I answered that I was inspired visually by Google Image Searches, the service provided by Google that allows you to enter in a term and get a list of images as a result instead of a standard web page. My interest in this area has continued and recently has manifested itself as in interest in the work of Stan Brakhage, Harry Smith, and other experimental filmmakers who used various techniques to explore the composition of visual moving textures.

The graphical methodology of this thesis is in many ways more experimental and involved than the audio component, that has a long history of academic and artistic work behind it. Texture Synthesis is an inherently more scientific and exploratory field, with many of its foundational components still being fleshed out and changed according to the hardware capabilities of commercial and personal computers. To be able to tell a story with an animated visual texture is a goal I am

striving for with this component of my thesis. My predilection towards natural textures is displayed in a list of possible textures I made for a posting on my online thesis journal:

"facial hair,skin, body hair, leaves, brushed metal, dirty glass, blood splatters, cat hair, many different kinds of woods, plastics, anything dusty, spotty, naturally scattered, pinkies after day of writing with pen, scar tissue on right hand, flowers, wax, bright fabrics, thin line drawings, pencil smudges, mystical symbols, brakhage frames, leather, wires, solder splatters, animal trails, bug shells, sand, salt, spices, grains, gradients, modulo textures, noise patterns, clouds, freckles on face, cat litter, science fiction book covers, aged browning newspaper, chipped paint, oil, marble, trees, moss, ivy, dirt, closeups of erased chalkboard, acne, rashes, gums, tongue, inside of nose, mucus, vomit, saliva, shit, sea water, seaweed, raw meat, raw fish raw chicken, burned meat, as many animal skins as possible, toenail clippings, spray paint, dripping marker, nipples, lentils, bottoms of cd-rs, rugs, weaves, wool, spider webs…"

3.1.4. The Audio: Moth Wings and Gravel

The audio component of the texture performance is meant to have an organic and evolutionary sound that will be evocative of the visual. As Stan Brakhage has stated, having a film playing along with music is no more important than having a soundtrack playing while a painting is being displayed in a gallery. (Brakhage 2000) I would like to circumvent this by having the sound and image be connected from "birth." In effect, one is not playing along with the other, they are playing together, and could not exist without the other.

3.1.5. Jump Maps and Animated Texture Synthesis

Animated Texture Synthesis, as the visual component of this thesis, and beyond, is a novel concept that presents interesting questions to the problem set involved with texture synthesis. If you are generating textures for every frame of animation (ideally 30 per second), how do you maintain a sense of temporal coherence from frame to frame? The nature of Zelinka and Garland's jump map algorithm demands randomness: it is built into the heart of the system. Because partially stochastic, natural textures are favored, it cannot be a straight, predictable situation. The randomness is necessary.

Adding an extra set of functions to handle temporal coherence has been one of the most interesting challenges of this thesis, which relies heavily on the possibility that the system can be as visually dynamic as the sound has already presented itself to be. The ability to move from slow to fast, from noisy to patterned, from loud to quiet, etc., are all necessary to a system if it is going to be performable and capable of virtuosic composition and performance. Audiovisual Granular Synthesis has the potential to be a medium that grows with our ability to provide it with the right kinds of materials to synthesize.

3.2. Implementation

Informed by Zelinka, Xenakis, Roads, Brakhage, and more, I have created a performance system to perform audiovisual textures. This solution has an attempt to answer the questions I have posited about the nature of digital sound and image and how these media can be related. The textural component of this thesis is an aesthetic choice but it has also provided a scientific and research based framework to keep the work grounded.

Audiovisual Granular Synthesis is a term I have introduced to deal with the problems presented and discussed by previous attempts to marry the audio and the visual in the digital realm. This solution includes a method for implementation of the creation of audiovisual grains (the basis of the synthesis type I am introducing) and a second implementation for the tools used to perform textures built from these audiovisual grains.

3.2.1. The Audiovisual Grain

The definition of an audiovisual grain in terms of my thesis is a grain that is created by the pairing of an audio grain as previously defined with a small sample texture.

The composition of audiovisual grains was conceived of as a four-part process developed during the first part of my Fall Thesis Semester, 2003. These four parts were:

1) The composition of separate audio and visual grains
2) Grain analysis and the creation of intermediate representations
3) The pairing of these grains according to the results of step #2
4) The storage of these pairs in a database for later access

These four steps have been revamped and include many other more intermediary steps, but the performance software that uses these grains is now complete. I will now spend some time elucidating these four steps involved in the composition of grains, as they stand as a well-constructed document that serves to explain how the theory in this paper is applied, and then outline the methods I have pursued in terms of their performance.

The composition of separate audio and visual grains refers to the process of

creating and programming short bits of information (1 to 100 milliseconds in the case of audio, 64x64 to 256x256 pixels in the case of graphics) to later be joined, stored and played back. I have “sampled" larger images and created some synthetic textures using varying mathematical functions and other experiments in order to develop a large visual vocabulary for my system. This is an important concept in this thesis, and is prompted by Curtis Roads' attempts to make a Granular Synthesis system capable of real-time virtuoso performance. (Roads 2001) The performance capabilities of the method I am introducing must be capable of dynamic performances in both the audio and visual realm.

For the audio component of the final grain I have created synthetic textures and also sampled some existing digital sound files in order to have a wide sound palette. I am using mostly sampled sound for the audio source of the audiovisual grains because the variation that they offer both in terms of their sound character and endless supply is appealing to me on logistical and aesthetic levels.

3.2.2. Grain Capturing

Out of all of the methods I have employed to generate or capture audio grains, common audio editors like Peak for the Macintosh have been the most straightforward and aesthetically rewarding. Likewise for the visual grains I have used Adobe Photoshop to capture square Textures from existing digital images, or images I have scanned or imported in some other manner.

Sampling larger amounts of time than is necessary to make one audio grain results in fascinating synthesis when the grain is used as a “playhead” which cycles throughout the sample. The granulation of a sound file, as Roads refers to it, brings out many fascinating textural dimensions within the sound file.

High-resolution images are the best digital sources, as they often contain many different examples of possible texture sources. For example, when I sampled pieces of my skin and hair, these 64 x 64 pixel images came from scanning my face at 600 dpi, often creating images in the order of 100 times larger than the designated sample size necessitates. Having this much material made choosing areas to take from very interesting and also yielded great results.

3.2.3. Real-Time Performance

While the process of composition, analysis, pairing, and storage of the audiovisual grains raises mainly aesthetic questions, their usage in a real-time performance

setting raises mostly scientific or programmatic questions which I will outline in the course of discussing this final implementation section.

The existence of a database of audiovisual grains does not necessarily direct the method in which these grains can be performed once they have been composed. The nature of the grains, however, does demand that they be grouped and multiplied before they can make any significant perceptual sense -- it is this component that truly makes them "grains." A graphical or sonic texture is generally referred to as such because of the elements that compose them, and I will now discuss the methods I will be using to turn these grains into textures.

3.2.4. Graphical Texture Synthesis

The graphical style for this thesis involves the manipulation of fields of animated live textures synthesized in real time using the Zelinka-Garland jump map algorithm. Natural textures like hair, skin, open pores, and others of the variety listed above are animated and given life. Parameters of the texture, such as the size of grains, the number of grains, the activity of the grains and the randomness in the grain system, can be manipulated in real time using a software interface.

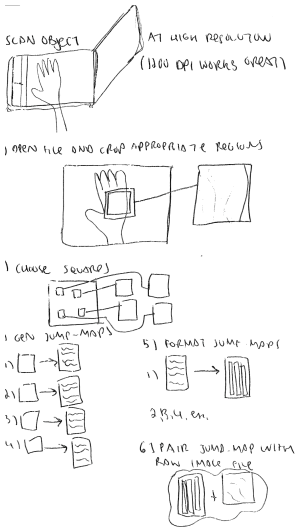


Figure 14. A sketch outlining my process for creating visual grains from scratch. A body part or object is scanned, a section of the scan is chosen, this section is turned into a jump map, and then a texture is synthesized from this section.

The visual components of this thesis owe heavily to Brakhage and the idea that he could tell the story of the life of a Moth by using textures from the Moth's wings alongside textural elements of grass, light, and other materials which speak to the existence of this creature whose absurd attraction to light almost always causes its death. (Brakhage 1963) The full screen textures I have created evoke moments of Brakhage pieces, and focus on the smallest elements that characterize them as natural and in many ways beautiful.

The jump map algorithm involves the creation of a data file that represents, for each pixel in the input texture, three other pixels who reside in a neighborhood with similar characteristics as the current pixel. In addition to the addresses of these pixels, the data file stores a floating point number (0…1) which represents a metric of "sameness" between the neighborhoods, determined by the jump map generation program according to a set of rules primarily the L2 norm.

By using the numbers generated in the jump map, the synthesis algorithm is reduced to the task of iterating through the output texture, copying pixels from the input texture, and occasionally jumping around the input texture according to the information in the date file, in order to maintain spatial visual coherence

throughout the output texture. The jumps, when probabilistically controlled, make up a large part of the reason why the Zelinka-Garland algorithm is capable of synthesizing natural textures, but does not always perform on structured textures. (Zelinka 2003)

Within the scope of a database of grains, I was determined to steer away from perceptually based and improvised selections, as I feel that a screen to display these grains would be cumbersome and unintuitive. I found it more interesting to create "scores" of a sort which limit the amount of possible grains to "pieces" or "compositions" where I see them fit. Within these compositions, I have included the ability to manipulate certain components of the graphics that will change parameters of the sound, and vice versa.

Given that the grains used in this performance are audiovisual in nature, it seems contradictory to proclaim that the graphics will control the sound or vice versa. However this is far from the truth -- the programmatic elements of the performance piece demand that one control the other. Rather than shying away from this component I have embraced it and made sure that the elements of control do not cause one component to dominate the other. Additionally, the sound component of the performance relies on many more stochastic strategies than do the graphics -- this is a function, to some degree, of the perceptual differences between sound and vision, and between Granular Synthesis and Texture Synthesis.

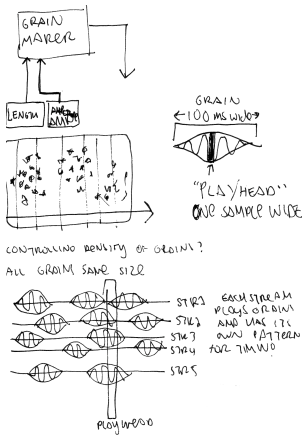


Figure 15. A sketch depicting the implementation of granular synthesis as outlined in *Microsound* by Curtis Roads. Grains, streams, and file granulation are discussed.

3.2.5. Audio Granular Synthesis

The granular synthesis engine of my performance tool utilizes the methods laid out in Curtis Roads' tome on granular synthesis, *Microsound.*

In making the connections between the audio and visual components of the grains, I have kept as strong of a relationship as possible between the spatial components of the graphics and the spatial components of the sound. Inherent to the concept of granular synthesis is the idea that the audio grains, while being played, are scattered throughout the stereo field. This has been scientifically proven and documented by Roads and others working in the field, and a detailed explanation of this concept is out of the scope, to some degree, of this thesis. It is important to note, however, that scattering these grains in cloud formations throughout the stereo field reduces the amount of overlap and hence allows more precise playback and detailed perceptual attention to each specific grain. The ear is extremely

sensitive to minor changes in the grains, and overlapping every grain in a monophonic output does not give the same effect as true stereo granulation.

Control over the parameters of sound in real time is a cornerstone of the performance system which I have designed and implemented using PortAudio/C++. The same parameters outlined in the visual section (grain size, grain population, grain activity and grain randomness) exist in the audio domain as well, abstracted as different properties which deal with controlling the audio.

3.2.6. Audiovisual Solutions

Using the idea that these grains need to be projected in the stereo field, I have decided to distribute the audio grains randomly throughout the stereo field as they are handled accordingly in the visual domain. Initial sketches I made for this thesis involved the "placement" of audiovisual grains, which has been overcome by the idea of mostly working with "fields" of audiovisual texture, in full screen and full speaker, so to speak.

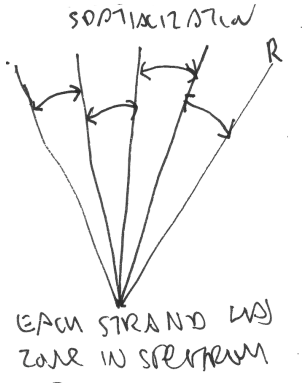


Figure 17. A sketch depicting the spatialization of sound grains. One of the early audiovisual connections I considered wass the positioning of sounds in the audio spectrum according to their positions in the visual texture.

In addition to the stereo treatment of the grains, a more fundamental question arose: how to determine how many audio grains to play in relation to how many visual grains exist on the screen. A 1:1 correlation is not only computationally oppressive, but perceptually unnecessary, as the amount of data capable of "filling" the ears is certainly not the same as the amount capable of "filling" the eyes. That is to say that our visual and audio capacities do not match -- this is a given. I have explored the question programmatically and have fine tuned the amount of audio grains to play at once. Using thirty-two grain streams at once sounds only nominally fuller than sixteen or twenty-four, and is more computationally oppressive. This speaks to the fact that the visual texture algorithm is more processor intensive than the audio component, and has more leeway built into it: it tends to slow down. I settled on sixteen simultaneous grain streams at 44,100 samples per second, which allows me to play an enormous amount of grains per second.

Finally, there is the issue of how to control the parameters of the audio grains with the parameters of the visual grains. These mappings speak to the heart of this thesis, and there has been much experimentation executed in this area. There were many obvious pairings that I wanted like to avoid, and more challenging pairings that I spent many hours exploring.

The abstracted parameters, as stated in the "ecosystem metaphor," speak to the system as an ecosystem that has as its population audiovisual grains. Each grain's size and even the number of grains which comprise the system at any one given time are controlled by the user. Additionally the amount of randomness in the ecosystem is controllable, along with the amount of activity.

It has been important to me that the visual textures are capable of evolving slowly or quickly, in a deliberate, creeping manner, or a manic, jagged manner. I decided early on that this would be a necessary component if the dynamism of the audio is to be matched. In many ways the two components direct each other, and their hybrid creation maintains characteristics of both.

3.3 Prototypes and Programmatic Exercises

This section outlines a production and research history that leads up to the finalization of the performance software that represents the technical component of my thesis. The algorithms used for the implementation of granular synthesis draw inspiration conceptually from Curtis Roads and Iannis Xenakis, and technically to Golan Levin, from the pseudocode section of his thesis, *Painterly Interfaces for Audiovisual Performance.* (Levin '00) All of the programming described in this section was completed in C++ using Portaudio for sound production and OpenGL for graphics. Using these libraries with C++ makes for a cross-platform, easily portable set of code that I could compile on any platform supported by the Libraries (Windows, Linux, Mac, etc.).

The subsections that follow are a history first of the construction of the audio component of the thesis, the visual synthesis method, and the interface, programmatically and graphically, to the program. Following this there will be an overview of the final program interface,

3.3.1. Realizing Sample Based Granular Synthesis

In the following subsections I will outline my methodology in realizing sample based granular synthesis, from sinusoids, to reading in sound files, to perform audiovisual textures.

3.3.1.1. Constructing Grain Streams

In order to move conceptually from "extremely fast Amplitude Modulation" to granular synthesis, I constructed a single grain stream that performed Amplitude Modulation on the micro-time scale. I used the sine wave generation code from

my mrbNoise project, and used a Hanning function with a fixed envelope size. (Bernstein '03) This procedural grain stream allowed me to hear the sound of asynchronous granular synthesis for the first time as I programmed it, and was very inspirational. By scripting the frequency changes in the grain, I was capable of hearing what frequency transitions in the granular domain sounded like, and my quest had officially begun. This procedural, non Object Oriented grain stream was extremely limited in that it did not offer the ability to be cloned and multiplied. The next step was to construct a class that contained the grain stream and all of its properties.

3.3.1.2. Object Oriented Grain Streams

Once I was satisfied with the procedural grain stream and had worked out the various properties that needed to be included in the class, I began to construct it. I defined a set of variables for the class: duration, starting time, current time, envelope index, frequency, and pan. Additionally I created two arrays for the class to hold, including a sine wave table and a Hanning window function computed to a fixed duration.

In terms of functions for the grain stream, I had varying success determining which were necessary and which were not. After some experimentation, I ended up with a function to construct the grain stream (the constructor), a "next" function to advance to the next sample for output, a "setPan" function to set the panning of the grain stream in question, a "setDuration" function to set the duration of the grain, a "playTable" function to play the sinusoidal wavetable, and a hanning function to compute the hanning table.

Once the class was constructed, I recreated the results in 3.3.1.1 using the new object oriented code. The results were similar but less prone to error during expensive frequency transmissions. It was during the construction of the class that I decided to work in samples instead of seconds as the main scheduling component in my granular synthesis program. When the sampling rate is 44100 samples/second, each sample is 1/44100 of a second long. To constantly convert samples to seconds and then back to samples was cumbersome and by the time I had gotten this far I was accustomed to converting samples to seconds in my head. I marked down some common time lengths in samples and memorized them, and their relative positions made it easy for me to comprehend the code when it was scheduled in samples. Particulary difficult in the beginning for me was calculating how many samples the grain duration should be. I settled on a range of 1800-

5000, which represents a time span from .04 to .11 seconds, a normal range, according to Roads, to deal with in granular synthesis. (Roads '01)

The grainMaker class having been established, I moved on to constructing and playing multiple grain streams at once, in order to begin to build complex clouds capable of producing lively sonic textures.

3.3.1.3. Multiple Grain Streams

The pursuit of multiple grain streams in C++/Portaudio has been very fruitful for me, and I am able to comfortably play 32 simultaneous grain streams over a long period of time, with frequency and duration changes with no audible error. At 1800 samples or .04 seconds, this is approximately 800 grains per second, a very large number considering the benchmarks used in other publications, where 100 grains a second is referred to as "a large number." (Roads '01)

Once I was capable of playing these grain streams and made corrections for grain overlap and clipping prevention, I moved on to the challenge of computing a large number of Hanning tables for use throughout the program. This would make it possible for grains to have varying durations and move seamlessly from duration to duration without having to constantly calculate envelope values. I pre-cached 100 Hanning tables in a structure inside the program and then gave access to these values from within the class. This proved to be a worthwhile optimization scheme, which takes the weight off of the processor in terms of calculating the Hanning window during every Portaudio callback, several times a second.

Randomizing the frequencies over the various grain streams gave subtle harmonic effects when the frequencies within the streams stayed constant but varied over the whole. Changing stream frequencies from frame to frame gave a bubbling sound that was lively but aimless. Restricting this movement, however, and changing it over time led to some interesting timbres that were fluid and natural.

Having achieved the benchmark of multiple grain streams containing sinusoidal wavetable data, I then played around with other waveforms such as square waves, pulses, and noise. Mixing these wavetables together had very interesting effects.

Desiring the ability to push the limits of granular synthesis in my own framework, I began entertaining the possibility of hooking the grain streams up to pre-sampled sound file data. This task proved to be more difficult than I had originally

imagined.

3.3.1.4. Reading In a Sound File

When I began to search for libraries that are capable of reading in sound files, I found that like many libraries which are available for access within the C++ language, most of these were "bloated," or contained many features which I would not use and hence would drag my program down. I additionally suffered from compatibility issues due to the various methods of compiling executables for the OS X platform (Carbon and Mach-o). I had so far been developing my application in Carbon, which makes it more suitable for cross-platform development. The libraries that I was investigating to read sound files were primarily of the Mach-o persuasion, and transferring from one to the other, which I attempted over the summer as part of the mrbNoise project (to include MIDI), is very difficult and cumbersome.

I began to port the project to Mach-o when I remembered that Processing had a function that made it quite simple to read in sound files. Recalling the work I had been doing in Zach Lieberman's "SetPixel" class, I began to wonder if there was a simple method, using Processing, which I could use to produce raw, C++ friendly sound files with no header.

3.3.1.5. Java/Processing Applet

Processing is a cross-platform, open source initiative started by Ben Fry and Casey Reas from the MIT Media Lab. They began to develop it as a learning platform (a la DBN) for designers interested in interactive artwork, and it has since developed into a mature language capable of stunning results. As is commonly the case with Processing, the commands for reading and modifying a sound file were extremely well documented and worked very well. I simply hooked up one function which read sound files to another which outputs text files, and I had a program to produce raw, 8-bit, headerless binary text files which I could feed to my C++ framework in the same way I was processing image files in Zach's Image Processing class. This program, which is around 15 lines long, looks like this:

```
BSound samp;  // Establish a placeholder for a soundfile

void setup()  // Initiate the setup routine
{
  samp = loadSound("stone.wav");  // Assign an actual file
  // Place the file into an array
  byte[ ] samps = new byte[samp.length];
```

```
  // Print the length of the sample to the console
  println(samp.length);

  // Go through all of the samples
  for (int i=0; i<samp.length; i++) {
    byte a;  // Choose a byte
    a = (byte)(samp.samples[i])+127; // Add constant value
    samps[i] = a; // Place the byte into the array
   }
 saveBytes("stone.txt", samps);  // Save to a text file
}
```

The "//" marks throughout the program are comments, which are bits of code which are there to help the programmer and cannot be read or interpreted by the computer.

Once this program was completed and deemed successful, I created several 5 second mono 8-bit sound files and began to reorganize the logic of the mrbGrains program to fit these sampled sounds.

3.3.1.6. Multiple Grain Streams Reading Sound Files

I returned to the framework that had 32 grain streams playing sinusoids, and decided to divide these 32 grains up into four sections of 8 which would each play the same sample. This would let me freely mix the samples as they were being granulated, which was a final goal of the performance tool.

The first implementation allowed each grain to randomly choose a bit of sound from one of the four samples. This sounded noisy and interesting yet it did not offer the amount of control I was interesting in executing over the streams and overall sound. I decided instead to divide the streams up into four groups that would allow for a constant number of grains playing each sample at any given time. The first 8 grains grabbed the first sample, the second 8 grabbed the second sample, etc. In this manner I was capable of more precisely controlling and monitoring performance during these stages.

The capability to simultaneously granulate four sound files with 32 grain streams was encouraging but I knew I needed to add interactivity in order to truly test the limits of the system. It was then that I decided to implement the GUI I had written for mrbNoise in order to quickly test the various parameters at hand.

3.3.1.7. Interface and the GUI

The mrbNoise program had a GUI that was simple, effective, and coded in a

relatively coherent manner, so it offered an easier and more elegant solution than any GUI library I could have found to work well within the Carbon framework. I began by invoking a simple x/y controller that allows for the simultaneous control of two parameters with the mouse. This interface was created in GLUT, a subdivision of OpenGL that allows for easy windowing, mouse and keyboard control.

Once I had the x/y controller hooked up to something (the duration of the grains in any given stream, for example), I wanted to increase the number of windows I had access to. The mrbNoise program had 7 or 8 windows that were crudely not object oriented, and to begin with I followed a similar path with mrbGrains.

3.3.1.8. Interface Design

A bit of Interface Design was necessary at this stage, where I knew interactivity was key for testing real-time capability. I devised a system whereby the user would control parameters of the grains from the point of view of the different samples being granulated. Since there were four samples, I would create four sets of windows and GUI elements to correspond to the sample.

3.3.1.9. Object Oriented Windowing

Achieving Object Oriented windowing for this section of my thesis development was only a partial success. I created a simple bit of code for each window type (output window, slider, x/y controller, etc.), and can create complex interfaces comprising of these bits of code.

3.3.1.10. GUI "Plumbing"

The Object-Oriented windowing code complete, I began to insert it throughout the program to test its capabilities. The GLUT system works extremely well, and OpenGL and Portaudio are quite complimentary libraries. Duration, frequency, jitter, and volume are all being controlled with the GUI as are the position of the grain on the sample and the panning of each grain stream. Keeping these variables as neat as possible and reducing them to the fewest amount possible (one for x and one for y values in the square x/y windows, just one for y in the slider windows) helps to prevent clutter within the program.

3.3.1.11. Interface Conclusions

At this point in my programming and conceptualization I realized that an entire section of my thesis Methodology needs to be focused on the software interface of

my performance software. The interface thinking and programming which I undertook during the creation of the audio component of my thesis were essential steps in finalizing an interface concept and executing it.

3.3.2. Visual Component: Texture Synthesis

Texture synthesis is the mechanism by which real time animated, non-tiled textures can be achieved. The visual dream of this thesis, to synthesize textures of the caliber achieved in early abstract films like those of Stan Brakhage and Harry Smith, can be realized using the principles of texture synthesis.

Animation in the realm of texture synthesis demands real time performance, which up until Zelinka and Garland published their papers on jump map based texture synthesis, was not available in a freely available algorithm or anywhere else. I have achieved animated texture synthesis with an OpenGL framework, with Jump Maps I generated using Zelinka's published Windows Software for jump map generation. Using this program, I was able to focus my attention on synthesis, the component that directs the majority of the "look" of the thesis.

The methodology I followed to execute the visual component of the thesis, texture synthesis, will be outlined here.

3.3.2.1 Real-Time Texture Synthesis

It is possible to synthesize visual textures in real time by restricting the types of textures that the synthesis algorithm is capable of handling. By limiting my palette to textures which are naturally derived and therefore by necessity somewhat stochastic, I am capable of synthesizing textures that other algorithms are incapable of doing in the same amount of time.

When compared to the texture synthesis algorithms that Zelinka and Garland, Efros, etc., have published, Audiovisual Granular Synthesis has an entire extra dimension to deal with: time. Once I was capable of synthesizing textures in real time with a C++ application, it became apparent, as predicted, that temporal coherence, or temporal aliasing, would be a concern. Visually, it is a fractured and jarring experience to witness a random generation each frame, even if the palette of the randomness is extremely limited. It is necessary, within the algorithm, to account for the fact that temporal concerns must come into play during the process of selecting pixels, making jump decisions, etc.

Considering this extra dimension is a concern with respect to the dynamism of the system in the visual sense. The ability to transition from slowly evolving to quickly randomly changing textures is key within the compositional structures I have aimed for. The sound component is incredibly dynamic and can act as a thoroughfare for any number of different sound types, lengths, etc. A major concern of mine during the programming of this thesis was to create a visual system which could match the dynamism of the audio component.

Controlling these dynamics comes down to keeping a history for each pixel which "remembers" what color the pixel in any given position was in the last frame, two frames, etc. Using the amount of these pixels, which repeat their histories (I call them "keeper pixels"), as a value with a gauge or "knob" in the system, I can effectively control sameness, and therefore how rapidly or slowly the texture evolves. This methodology is unique to this thesis and represents an adaptation to the Zelinka-Garland algorithm that acts in similar ways to many of their own methods – a manner of controlled randomness.

I have implemented the "keeper pixels" and have demonstrated them along with the granular synthesis as a prototype for this thesis. I have also investigated but not implemented methods of tweaking the statistical components of the jump map in order to weight the jumps according to jumps which have already been made, a sort of "fine tuning" knob for keeper pixels.

Real Time texture synthesis is an essential component of Audiovisual Granular Synthesis, and the methods used in the Zelinka-Garland algorithm do not greatly differ from the methods used in the audio granular synthesis. Combining these two became much easier and more natural once the visual component fell into place.

3.3.2.2. The Analysis / Synthesis Schism

Central to the concept of real-time texture synthesis is the separation of the work into two main operations, one concerned with analyzing the input texture, and the other involved with synthesizing the texture for the novel, larger output. These operations break down into a program that analyzes the image to generate the jump map and makes a lot of statistical conclusions about the input image during the process, and a program which implements the synthesis algorithm and offers visual credence to the concepts contained therein.

Because the analysis of the input texture, and the operations involved in comparing an extraordinarily large number of pixels to each other, is extremely expensive computationally, the need to only run these routines once are extremely beneficial. In this way, even a texture whose jump map takes a few minutes to generate, which is very long using Zelinka's texture_analyze program, only takes a fraction of a second to synthesize.

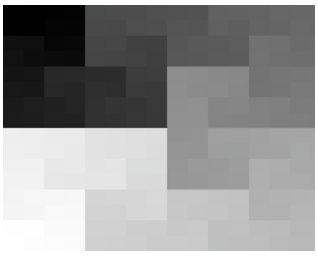


Figure 18. A visualization of the Hilbert path, from beginning to end (black to white). Notice the square artifacts. This caused initial troubles with my implementation of the texture synthesis algorithm, but have since been removed. The fractal quality of this image delineates the methods of creating it.

Synthesizing the output texture involves following a path along the output texture and placing pixels along the path according to the condition of its immediate neighbors. Zelinka and Garland discuss various paths to follow and settle on a space-filling curve known as the "Hilbert Path" when iterating through the output texture pixels. The reasoning behind this will be discussed more extensively in the section devoted entirely to the synthesis algorithm.

3.3.2.2.1. The Jump Map

An integral component of synthesizing novel textures is being aware of the spatial information of the input texture at all times. This intuition is confirmed by nearly every texture synthesis algorithm since its inception. From the filtering based methods which relied on theories of psychophysical theories of eye movement and visual perception to the tiling based methods of recent years, spatial preservation is the key component or "problem" that any algorithm has to solve.

The solution which the jump map algorithm offers is to pre-compute, for each pixel in the input image, three addresses elsewhere on the input image which reside in neighborhoods which are similar to that pixel. Along with this address, there is a metric of "sameness" which is applied to each of these three choices, a floating-point value between 0 and 1.

In order to determine these values, and build the jump map, Zelinka and Garland, in their 2003 paper on the subject, outline methods for executing these comparisons in an extremely quick and efficient manner. Based on the desire of the algorithm to preserve spatial sameness, according to the principle that "This texture synthesis algorithm is based on the assumption that "the neighbourhood of each output pixel to be synthesized will closely resemble the input neighbourhoods from which the nearby already-synthesized pixels were copied." (Zelinka 2003)

In the program that generates the Jump-Map, windows software which runs inside of a "Linux-like environment" called Cygwin, several statistical operations and

evaluations are made on the input texture. Based on the idea that if each pixel has a set of options in the form of addresses of pixels in similar neighborhoods that synthesis through iterative pixel selection (per-pixel processing) will be more coherent over the entire surface of the output texture, the texture analysis software is chiefly concerned with comparing every neighborhood in the input texture with every other neighborhood.

A neighborhood is defined as a group of pixels that are spatially continuous, and usually takes the form of squares from 3x3 to 15x15 pixels. By working with varying neighborhood sizes, it is possible to synthesize textures with different granularities in terms of the details of their contents. For example, a texture containing pebbles may benefit from a smaller neighborhood size than a texture containing a boulder or two. These rules are not hard and fast however, and certainly vary from texture to texture.

Comparing every neighborhood to every other neighborhood, using a process known as "brute-force comparison" is not an appropriate solution for this thesis because of the sheer amount of time this would take for a series of textures with varying neighborhood sizes. Zelinka and Garland have applied statistical techniques to reduce the dimensions of the vectors being compared when each neighborhood is stretched out and considered in the vector domain. The steps involved in generating a jump-map for a texture include:

1) *Choose an input texture* – this input texture must be a square, and a power of 2

2) *Create a Gaussian Image Pyramid* - for this input texture, create an image that consists of multiple resolutions of the same image superimposed on each other. This maintains and emphasizes spatial information and makes the image more susceptible to further vector dimension reduction.

3) *Principal Components Analysis* (PCA) – in order to reduce the dimension of the vectors being compared, that is, in order to decrease the number of values which need to be considered in the analysis phase, it is necessary to run a scheme similar to Eigenvalue extraction on the input texture. Consult appropriate literature about statistical analysis and reduction of large dimension vectors for explanations of these principles, as they lie somewhat out of the scope of this thesis. To simplify the issue, PCA attempts to summarize the information in the input texture to the extent that operating comparisons between fewer objects could be some percentage as accurate as operation comparisons between every neighborhood.

4) *Approximate Nearest Neighbor Searching* (ANN) – using the remaining vector dimensions after PCA is complete, ANN attempts to find the

neighborhoods which are closest to the neighborhood being analyzed. This is another method which is capable of "sampling" some percentage of the original image in order to approximate analyzing the entire image.

5) *L2-Norm Measurement* – Once the remaining vector dimensions are in place, it is only necessary to go through the remaining components of the image and determine how similar they are to each other. The top three most similar neighborhoods are stored, along with a normalized value representing the amount of sameness the neighborhoods share, in a data file which is the Jump Map.

Along with these steps, other techniques are applied to preserve spatial coherence and reduce the amount of time necessary to compute the jump map for a reasonably sized input texture.

Zelinka and Garland report that "brute force" comparisons can take up to 30 minutes. (Zelinka 2003) I have used the "texture_analyze" software that Zelinka has written, and normal textures take in the order of seconds to complete.

The Jump map is a simple data file with a .jtm file extension, which looks like:

128 128 // This connotes the dimensions of the texture
JumpList: 3 // This is the number of jumps stored for this pixel
33 56 .4500 // XPOS YPOS VALUE for this pixel, first jump
110 24 .5000 // Second jump, xpos ypos value
100 100 .821 // Third jump xpos ypos value

For the purposes of my program it was easier for me to strip this image down into all floating point numbers and read it into an array when the program is engendered.

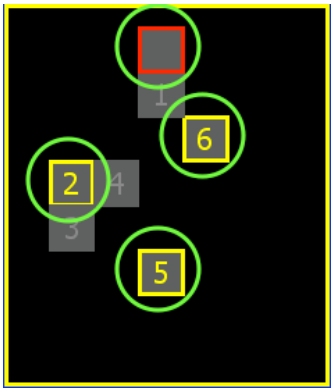


Figure 19. Mr. Zelinka's diagram explaining Poisson Disc Distribution, a method which ensures the reduction of vector dimensions during texture analysis. The red square represents the target neighborhood for analysis, and the other circled squares represent neighborhoods which are a certain distance away from the target neighborhood. This reduces the amount of overlap during comparisons. (Zelinka 2003)

Because of the extremely advanced nature of the vector math and statistical analysis involved in the creation of the jump map generation program, it was highly beneficial for me to use Zelinka's software. The overhead, time wise, involved in implementing these various mathematical methods and functions is overwhelming. Additionally, my choice to program this software on the Macintosh platform, a choice which has afforded me an incredible amount of leeway in terms of its ability to process synthesized frames with extreme speeds, has also handicapped my abilities to implement the libraries which have been used and created by other programmers and engineers interested in PCA, ANN, Eigenvalue extraction, *etc.*

Because the creation of the Jump Map is kind of a "given" and will not really change with changes in the synthesis algorithm, in many ways Zelinka and Garland have split up the "science" and the "art" of the problem of texture synthesis. While

the decisions to use such beautiful mathematical concepts such as Poisson Disc Sampling (a scheme to avoid overlapping in neighborhood searches, generally used in N squared N problems to reduce the dimensions of the comparisons) are quite artful, the visual components of placing pixels are all left to the synthesis component, which is occurring in real time. Here we see the true brilliance in the algorithm and the justification for the Jump Map itself.

3.3.2.2.2. Jump Map Based Texture Synthesis

While it may seem counterintuitive that natural, partially stochastic textures may conceivably be easier to synthesize in any context, due to their complexity and natural beauty, Zelinka and Garland's algorithm has the distinction of working best with these types of structures. It is ironic that a method like Jump Map Based Texture Synthesis, which uses no psychophysics, no filtering or steerable pyramids, and none of the other flourishes or mathematical tricks that other computer graphics algorithms which fetishize naturalism rely on, can perform so well under these circumstances.

The output texture is the main target of this component of texture synthesis, which in fact consists of the synthesis itself. In a setting like Audiovisual Granular Synthesis, where texture synthesis will be animated and performed in real time, it is important that the algorithm for synthesis is efficient and works well in a computationally taxed setting.

The objective of the algorithm is to cover the output texture with a larger, novel texture synthesized from the jump map that was generated by analyzing the input texture. By following a path through the output image, it is possible, using the jump map, to synthesize a new novel texture by using simple rules which, when running side by side, constrain the development of the image to reproducing, in a manner of speaking, a new novel texture from the input texture.

3.3.2.3.1. Pixel Ordering

As Zelinka and Garland have pointed out, one of the major factors involved in synthesizing for an output texture is the order of the pixels that you follow during the synthesis algorithm. (Zelinka 2003) The three major paths they investigated were scanline, which is a traditional method of starting with the 0,0 pixel and continuing to the width,height pixel, line by line. Serpentine order follows lines but instead of going from 0, Width to 1,0, it would go from 0, 0 to 0,Width, to

Width,1 to 0,1, etc. Finally Zelinka and Garland present a special type of path that is classified as a space-filling curve, known as the Hilbert Path.

A space-filling curve is a curve that hits all of the pixels in a space without crossing over itself or repeating any pixels. The Hilbert path is such a curve that works with squares whose sides are powers of two, perfect for texturing scenarios where textures are often limited to these sizes and dimensions anyhow. (Zelinka 2003) Understanding the reason why following the Hilbert Path is important goes along with understanding the texture synthesis algorithm as a whole.

Starting with a random point on the output texture and following the Hilbert path from there, the output texture is filled in with values by determining which pixel from the input texture to copy to the current pixel. In an animated setting, these pixel decisions occur Texture_Width*Texture_Height times per frame, or over seven million times per second for a 512x512 texture.

3.3.2.3.2. Temporal Aliasing and the "Keeper Pixels"

The questions of the algorithm involve how to choose which pixel from the input texture to choose, and when to jump to another neighborhood according to the jump map, and of these jumps, which jump to take. The temporal domain involved in animating texture synthesis adds the possibility of the "keeper" pixel, that is meant to create stronger temporal coherence by forcing sameness between frames.

The decision of which pixel to take from the input pixel in a situation where there is no jump occurring, we simply choose a neighbor from the available pixels (those which have already been synthesized). For example if the last point on the curve is the pixel below the current pixel, the current pixel would take its value from the pixel above the address used for the last pixel, the corresponding pixel. This is a strong measure to ensure spatial cohesiveness across the surface of the output texture, and it works very well as a backbone to the jumps that are involved to ensure randomness to work well with the semi-stochastic natural textures I thankfully desire to synthesize.

The question of when to jump and which of the jumps to take, and when to use a keeper pixel, are handled the same except for one situation, which is known as the border condition. Every other instance when jumping is desirable is controlled by probability and statistically controlled environments, which ensure that events

occur in certain ratios to each other and in certain ratios in relation to the entire output texture.

The statistical control over these decisions allows for great leeway, and therefore the control over the graphics is similar to the control over the audio. Granular synthesis is dependant on the idea that, when composing textures out of smaller elements, the statistical control over the placement and properties of those elements is an essential set of controls to have in order to ensure a dynamic textural system.

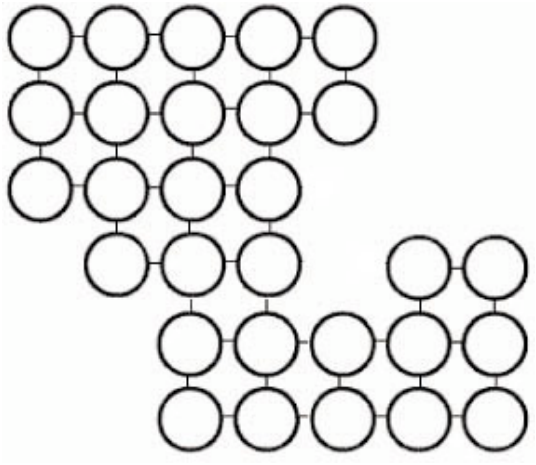
The Jump Map allows the animated textures to take life and change over time. The augmented, time dimension handling Jump Map ensures temporal coherence by forcing sameness frame to frame in the same way that the original algorithm did spatially.

3.3.3 Performance Software Interface

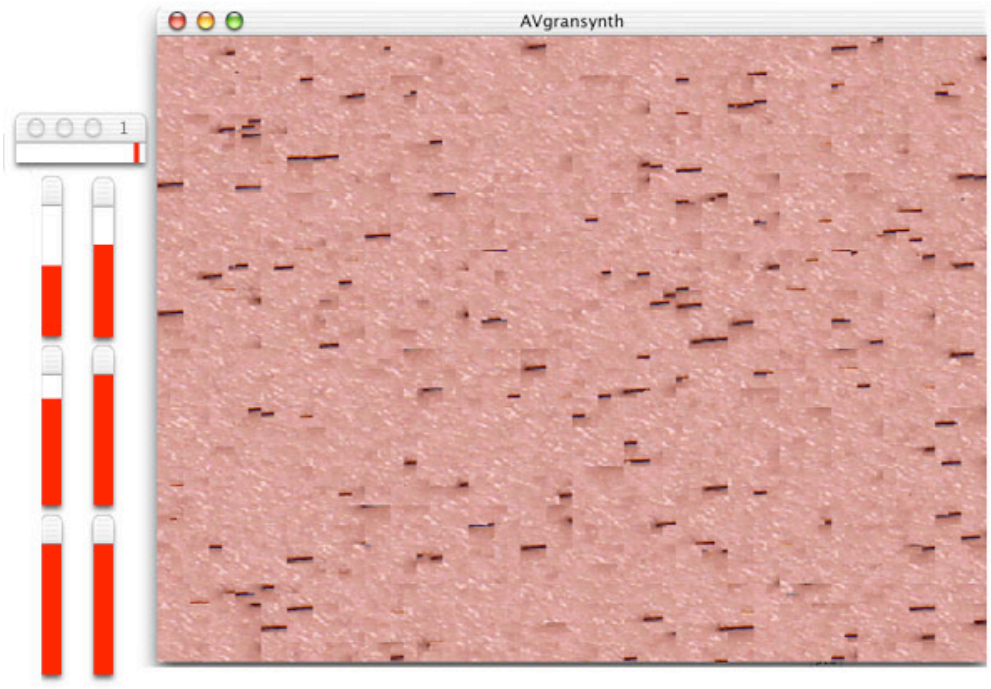
As the other subsections in this Methodology are mostly concerned with the implementation and fine tuning of the algorithms which when combined make up Audiovisual Granular Synthesis, this subsection will deal with the software interface which allows for the real time performance of the audiovisual textures I have created.

I will first discuss the narrative structure that is tied into the interface, and I will then proceed to discuss the graphical user interface itself. In this way I can discuss the conceptual components of the interface and their visual implementations separately.

3.3.3.1. Narrative Map



3.3.3.2. Graphical User Interface



3.3.3.3. Interface Advancements

4. Evaluation

4.1. Evaluation of Audio Component
4.2. Evaluation of Visual Component
4.3. Evaluation of Audiovisual Solutions
4.4. Evaluation of Interface
4.5. Evaluation of Performance

5. Conclusion

Audiovisual Granular Synthesis is a solution to the problem of how to relate audio and visual elements in a textural performance context. By investigating and unifying advanced synthesis types in the visual and audio domains, Audiovisual Granular Synthesis intends to prove that linkings on the micro scale are beneficial to creating the feeling of unification between the audio and the visual, toward the end of creating a series of performable, animated, audiovisual textures.

6. Works Cited / References:

Books / Articles

Ashinkmin, M. 2001. *"Synthesizing natural textures."* In *Proceedings of 2001 ACM Symposiom on Interactive 3d Graphics.* ACM SIGGRAPH, North Caroline, 217-226.

Bencina, Ross. *"Implementing Real-Time Granular Synthesis."* Self-Published Draft, 2001.

Brakhage, Stan. *Essential Brakhage*. New York: McPherson & Company, 2001.

Chusid, I. *Songs in the Key of Z*. Chicago: A Capella Books, 2000.

Cowell, Henry. *New Musical Resources.* Reprinted in 1969, Something Else Press, New York. New York: Alfred A. Knopf, 1930.

Criminisi, A. et al. *"Object Removal by Exemplar-Based Inpainting."* Proceedings of CVPR 2003, Madison, Wisconsin, 2003.

Efros, A. and Freeman, W. *"Image Quilting for Texture Synthesis and Transfer."* Proceedings of SIGGRAPH '01, Los Angeles, 2001.

Harley, James. *Persepolis Record Review.* Computer Music Journal. Cambridge: MIT Press, Winter 2001.

Harley, James. *"The Electroacoustic Music of Iannis Xenakis."* Computer Music Journal. Cambridge: MIT Press, Spring 2002.

Liang, Lin, et. al. *"Real Time Texture Synthesis by Patch Based Sampling."* ACM Transactions on Graphics, 2001.

Levin, G. *"Painterly Interfaces for Audiovisual Performance."* Master's Thesis in Media Arts and Sciences, Massachusetts Institute of Technology, 2000.

Papoulis, A. *"Brownian Movement and Markoff Processes."* Probability, Random Variables, and Stochastic Processes, 2nd ed. New York: McGraw-Hill, pp.515-553, 1984.

Perlin, K. "An image synthesizer." Proceedings of SIGGRAPH '85, 1985.

Roads, C. *"Multiple Wavetable, Wave Terrain, Granular, and Subtractive Synthesis."* The Computer Music Tutorial. Cambridge: MIT Press, pp. 168-184, 1996.

Roads, C. *Microsound*. Cambridge: MIT Press, 2001.

Sitney, P. Adams. *Visionary Film: The American Avant-Garde, 1943-2000.* UK: Oxford University Press, 2000.

Trocco, Frank and Pinch, Trevor. *Analog Days: The Invention and Impact of the Moog Synthesizer*. Cambridge: Harvard University Press, 2002.

Xenakis, Iannis. *Formalized Music: Thought and Mathematics in Composition*. Indiana: Indiana University Press, 1971.

Zelinka, Steve and Garland, Michael. *"Jump Map-Based Interactive Texture Synthesis."* ACM Transactions on Graphics, Vol V. No. N, September 2003, Pages 1-31.

Zhu, S. et al. *"Filters, random-fields and maximum-entropy (Frame)."* 1998.

Recordings

Henry, Pierre. *Variations Pour une Porte et un Soupir*. (Variations for a Door and a Sigh), 1965.

Roads, Curtis. *Klang-1*. 1974.

Roads, Curtis. *Prototype*. 1975.

Stockhausen, Karlheinz. *Kontakte*, 1959. Performed by James Tenney and William Winant, 1978.

Xenakis, Iannis. *Analogique A-B* for string orchestra and tape. 1959.

Xenakis, Iannis. *Persepolis*. 1971.

Films

Brakhage, Stan. *Eye Myth*, 1972.

Brakhage, Stan. *Mothlight*, 1963.

Brakhage, Stan, *The Dante Quartet*, 1987.

Smith, Harry. *Early Abstractions.* 1964.

Smith, Harry. *Late Superimpositions*. 1964 .

Smith, Harry. *Mirror Animations*. 1979.

Smith, Harry. *Oz: The Tin Woodman's Dream*. ca. 1967.