Freesound Radio: supporting collective organization of sounds

Gerard Roma Trepat

Master Thesis submitted in partial fulfillment of the requirements for the degree:
Màster en Tecnologies de la Informació, la Comunicació i els Mitjans Audiovisuals

Supervisors: Perfecto Herrera, Xavier Serra

Departament de Tecnologies de la Informació i la Comunicació
Universitat Pompeu Fabra. Barcelona, Spain
September 2008

Abstract

This thesis addresses the problem of creating music collectively on the basis of a shared repository of sound files. The classification of sounds in an heterogeneous database according to both a semantic taxonomy and a music oriented taxonomy based on content descriptors are analyzed, and their application to music creation is discussed. A data structure is presented that allows expressing musical ideas as combinations of sounds in the database, along with an interface where users can create and share such combinations. Finally, an interactive genetic algorithm is devised that automatically generates new combinations of sounds from existing ones influenced by ratings and preferences of listeners. The application of this concepts is demonstrated in Freesound Radio*, an interactive radio station developed on top of the collaborative sound database* Freesound*.*

Acknowledgments

I would like to thank Xavier Serra for giving me the opportunity me to work at the Music Technology Group and for his proposal and guidance of the *Freesound Radio* project, as well as Perfecto Herrera for his continuous support and advice. I would also like to thank Bram de Jong for his support from the *Freesound* side.
This work would have neither been possible without the help of all my colleagues at the MTG and the TICMA master. In particular I am indebted to Graham Coleman, Martin Haro, Amaury Hazan and Anna Xambó for their help with early prototypes, and to Jordi Funollet and Markus Koppenberger for their support with system administration. Finally, I'd like to thank Josep Guasch, for his interest and feedback, along with all the known and unknown visitors who participated in the project.

Gerard Roma
Barcelona
October 1, 2008

To Anna

If this word "music" is sacred and reserved for eighteenth- and nineteenth-century instruments, we can substitute a more meaningful term: organization of sound.

John Cage

Contents

Chapter 1

Introduction

1.1 Sound files

Sound files are now everywhere. Almost anywhere a computer is involved we find sound files. They are used in all kinds of digital media: music, games, videos, and so on, as well as in user interfaces. Large quantities of sound files are now carried around in mobile phones and portable devices such as music players.

Obviously there is a difference between music files, which are massively distributed for entertainment, and files containing sounds for being used in media productions. From a cultural point of view, the difference becomes blurry very quickly as people find all sorts of creative uses in commercial recordings. However, there are a number of practical issues that preserve the interest of such distinction, including among others copyright law and traditional licenses, duration of the recorded sound (and consequently size and bandwidth usage), dynamics and sound quality. For the purpose of this work we will rule out sounds that are distributed as finished works, although it will be in fact in our aims to help the departure from this asymmetric producer/consumer model of communication that dominated most of the XXth century.

Sound files have become prevalent in many applications that require the production of sound. Again, mobile phones provide a god example of this transition: cheaper synthesis techniques were used for ring tones until the chips became efficient enough to play sound files. This evolution towards the use of sound recordings has happened all around and may be considered a matter of usability. An audio file is a very simple thing, regardless of how complex is the sound contained in it. This is very convenient, since all computer users are familiar with the concept of a file: sound files are no less files than video, text or image files. They are fixed, they don't contain information about the production of the sound, they contain just sound. For the purpose of music production they are less flexible than a synthesizer, and for

this reason they are easier to use.

However, one important difference in the use we make of sound files with respect to other media is the level of abstraction. While the comparison of the use of recorded sounds to make music with visual collage has become a *cliché*, the reality is that music does not usually represent real world objects. Hence, music can be made by assembling recordings without the listener knowing much about the production process. A good deal of research in technologies for music production in the last two decades can be seen as a dialogue between physical modeling and spectral modeling of sound (Serra 2007). By focusing on the way sounds are produced, physical models offer control over many parameters that influence the production of sounds. In this sense, physical modeling (including models of electronic devices) has become widespread for applications to music production. On the other hand, spectral models deal with the mechanisms of sound perception, so that they allow to manipulate sound with a much greater level of detail and realism. Hence, spectral models and spectral analysis have been extensively applied to technologies related with sound reception, especially in speech and music analysis, and are now standard for applications that require realistic imitation of voice and musical instruments.

Historically, the rigid nature of sound recordings has allowed the birth of discourses whose aesthetics are based on the selection and combination of sources, and do not depend on complex gestures or instruments. Because in general no academic training is needed, these practices are widespread in popular culture, and continuously present new applications for spectral modelling beyond indexing and retrieval of music files.

1.2 Music made from recordings

The possibility of making sound with recordings was kind of awaited. It is notorious how Francis bacon predicted the possibility of mechanical production of any sound in *The New Atlantis* (published e.g. in (Bacon 1862)) as early as 1626. While misunderstood in its time, Russolo's futurism had stressed the need of escaping from the tradition of orchestral music. László Moholy-Nagy speculated, back in the 1920's, about the possibility to create music synthetically by working on the wax discs of the phonograph (available in (Moholy-Nagy 2004)).

Working as an engineer at the *Radiodiffusion-Television Francaises* (RTF), Pierre Schaeffer had access to the technologies that would allow him to experiment with

the idea of making music from fixed recordings, first with the phonograph and later using magnetic tapes. *Musique concrète* was born with the technique of splicing and gluing tape. Many composers would visit Schaeffer's studio and the technique was incorporated in the vocabulary of experimental music. Composers like Nono, Cage and Varèse made extensive use of the tape recorder. Schaeffer later published *Traité des objets musicaux* (Schaeffer 1966) with the aim to provide some methodological ground to the use of recordings. While the adoption of the ideas contained in the *Traité* has been slowed by its complexity and extension, the concept of reduced listening (*écoute réduite*), inherited from phenomenological *Epoché*, has become key in the development of electronic music by putting the focus on the perception of sound regardless of its source.

Apart from the possibility of splicing and gluing tape, a prominent feature of the tape recorder was the possibiity of creating tape loops. This technique had also an influence on the development of the repetitive music style of the sixties. Terry Riley pioneered this technique some of his early pieces (*Mescalin Mix, Music For the Gift*). Also Steve Reich explained (Reich 2002) how his own approach to this style was influenced by experimentation with tape loops (*It's gonna rain, Come on*). Tape loops were also used as primitive delay units, for example by pioneer Pauline Oliveros. The degeneration of sound produced by the tape medium itself was explored e.g. in Alvin Lucier's *I'm sitting in a room*.

With the popularization of integrated modular synthesizers like the Moog, synthesis gained the focus in electronic music production as being much more user friendly than the laborious task of splicing tape. In the seventies, electronic music was thus dominated by synthesizers, which started to appear in popular music. On the other hand, the publication of Murray Schafer's *The tuning of the world* (Schafer 1977) was an important milestone in the aesthetics of recording-based music . In his book, Schafer drew attention towards environmental sounds and their use for music composition, and introduced the concept of a *sonic heritage* that deserves some preservation just as other cultural aspects of quickly changing societies.

Inroduced in 1979, the Fairlight CMI marked the beginning of digital sampling. Digital samplers allowed to overcome all the inconvenience of manipulating tape by providing a much more precise control over the recording. Also, unless tapes, they allowed real time triggering and were designed to be played with a keyboard. Samplers became popular along with synthesizers, sequencers and drum machines thanks to the MIDI specification. Their primary use appeared to be the imitation of traditional instruments, but their potential was clearly much wider than that. Soon,

the reutilization of existing recordings using samplers was naturally exploited in hip hop music, where the turntable had already become a musical instrument. Along with hip hop, techno music became another important genre in popular electronic music. While making more use of drum machines and synthesizers, techno producers inherited from minimalist composers the habit of using loops as the basis for their repetitive compositions. A market appeared for distributing libraries of royality-free samples, be it instrument notes or ready-made loops, for speeding the production of electronic music.

As computers started to be powerful enough to play samples, they begun replacing both samplers and traditional tape based multitrack recorders. Computers made the management of sounds much easier, and disclosed an infinite potential for manipulation of sound. The graphical display of the computer allowed much more sophisticated sound editors than the ones present in samplers. With computers the utilization of samples to create composition was in the hands of everyone with a general purpose machine. An important aspect of using computers was that using samples was no longer tied to the tradition of musical instruments. It was suddenly possible for people with absolutely no background in traditional music theory or instruments to create their own discourse by editing digital audio. In this context, the usefulness of reusing existing music was so obvious that some people started to question the copyright law. Music has always evolved by copy and imitation, but digital audio allows to make it so explicitly that the clash between popular creativity and the interest of the music industry was unavoidable. *Plunderphonics*, named after John Oswald's album which was burned by the copyright authorities, became a musical genre based on the fight against the established laws and the industry over the right to use copyrighted material to create new music. At the same time, the tradition of soundscapes grew along with the progressive development of better and cheaper devices for field recording. Sound artists like Francisco López, advocated for a return to the acousmatic *pure listening* away from the shift towards representation introduced by Schaffer. Still, many artists in this style proudly restrict their music to mixes of recordings, refusing further processing of samples that was made possible by computers and generally abused in many electronic music genres.

With computers, the specifics of using samples diluted into the sea of possibilities for sound manipulation offered by the computer. However, the use of existing music, synth loops, field recordings and so on still carries much of these aesthetics sometimes with their political implications (fight against copyright, acoustic ecology). With the change of the century, internet has become the ideal breeding

ground for new ways of using samples by making it possible to share them around the globe.

1.3 Motivation

Computer networks are increasingly being used of communications, but in general they are less used for music production than one would expect. Music is in fact much of a social activity. However, computer music has remained rather individualistic by the design of computer workstations. This is certainly starting to change, especially as computer literacy and access to the internet is generalizing.

The *Freesound* [1] project was developed by Bram de Jong at the Music Technology Group during the organization of the ICMC conference in Barcelona [2]. The aim of the project was to support a large repository of sound files under the Creative Commons (CC) license that could be used for research. Over time, the site attracted a large and lively community of users interested in sharing sounds, be it field recordings, synthesizer loops, voice and instruments or any other kind of sounds. This enthusiasm in media sharing has become an important trend in internet usage, with sites like Flickr and Youtube becoming among the most visited in the world. While *Freesound* constitutes an interesting experiment in respect to sharing sounds as a way to report on personal experiences, this is more typically done through video and images, and the exchange of sounds is mostly driven by their potential for media and music production. Thus, this sharing activity seems to point to a further step: the combination of sounds in the database to form an audible discourse. On the other hand, as the number of sounds increases, finding interesting sounds becomes more and more difficult. A culture of tagging and description was established by design on the site, which helps finding sounds. However, looking for sounds for music creation remains difficult. As implied by the concept of reduced listening, the utility of many sounds is not necessarily connected to their source, which makes it difficult to rely on text descriptions.

The *Freesound Radio* project was intended to help in the discovery of interesting sounds by providing a continuous soundscape. It soon became clear that the output of the radio would not be interesting unless some kind of musicality or meaningful organization was present in the mix. Also, being the site driven by the community, it only made sense that such process would be equally community driven. The

[1] http://www.freesound.org
[2] http://mtg.upf.edu/icmc2005/

metaphor of a radio station was thus suitable: during the XXth century analog radio became a prominent way to connect communities through sound waves, allowing for more interaction (typically through phone calls) than television. The development of internet radios has been slowed by the lack of regulation with respect to copyright laws, but has regardless continued to grow. With computers invading the space of media distribution, consumers become active players, and no longer behave as passive spectators. Many possibilities are still to be explored regarding the creation of music through interaction among listeners. After all, *Musique concrète* was born in a radio station.

1.4 Goals and methodology

In this thesis, we present the methods and concepts that have been implemented in *Radio Freesound* in order to explore the possibility for internet users to drive a musical process. It has been shown (Tillmann et al. 2000) that musical structures are learnt implicitly through listening. Therefore, it should be possible from the basis of working with samples to see that music itself can be created directly from social interaction among users with no specialized education. This musical process can be seen as a collective exploration of the database. In this work, we don't make a great distinction between both: any sound that is played in the music is available under CC license for the listeners to use on their own compositions. Both music (especially at the initial stages of creation) and exploration often share the lack of very clear goals, which makes it difficult to objectively evaluate wether a given tool is adequate. This contrasts with the established methodologies in Music Information Retrieval (MIR) and task oriented Human Computer Interaction (HCI).

It has been debated in the field of Multimedia Information Retrieval wether the intent of bridging the so-called *semantic gap* between user semantics and content from bottom-up analysis is taking too much to reach meaningful applications (Shamma et al. 2007). In the mean time, users have shown they disposition to help themselves, organize in communities, create the content and help indexing it through tag folksonomies. Hence, it has been proposed that methodology is refocused in a more pragmatic approach to involve users in learning how to analyze and index media in meaningful ways. This should be made possible by throwing prototypes at users that allow the research community to understand what works and what is interesting. This information can be gathered by qualitative and quantitative evaluation

through log analysis. In a way it means that Information Retrieval research can progress iteratively not very differently from software development.

In this sense, the *Freesound Radio* prototype developed in the context of this thesis can be regarded as an initial step towards learning how a large shared database of sounds can be used for making music collectively. We report three main developments that were carried on for this purpose: initial experiments regarding how to organize and retrieve sounds from the database, a simple data structure that allows users in the community to encode ideas as combinations of samples, and an algorithm to ensure a continuous provision of content by remixing and mutating user creations.

The following chapter reviews three different traditions from which these concepts are respectively derived: Concatenative synthesis and Musical Mosaicing, Network Music and Evolutionary Computation. Chapter 3 contains two experiments for a combined top-down and bottom-up approach to sound object similarity and retrieval. Chapter 4 presents the concept of a *Sample Patch* as a means to allow people without specific musical training to express musical ideas from samples. Chapter 5 describes a genetic algorithm used to evolve new sample patches. Finally, in chapter 6 we discuss the application of these concepts to an online prototype, and in 7 we extract some conclusions and point to future developments.

Chapter 2

Background

2.1 Musical mosaicing and concatenative synthesis

Musical mosaicing and Concatenative Sound Synthesis (CSS) allow the application of sound and music description technologies to music creation. Especially in the case of CSS, they can be seen as an evolution of analysis/synthesis techniques that were developed in the 90s. *Musical Mosaicing* (Zils and Pachet 2001) was presented as a means to assist the retrieval of suitable sound samples in the context of music composition. By using high level descriptors to specify the musical work, samples could be retrieved according to their fitness to the specification. The problem of retrieving sounds from a large database was thus modelled as a constraint solving problem. The term was proposed as an analogy to the technique of mosaicing in visual arts to emphasize the use of descriptors at the compositional level, the retrieved then acting as the building blocks of the mosaic. This analogy may be reminiscent of the *Musique Concrète* metaphor, which was used to distinguish the use of *concrete* recordings from traditional *abstract* music (based on scores) in analogy with abstract art. In this sense *Mosaicing* could be depicted as a return to this abstraction, as content descriptors allow the use of abstract specifications to compose the work independently of the concrete recordings that are used to fulfill the specification. Thus, *Mosaicing* is sometimes used in conjunction with preexisting music compositions or scores (Jehan 2005), and in general requires an *a priori* definition of the musical work.

Concatenative Synthesis (CSS) was adapted to the field of music creation from the tradition in speech synthesis of using a corpus of sound samples to realistically emulate the human voice by concatenating them using spectral domain techniques such as Pitch Synchronous Overlap Add (PSOLA). Its application to music synthesis (Schwarz 2004) extended the concepts of musical mosaicing by defining the ability to transform retrieved samples mainly at the concatenation point. This application

has been clearly successful in applications that are close to speech synthesis, such as singing voice synthesis (Bonada and Serra 2007) or realistic synthesis of music instruments like the saxophone (Kersten et al. 2008), but it has also been applied more experimentally to texture synthesis (Schwarz 2004) .

Most Mosaicing and concatenative synthesis systems share a general architecture and implement a set of concepts. A significant number of systems have been reviewed and summarized in (Schwarz 2006, Schwarz 2005). What follows is a short summary of these concepts in order to analyze their utility for our own problem.

2.1.1 Sound Databases

The database contains a collection of samples that will be used to generate the sound. In many systems (Jehan 2005, Schwarz 2004, Casey 2005) the database is generated from a corpus of preexisting music compositions, often with some reference to Oswald's plunderphonics. In concatenative synthesis systems aimed at realistic synthesis of voice or instruments, databases are usually created with appropriate samples of these instruments. Databases can be labelled as *homogeneous* or *heterogeneous* (Schwarz 2006). In many cases, the database is specifically created for the application by recording or segmenting sounds contain homogenous units of similar size. In these cases, the size of the database typically determines the quality of the synthesis, since the probability that a sample will match a given specification is higher. There is however a compromise between perceived quality and flexibility. The more the synthesizer relies on models (e.g. spectral models of a given instrument), the smaller will be the required size of the database. Such models will often allow a finer degree of control over the synthesis process. Although the result of the synthesis may be less realistic than using systems purely based on samples, this kind of flexibility is not optional for applications like singing voice synthesis (Bonada and Serra 2007). Examples of heterogeneous databases are less common. Clearly, a database that has not been segmented to small units is less suited to algorithms based on a very specified target using descriptors, since these will vary along each segment. In (Casey 2005), a system based in acoustic lexemes is presented as a solution for indexing heterogeneous internet databases for music applications. Sounds are defined as sequences of 40 possible spectral archetypes. However, no high level interface is described to deal with this abstract vocabulary. In general, problems dealing with such databases may require more application specific solutions. For example in (Cano et al. 2004) a system is described that allows building ambiances

with environmental sounds representing real world concepts (e.g., animals, footsteps). Retrieval from a database of sound effects is based on a WordNet ontology.

2.1.2 Segmentation

Sound databases are composed of many (usually short) segments of sound. In mosaicing systems, these are often obtained from longer recordings, so a procedure for breaking them into segments is needed. Several segmentation algorithms are used. Algorithms often involve some kind of content analysis, so the process is tied to the extraction of features that will be used for retrieval. For example in the system proposed by Casey (Casey 2005), a Hidden Markov Model is used in the segmentaion phase to cluster all possible sounds into spectral archetype classes. Jehan (Jehan 2005) proposed that segmentation is based on automatic detection of note onsets, which seems to be a generally accepted approach. In the Caterpillar system by Schwarz (Schwarz 2007) segmentation is based on the original score of the music piece that is used as material for recomposition. Some systems also employ "fixed" segmentation schemes and cut the input sounds into short slices of equal duration (Sturm 2006). While manual segmentation is more related to the *Musique Concrète* and digital sampling traditions, few real world applications rely o automatic segmentation of long recordings. On the other hand, commercial concatenative synthesis products (e.g. Synful[1]) require manual segmentation for realistic voice or instrument synthesis. The *Freesound* database is a very heterogeneous one, containing at the time of this writing more than 50.000 sounds. Since the database is already used for sharing sounds for music creation, many users already segment the sounds in the ways their consider useful, like notes, chords and rhythmic loops. Our take on this issue is to respect this tradition and rely on the segmentation performed by users. However, we make use of onset detection for rhythmic similarity.

2.1.3 Audio Features

One crucial aspect of mosaicing and CSS is the descriptors that are extracted in order to index, search and retrieve sounds from the database. For music creation applications, most systems revolve around the traditional "music aspects": pitch, loudness and timbre. When longer segments are used, it is also common to analyze rhythm.

[1] http://www.synful.com/

We review some of the most common approaches to automatic content-based description.

Pitch

Pitch is usually defined as the perceptual correlate of the fundamental frequency. For monophonic signals, many algorithms have been developed for the estimation of the fundamental frequency, both in the time domain and the frequency domain. Time domain methods are usually faster and may be more adequate for sounds with temporal variations of pitch. For example in (Ricard and Herrera 2004), the autocorrelation function is used to determine both pitch and pitchness. The Zero Crossing Rate (ZCR), i.e., the speed at which the signal changes sign, is also used as a pitch descriptor in (Zils and Pachet 2001). One commonly used method based on the autocorrelation function is the Yin algorithm (de Cheveigné and Kawahara 2002). Also widely used is its spectral domain version YinFFT (Brossier 2006).

In the case of polyphonic music, the problem of tracking pitch for every instrument or sound source is still under intensive research in the literature. A kind of workaround, for certain problems, is to calculate the chroma feature (Jehan 2005), which consists in folding the entire frequency spectrum to a histogram of twelve pitch classes representing a single octave. One prominent example of chroma feature is the Harmonic Pitch Class Profile (Gómez 2006).

Loudness

Loudness is considered to be the perceptual counterpart of the sound wave amplitude. The amplitude envelope is the curve that describes its evolution over time. Usually, the problem in the context of indexing and retrieving sound segments is reducing this temporal information to scalar values, such as the attack and decay times, statistic moments (mean, standard deviation, temporal centroid, temporal kurtosis and temporal skewness), or coefficients of fitted polynomials. These may be called *characteristic values* (Schwarz 2007). Characteristic values are also needed in other situations where large vectors are involved.

Rhythm

Rhythm is usually related to some regularity in the evolution of some aspect of sound, usually loudness. Rhythm description is typically based on detection of on-

sets. While several methods exist for estimation of beat and tempo, a common basic descriptor of rhythm is the histogram of inter-onset intervals (Gouyon 2005). On the other hand, in (Jehan 2005), an autocorrelation function is applied to the outcome of the onset detector to represent rhythm.

Timbre

Timbre is by far the most complex concept of this traditional categorization, to the extent that its consistency is put in doubt by its negative definition (Donnadieu 2006) as the property that allows discriminating musical instruments *regardless* of pitch or loudness. As we have seen, part of the interest in working with samples has been the possibility of playing with timbre, for some in order to escape the closed timbre space defined by orchestra instruments and for others in order to have richer representations of them. Timbre is usually described in terms of the shape of the spectrum. A vast number of features are available as a result of extensive research in speech and music analysis. We summarize some of the most used ones.

Like in the case of the amplitude envelope, the spectrum envelope can also be described using scalar values by means of statistic moments . The spectral centroid, particularly, is traditionally considered to be linked to the perception of brightness in timbre. Spectral mean, standard deviation kurtosis and skewness are also indicators of the energy distribution in the spectrum. Spectral moments are routinely used for music classification tasks in MIR, and also in music creation oriented systems (Schwarz 2007, Coleman 2007). Since they are usually calculated from the Short Time Fourier Transform (STFT), one value is computed for each frame. Thus, temporal moments of the spectral moments are also computed to describe sound segments.

Mel Frequency Cepstrum Coefficients (MFCC) have been traditionally used in speech processing to decorrelate the excitation signal of the voice from its timbral characteristics imposed by the vocal tract. This strategy also works well with music instruments and in thus MFCCs are used as a descriptor of timbre in several mosaicing systems (Casey 2005, Lazier and Cook 2003). MFCCs are also widely used in MIRl as general descriptors for polyphonic music files. These coefficients are extracted from the short time spectrum of a signal by mapping the magnitudes to the Mel Scale using triangular overlapping windows. The Discrete Cosine Transform or the Inverse Fourier transform of the logarithms of the Mel-scaled coefficients are then computed depending on the variant.

Bark bands are also a discretization of the spectrum according to a psychoacoustical scale, the Bark scale, which represents the 24 perceptually critical sound frequency bands. In order to obtain a perceptual estimate of loudness, energy corresponding to each band is summed yelding one value per band. Bark bands are used in (Jehan 2005) as a descriptor of timbre. They have been showed to be effective for identifying the timbre of musical instruments (Herrera, Dehamel and Gouyon 2003).

2.1.4 Unit Selection

Unit Selection is the term inherited from concatenative speech synthesis for the algorithms that take care of finding the best sound segment for a given situation. In musical applications this means that a sample is selected either manually on the basis of its content descriptor, or automatically because of its similarity with a target specification. The target may be specified in a score (Schwarz 2007) or analyzed from a musical piece (Jehan 2005) . Audio input may also be used, so that the sound produced by a performer is used to control the system (Lazier and Cook 2003). Automatic unit selection may be performed by the path search unit selection algorithm imported from speech synthesis (Schwarz 2007) . Here, the Viterbi algorithm is used to find the sequence that minimizes the cost represented by several distances with the target specification. A generalization of this idea for music applications consist in mapping any desired property of the output as a constraint of the input units, so that the problem becomes one of constraint satisfaction (Zils and Pachet 2001).

2.1.5 User interface

Many corpus-based synthesis systems are designed to be used in real time performance, with the Mosievius system by Lazier and Cook (Lazier and Cook 2003) considered as the pioneer. This system included various GUIs, an interpreted scripting language and MIDI input as mechanisms for control. Systems based on manual selection typically require an interface to build dynamic queries with sliders or other controls (Coleman 2007). Also, some feedback of the sounds that are selected with a given configuration is sometimes provided, along with their position in the space determined by one or two features. This allows to define interfaces on the basis of a scatter plot, on which paths can be drawn (Schwarz 2007).

2.2 Network Music

The use of computer networks for musical practice has been extensively researched, either as an aid for existing music production techniques, or as a means to explore new possibilities for music creation. On the other hand, adoption to real world applications has been slowed down by the intrinsic limitation in speed and initially reduced bandwidth. Research in network music is not only related with the mechanics of sound production in the tradition of musical instruments, but involves the complex world of collaboration and social relationships. Perhaps related to this new complexity, research in this field has been very diverse, and developments have sometimes followed rather different (although connected) paths. With the boom of the internet, it has become clear that there will be demand for technologies that exploit its potential for music creation, and several attempts have been made to provide a unified theoretical base that explains most common strategies.

Since the specific affordance of the network is collaborative work, the same classification space that is used in the more general field of Computer Supported Cooperative Work (CSCW) may be employed to classify different approaches to networked music, as proposed by Barbosa (Barbosa 2003). Under this point of view, different projects are organized in one axis in relation to time, depending on wether interaction is synchronous (like for example in a live performance) or asynchronous (like in a collective composition process). The other axis in this space represents distance: some strategies imply *Co-located* interaction (like using local networks), while those involving longer distances can be considered *Remote*.

Weinberg, on the other hand, proposed a theoretical framework for interconnected musical networks (Weinberg 2005), where different approaches could be classified according to the level of interconnectivity among players and the role of the computer. Proposed classes are the Server, where several players just send data to the computer, the Bridge, where the computer is used to simulate proximity between remote players, the Shaper, where the computer provides musical materials for users to modify, and the Construction Kit, where participants may submit their creations as well as modify each other's.

Another overview is provided by Rohrhuber (Rohrhuber 2008). Here, the main aspects of network music are analyzed, although no attempt is made at a systematic classification. One important aspect is *Transmission*: many artistic projects have revolved between the fascination for telepresence and transparent communication on one side and the investigation of the medium and its opacity on the other. Not very

differently, communication protocols are often used transparently, but many works are based on the definition of protocols. Like programming languages, protocols define procedures that help defining *networks of relations*. In this sense, computer networks can be analyzed in the same way as computer programs, especially in the era of object oriented (message passing) and distributed programming. In such networks, the problem of sharing information appears. The two main approaches to informaiton sharing are concurrent access to *shared objects* or *distributed objects* .

We have seen several points of view of this generally irregular area that has been called *network music*. In order to explore the possibility of music production by casual internet users, we can narrow this field to approaches dealing with *remote* collaboration. Because of the different time zones and habits, most of the time asynchronous interaction must be taken into account in some way. On the other hand coincidence does also happen but because of the nature of network connections (intrinsic delay, variable bandwidth) stress is rarely put on real time performance like in physical collaboration. In order to explore the creation of new musical objects, we focus on the *Construction kit* kind of environment. This usually means dealing with *distributed objects*.

While for the case of mosaicing and CSS in most research efforts we found a general pattern in combining the same elements (database, segmentation, features), network music research is at this point focused on exploring different configurations and strategies. While common elements exist in implementations, we focus on the different solutions given to the problem of how to organize remote collaboration.

2.2.1 Strategies for web based collaboration

As the adoption of internet grows, especially through web based tools, its potential for music collaboration is becoming obvious. During this decade, several experiences have proposed a number of concepts and strategies.

Faust Music Online (Jordà and Wüst 2001) was a pioneer project which explored collective composition over internet in 2001. An emphasis was put on allowing casual internet users and *enforcing* collaboration. Music compositions were created using a custom browser plugin for the *wintel* platform, which allowed composition of sound textures by playing several layers using different generators and modulators. Thus, a particular, experimental type of synthesis was common to all compositions, and the project allowed unprepared visitors to become familiar with experimenting with electronic music. Compositions could be evolved from others, and a composi-

tion tree was used to store and browse the available compositions.

In Public Sound Objects (Barbosa 2005) the concept of a shared virtual space was used to allow collective creation of music in real time. In addition, the space had an actual physical dimension as an art installation. While the synthesis engine was based on FMOL, it was run in the server side, and the resulting sound was streamed to clients implemented as Java applets. Each user was presented with an interface representing a square box with a bouncing ball, whose collisions emitted sound. The timbre of the ball was produced by a sample, the metaphor being inspired on Schaeffer's sound object.

In DaisyPhone (Bryan-Kinns 2004), a music loop is shared among participants who collectively add and remove notes with a choice of four different timbres. The user interface is based on a radial sequencer, and the playback head is a rotating axis. Distance from the center of the circle represents pitch, color saturation volume, and note shapes represent timbres. Each user is assigned a color hue so each users contributions can be traced. Annotations can be made along the surface to communicate and interact with other users by drawing or writing with the mouse or stylus (if a tablet pc is used). The concepts of *Localization* (provided by graphical annotation of any parts of the shared work), *Mutual awareness* (supported by color hue), *Mutual modifiability* of other users' contributions, and *Shared and consistent representation* are prioritized in the design. Users have the possibility to join several sessions, in which one such loop plays while it's being collectively created. Synthesis is performed on the client side using a Java applet, and synchronization is loose, but sufficient to maintain the feeling of a shared experience.

The CODES project (Miletto et al. 2005) proposes the concept of *Music Prototype*, an analogy of music with software prototyping that aims at facilitating the creation of music for people with different musical backgrounds through collaboration. Music prototypes are composed of a number of layers, of which each user is allowed to control two. Layers consist in sequences of choices among predefined musical patterns of different styles extracted from MIDI files. Interaction is fully asynchronous. Three awareness mechanisms are presented to support community work: *Music prototyping rationale* defines the possibility of users to annotate and comment on their actions and decisions. *ActionLogging* provides a way to access the history of events that have lead to a particular state of a music prototype. *Modification marks* indicate, at an overview level when one user's work has been modified by other users. Consistently with the analogy with software prototyping, these concepts and their interfaces bear a resemblance with group work support in software development

tools, such as version management systems.

These experiences have showed the potential for internet based applications to foster collaboration either in real time or by asynchronously sharing musical artifacts. Although some of them use sound samples, in general synthesis engines are rather specialized, and they don't provide the level of timbral richness provided by a reasonably large corpus of samples. On the other hand, our purpose is to explore the potential of using sound samples as ready-made building blocks of musical compositions (more than as resources for a music synthesizer) in order to facilitate quick exchange of ideas to people regardless of their knowledge of traditional music concepts such as notes or patterns. Moreover, in most of these projects the server side is merely a hub that manages communication among participants and/or storage, but doesn't actively participate in the process.

PIWeCS (Whalley 2004) is a project that aimed at interactive publication of a corpus of sound recordings of traditional Maori instruments played by an expert performer. A multi-agent system (MAS) is devised as a way to assist composition to users without any experience. The system is based on a flash client that communicates through OSC with a MAX/MSP program running on the server. While few details are given about the actual interface, apparently a mixing-board style interface allows the user to switch on or off different combinations of the pre-recorded performance. More possibilities at a compositional level are offered through graphical menus which seem to require some level of knowledge. Thus, the weight of the musical expression seems to be put on the recordings, and the project adopts the *Shaper* approach in Weinberg's taxonomy. No information is given regarding any possible means for collaboration or interaction among users. However, the project poses interesting questions about the use of some intelligence in the server side. Evolutionary methods were apparently discarded and the MAS approach was chosen instead to aim at a conversational interaction between human and machine. However, such conversation seems to be defined in advance to put the user in the level of the helpless ignorant and the program as an expert advisor. We expect evolutionary models and facilities for social interaction to foster learning more as a process of discovery and collective construction.

CC-Remix and Malleable Mobile music (Tanaka et al. 2005) explore the possibilities for bridging between music reception and social creativity of CC licenses. The first project is web-based and allows a group to up to four users to engage in a collective remix session by choosing among previously segmented loops of a collection of songs licensed under CC. Loops chosen by each user are beat-synced and mixed

on the server. The software allows each participant to see what others are doing at any moment. In the second project, the possibilities of portable devices for social interaction are explored, along with their sensors as sources for music control. Users meet and form groups in chat rooms and choose an identity, related to instrumentation or timbres in the song, for collective mash-up. Their gestures and geographical location are interpreted by the device sensors to control a musical flow based on the song where each user controls a module of the synthesis engine. Both projects are based on the same framework, involving segmentation of a short collection of CC songs on the basis of statistical similarity of FFT vectors, and a Max/Msp synthesis engine that performs time stretching slice sampling and sample playback at variable rate. While the musical outcome of these projects is limited to very constrained mash-ups of popular music, they explore important concepts for building communities in the field of music creation. The authors enumerate *Shared goals* (which are inherent in music creation), *Reciprocity* provided by live interaction, *Engagement* as a measure of the challenge presented by the system, *Awareness* of users of their own actions, and *Belonging* to the shared activity. These concepts may prove useful as a framework to analyze community oriented software.

2.3 Evolutionary computation, music and collective knowledge building

2.3.1 Overview

Concepts inspired in natural evolution have a long history in computer science and have been applied in many kinds of problems. Genetic Algorithms (GA) were formulated in many different ways as a general technique for search and optimization problems. While many earlier formulations exist, there is a certain agreement on the foundational role of Holland (Holland 1975). Although many variations exist, most GAs are based in some type of encoding (typically a string of bits) that represents a chromosome over which several genetic operations may be applied, along with a fitness function that defines the characteristics of the desired solution. A sequence of steps simulating natural evolution is repeated until the solution is reached. A typical GA may be summarized as follows:

- Initialization. A population of potential solutions is generated (typically randomly)

- Selection. The individuals are evaluated according to the fitness function. Several selection algorithms exist for selecting a group of individuals with high fitness.

- Reproduction. Genetic operators, typically mutation (alteration of a gene in the chromosome) and crossover (derivation of a new chromosome from two parents) are applied to the selected individuals to generate a new population.

- Termination. The process is repeated until a certain termination condition is met.

Evolutionary models have also been applied to more creative domains like industrial design, visual art or music. In these domains, the fitness function often can't be automated, and Interactive Genetic Algorithms (IGA) are used, where fitness depends on human evaluation of a succession of candidate objects. An interesting example was presented recently that bears a certain similarity with network music projects: Picbreeder (Secretan and Beato 2008) is an online system that allows users to evolve pictures by selecting candidates generated by the algorithm. Users can start from scratch but they can also depart from published images by other users.

2.3.2 Evolutionary computer music

In the field of music, evolutionary methods have been used in a variety of ways. One classical example is the automatic generation of patches for FM synsthesizers (Horner et al. 1993). For the problem of imitating existing instruments, an automatic fitness function can be implemented in terms of similarity with a certain target sound. Another example where automatic fitness function may be used is modeling expressive performances (Ramirez and Hazan 2005).

For music creation applications, however, fitness generally is evaluated manually by composers. One classic and very documented system is GenJam (Biles 2003), an artificial saxophone player that has *evolved* since 1994. In explaining his work, Biles details the main problems that associated with genetic algorithms in the practice of music performance. An important one is the *fitness bottleneck* when fitness evaluation must be done in real time. Biles proposes the addition of musical constraints to the algorithm, so that his system can't play bad notes.

2.3.3 Evolutionary models and collective knowledge building

In 2001, Kosorukoff proposed a completely *social* implementation of genetic algorithms, the *Human Based Genetic Algorithm* (HBGA) (Kosorukoff 2001). HBGA departs from IGA and is presented as a workflow management aid, by involving users in all functions of the genetic algorithm. This process was demonstrated for collective knowledge generation in several projects, bearing a certain resemblance with the collective knowledge building site Yahoo Answers[2]. In fact, Kosorukoff claimed that many of today's tools for collective knowledge building and sharing, such as wikis or social news promotion[3] are related with evolutionary computation and HBGA. Thus, in a way, HBGA ca be considered more a perspective than an actual algorithm. Clearly, collective knowledge building is not something exclusively allowed by computers, the rules that govern it are usually studied by social sciences. What we have is the claim that there is a use at considering knowledge production under the point of view of biological evolution. This perspective was proposed by Richard Dawkins in *The selfish gene* (Dawkins 1976). The *Meme* is there proposed as the cultural equivalent of the Gene. Interestingly, though, the origin of the concept is explained in terms of music. Dawkins explains the experience of P.F. Jenkins analyzing the songs of a bird in New Zealand:

> During most of the time Jenkins was there, there was a fixed number of songs on the island, a kind of 'song pool' from which each young male drew his own small repertoire. But occasionally Jenkins was privileged to witness the 'invention' of a new song, which occurred by a mistake in the imitation of an old one.

A lot of people would be willing to accept this process of mistaken imitation as a definition of how music progresses. The so called *Memetic perspective* has since been adopted by those that could take some advantage of it. Regarding network music, in 2.2.1 we have reviewed many experiments dealing with the possibility of users creating musical objects and modifying other users' ones, which could be easy to interpret under the memetic perspective. HBGA, however, provides an interesting addition: the inclusion of artificial agents in the process. Artificial agents can undertake some of the tasks when users are not motivated enough to do them.

[2] http://answers.yahoo.com/
[3] http://digg.com

Chapter 3

Organizing free sounds

3.1 Introduction

One important challenge when dealing with a large database of sound samples finding methods for efficiently retrieving the desired sounds. We have seen that content based indexing has been applied to creative applications in mosaicing and concatenative synthesis. Yet several issues have not been solved. As we have reviewed these approaches usually rely on the existence of a previously specified composition (the target). In many applications, however, the target is not known. Thus, we find an important contrast between content based approaches used in mosaicing systems and the real world usage of databases like *Freesound*, where users describe the content themselves in order to find it. While manually describing and tagging sounds is tedious, online databases have showed the potential of doing it collectively. The problem, though, is that collective action brings a certain level of noise. The schaefferian metaphor of the attic illustrates our necessity to classify sounds:

> Should I measure the bird in order to put it among the tables? Should I put a deciliter of shavings among my jars? Physics don't help me, but to the contrary. If I'm advised to place the clothes by size, this won't allow me to sort it in relationship to the bird or the bottle. (Schaeffer 1966)

In *Freesound* we find a varied mix of audio cultures. There are field recorders who upload professionally recorded sounds from carefully chosen settings, field recorders who report on their environment at a more personal level, people interested in designing strange sounds, people who play traditional instruments, people who upload rhythm and synthesizer loops, people who exchange recordings of their voice. How could we possibly compare or find similarities among all those sounds?

In this chapter we describe two initial experiments towards better ways to organize sounds in a large and heterogenous database that were carried during the

development of *Freesound Radio*. The first one tries to obtain a conceptual map of the database from tags. The second one deals with the problem of using musical descriptors to define similarity when appropriate. Because its a large database built collectively by a number of users, we hope that the methodology and results should be applicable to other contexts. However, dealing with the whole database implies abandoning almost any assumption about the nature of the sounds, except for the built-in restriction that prevents users to upload complete music songs. Except music songs, we may find anything, from long speeches to tiny noise clips or tones. Thus, our attempt at classifying the sounds is necessarily coarse.

3.2 Background

In traditional instrumental music, all sounds can be classified according to the instrument that produces them. Within symbolic notation, all sounds to be produced are defined in therms of timbre (instrument), pitch (note) and loudness (note duration and dynamics notation). This organization is lost when trying to incorporate a wider range of sounds into the musical language. Especially in the case of music based on recordings, composers are faced with an initial step of choosing among a potentially large collection of discrete objects. Hence, the first step of the schaefferian program for experimental music was the *typologie*. In trying to avoid references to the musical tradition or to the sources of sound recordings at a semantic level, the *typologie* was strongly tied to the *morphologie*, forming a recursive relationship in which each one had to be defined iteratively with the help of the other. This resulted in a rather complicated system. As summarized by Chion (Chion 1983), the *typologie* resulted in thirty classes defined with the help of six pairs of criteria (*Masse/Facture*, *Durée/Variation*, *Equilibre/ Originalité*) A different perspective was suggested by Murray Schafer (Schafer 1977). While he also proposed a classification based on psycho-acoustical properties of sounds by extending the one in the *Traité*, his proposal was more directed to the analysis of all the sources of sound in the world. Thus, he devised a detailed taxonomy of sounds according to their sources.

With the large collections of sounds allowed by computers, the need of classifying sound files became general. The issue of describing and classifying sounds from a general perspective was addressed in MPEG-7 standard (Casey 2001). In this new context, both the tradition of classifying sounds according to their morphology (Ricard and Herrera 2004) and according to their semantic reference (Cano

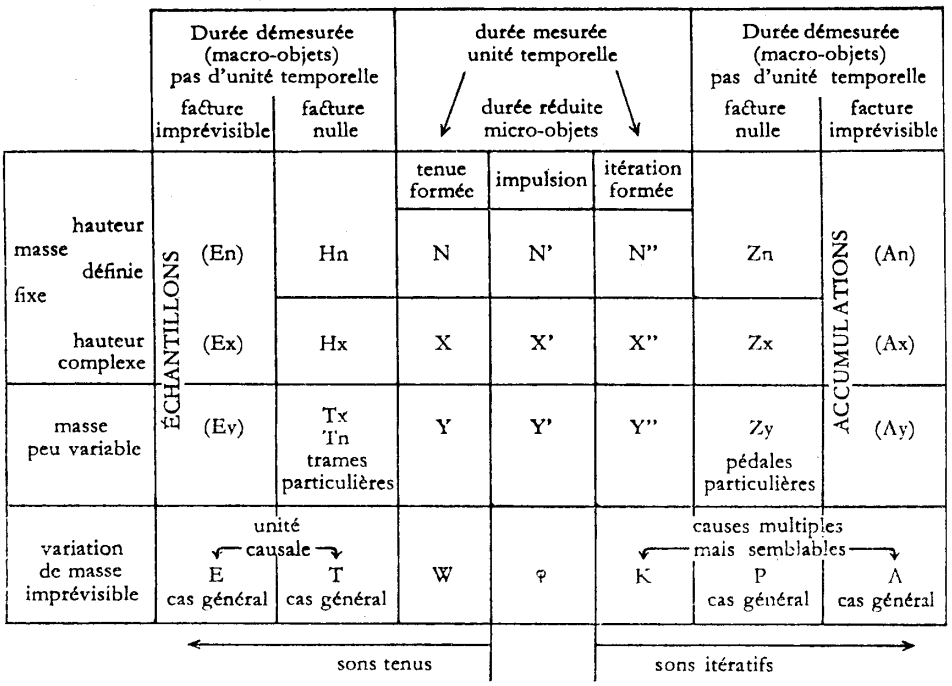


Figure 3.1: Schaeffer's typologie
(Schaeffer 1966)

2007) have been pursued, along with classification focused at musical instruments (Herrera, Peeters and Dubnov 2003).

The *Freesound* database allows us to experiment with both approaches. Users describe sounds at a semantic level using tags. These tags are not always descriptive of the sound sources, but they represent what the sounds refer to users. For example the instruments used to produce an electronic loop may not be relevant. On the other hand, automatic content descriptors make it possible to create morphological taxonomies that are independent of the source of the sounds. We explore both approaches on one hand by analyzing the tag folksonomy and on the other by devising a taxonomy based on content descriptors. In order to avoid an arbitrary classification based on subjective criteria, we use some specific tags as the ground truth to define the classes of the content-based taxonomy.

3.3 Database and tools

Four our experiments towards a coarse classification of the sounds in *Freesound*, we depart from a copy of the database containing 44790 sounds and 13108 tags. We use an in-house implementation of some common low level descriptors of pitch, timbre, loudness and rhythm, together with a decision tree classifier.

3.3.1 Descriptors

As summarized in 2.1.3, in the context of music creation we can classify sound descriptors in four groups: Pitch, Timbre, Loudness and Rhythm. In general, we use characteristic values (mean, variance, minimum and maximum) of descriptors for each segment.

- **Pitch**. In order to represent pitch, we use both the pitch estimate and the confidence values from the YinFFT (Brossier 2006), as well as HPCP histograms (Gómez 2006)
- **Timbre**. Timbre features include a vector of several spectral shape measures (centroid, crest, decrease, flux, kurtosis) (Peeters 2005, Tzanetakis and Cook 1999), as well as MFCC coefficients (Rabiner and Juang 1993) and bark features (Herrera, Dehamel and Gouyon 2003)
- **Loudness**. The amplitude envelope is described using log-attack time, temporal centroid, decrease and kurtosis (Peeters 2005, Schwarz 2004) as well as the number of detected onsets(Brossier 2006),
- **Rhythm**. Rhythm is described using the relative inter-onset interval histogram, along with position and value of the three largest peaks estimated from it and the rate of onsets per second (Gouyon 2005).

3.3.2 Decision trees

Decision tree learning (Mitchell 1997) is a very popular family of data mining algorithms, which are able to extract a predictive model, in the form of a decision tree, from a series of observations. The advantage of decision trees, especially of small ones, is that they are intelligible and allow an understanding of which features in the data lead to its assignment to a given class. In our case, we are dealing with organizing a very heterogeneous database in a way that is useful for music creativity.

Thus, we don't depart from a predefined, clear cut classification scheme, we are in the middle between the free tagging activity of users and the concepts implied in content descriptors. Therefore we are interested in keeping intelligible models that can be iterated and refined in a dialogue between both. In our experiments, we use the $C4.5$ decision tree algorithm available in Weka [1] (Witten and Frank 2000).

3.4 Conceptual taxonomy

Tag folksonomies have become an interesting problem in recent years. On one hand, they provide valuable information about the communities that use them and the objects that they tag; on the other, they are noisy and unstructured, which makes it difficult to obtain a general view. Since folksonomies seem to operate on a certain level of agreement among users, in some cases the assumption of a latent taxonomy can be made. In (Heymann and Garcia-Molina 2006) a simple algorithm is presented that allows to extract a hierarchical taxonomy from folksonomies. This algorithm requires a similarity function to be defined between tags, such as cosine similarity in the vector model (Baeza-Yates and Ribeiro-Neto 1999). We applied this algorithm to the *Freesound* in order to obtain a conceptual map of the database. The process may be summarized in the following steps.

- The folksonomy can then be represented in a matrix F where $f_{i,j} = 1$ if document i is annotated with tag j and 0 otherwise. Rows in this matrix are sound files, and columns are tags.
- Given two tags, t_1, t_2 we define their similarity using the cosine of their vectors: $sim(t_1, t_2) = \frac{\vec{t_1} \cdot \vec{t_2}}{|\vec{t_1}| \times |\vec{t_2}|}$. This can be implemented by defining a matrix M that has the inverses of modules of each tag at the diagonal, $m_{i,i} = \frac{1}{|\vec{t_i}|}$. Then the distance matrix D can be computed as $D = ((F^T \times F) \times M) \times M^T$
- We obtain a centrality score for each tag using the principal eigenvector of D
- The taxonomy extraction algorithm (Heymann and Garcia-Molina 2006) can then be applied in the order of centrality: we iterate over tags starting from the most central one and adding them to the most similar tag already in the hierarchy if the similarity is above the threshold, or to the root if no suitable tag exists in the taxonomy.

[1] http://www.cs.waikato.ac.nz/ ml/weka/

The threshold parameter determines the number of concepts that will appear on top of the taxonomy. A small threshold allows tags to be added below tags that are less similar instead of to the top level. A higher threshold will increase the number of tags in the top level. We used a value of 0.15 to obtain a suitable set of top level concepts that are representative of the sounds in the database. In addition, we require tags with a frequency greater than 2000 in order to go to the top of the hierarchy. This allows us to obtain a general overview of the content in the *Freesound* website:

- *Field Recording* describes the use of the site to document the acoustic environment. Tags under this category usually indicate real world concepts, and do not hint towards potential uses of the recordings. These potential uses include video or multimedia productions where real world concepts must be represented, or music based on soundscapes.

- *Electronic* includes sounds of synths, loops and effects, but also acoustic sounds, along with sounds of electronic devices. Most music related sounds are connected, in one way or another, to the *electronic* tag, which is the most central. Thus, one possible interpretation is that this category is focused on the *electronic* word more in the sense that musical sounds in the database are intended to create electronic music.

- *Noise* includes all kinds of sounds that users find interesting because they are weird, like glitches or sounds of machines. this category may include field recordings, but in general it seems also more oriented towards electronic music and sound design.

- *Voice* is the category for recordings of speeches, words, phonemes. In this category we find often sounds that are uploaded per request containing specific words in different languages, along with more casual recordings. Thus, this category is similar to *field-recording* in that sounds can be useful for media productions that require specific utterances.

An additional step required to use this taxonomy to analyze the database is to define a set of general concepts and assign the samples to them. One problem here is that since samples have several tags, there is no apparent reason to assign a sound uniquely to one class, although this may be needed for some applications. In order to classify all sounds according to the top level classes, we just locate their most

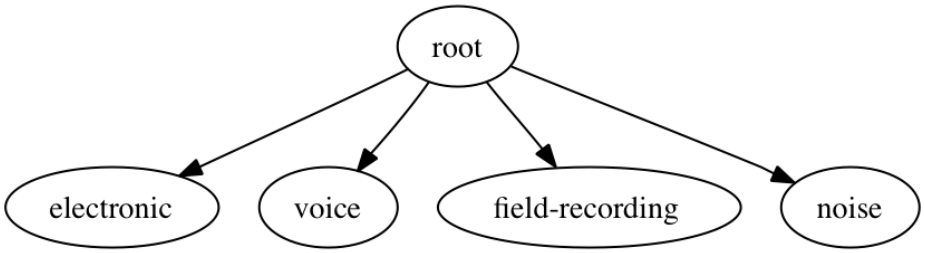


Figure 3.2: Top level concepts

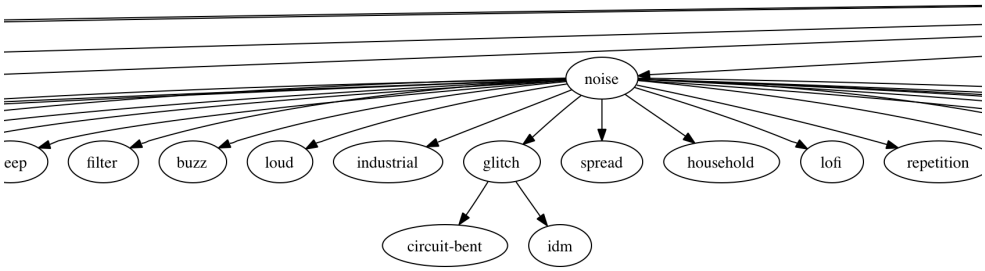


Figure 3.3: Concept taxonomy detail

general tag in the taxonomy. In case of ties, we keep going down and examine less general tags to break the ties. A general problem of this way of classifying sounds is that some sounds are annotated only with tags that are never connected to other tags (i.e. they never appear together with other tags). These rest unassigned. Table 3.1 shows the number of assigned sounds to each category along with concepts at the second level of the taxonomy. About half of the database falls under the *electronic* category. This shows that despite the (perhaps surprising) growth of the field recording culture in the site, electronic music creation is a prominent application of the database.

Initial experiments with classification based on low level descriptors shown that the top categories obtained with this system are not very consistent with respect to

concept	Instances	subconcepts
electronic	23926	scanner whine harmonic radio soundeffect generative fear analog police mallet whoosh hardpcm synth destroy electro war rhythmic tones power gadget stutter game stab fm cable alarm oscillator waldorf processed circuit loop
noise	9476	layer industrial feedback burst rattle scratch lo-fi pop glitch frequency static bend meditation repetition micro electricity screaming mic spread pen wood sound-design digital white household electric bleep crack effect experiment lofi sine data white-noise noisy raw signal metal hum filter record paper buzz reverb loud silence
field-recording	7077	summer walking outdoor office insects sound-fx london earth birds announcement tweet environment south-spain whistle traffic background bat morning sonar spring village ambiance thunder-storm street sea dawn canada thunder diesel children bird explosion nature water frequency-sweep temple wildlife purist foot-steps conversation footsteps road cracking ultrasound church turkey city jet engine rainfall fire crackle park train tapping town christmas streetnoise car inside tree morocco aeroplane robin binaural india voices brake ship airplane winter machine night-time forest animal birdsong high-pitch rain insect moving marshes building stereo natural dog block wind
voice	3611	human echo cell-phone female singing talking funny ringtone speech freesound woman mouth baby word morph request robot vocal recording laughing english male

Table 3.1: Subterms and instance assignment of top level concepts

low level descriptors. However, they offer a good characterization of the database from a semantic point of view. While the obvious application of concept taxonomies is the navigation of the site, in this thesis we will use this set of classes in order to analyze the use of samples in collective music making in the sense explained in 1.4.

As an initial step towards the use of low level descriptors, we devise another taxonomy specific to music creation in a sense that is present both in descriptors and in the database, using specific tags as the ground truth for the decision tree.

3.5 Music oriented taxonomy

While a taxonomy based in classic western music concepts may impose a certain stylistic bias to the system, these concepts involve already a significant part of the database, following the tradition of sample libraries that offer instrument notes and rhythmic loops. Thus, it makes sense to use available description technologies to

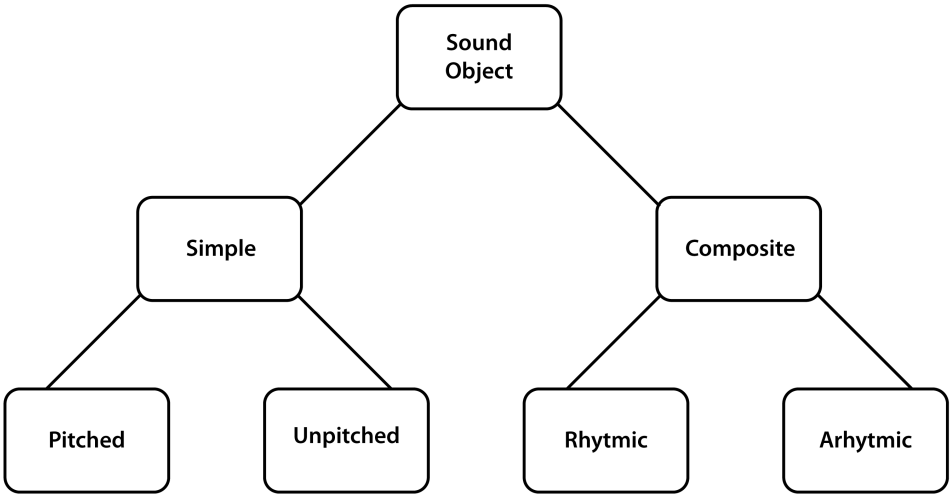


Figure 3.4: Music oriented taxonomy

index files that do have tempos and pitches. One point of departure is offered by the definition of sound object as an *energetical event* in the *Traité des objets musicaux* (Schaeffer 1966). One agreement with traditional music concepts represented in content descriptors may be based on the identification of an event and an onset. Thus, we may use an onset detector to distinguish among *simple* and *composite* sound objects. For simple sound objects we can provide a distinction between pitched and unpitched objects, and for composite objects one between rhythmic and arrhythmic fragments.

This simple distinctions represent an initial step towards mapping existing content descriptors to a real world database. Further steps and refinements may proceed iteratively so that the actual categorization can be driven by users themselves. In this sense, we rely on tags assigned by users to define the partitions. Tags are used as annotated ground truth data to drive a decision tree classifier. We expect that this way of proceeding should be easy to extend to any tags that provide some consistency with descriptors.

We trained the decision tree classifier with pairs of tags representing the concepts we wanted to split. For the pitched / unpitched distinction, we used the *noise* versus *note*. For the rhythmic / arhythmic distinction, *drumloop* against *rain* provided good

results (rain drops may be considered a classic example of random). Parameters were set in order to obtain simple trees that provide some understanding of the relationships between descriptors and tags.

The resulting tree for the note versus noise model is summarized in listing 1. Here, the algorithm found a threshold in the spectral flux descriptor to discriminate between notes and noises. Spectral flux (Tzanetakis and Cook 1999) is a measure of the speed at which the spectrum changes. This doesn't seem to say much about pitch directly, but it seems to make sense to discriminate sounds that have a certain stability in the spectrum. Finding suitable tags to represent this distinction turned to be more problematic than expected. Thus, the number of note examples is low, and the recall measure shows some room for improvement in cross-validation results.

Summarized results for the drumloop vs. rain tree can be seen in listing listing 2. The tree is also rather simple. Here, the emphasis was put on rhythmic features, including the two greatest peaks in the inter-onset interval histogram. The second peak of the inter-onset interval histogram was chosen along with the onset rate. Thus, the second peak seems to be a stronger indicator of rhythm than the first one, suggesting that non-rhythmic sounds may have a high value for one peak by chance. In addition to a higher second peak, the model selected a threshold for the number of onsets per second for rhythms. This may be biased towards the kind of rhythms represented by drum loops and will probably rule out very slow rhythms. However it makes sense to put a minimum speed for determining rhythm. Application of this model to the rest of the database provided interesting results as a way to detect rhythm in sounds of different sources.

Class	**Instances**
Simple Pitched	1001
Simple Unpitched	8390
Composite rhythmic	13950
Composite arhythmic	21449

Table 3.2: Number of sounds per morphological class

3.6 Similarity

The music oriented taxonomy defined in the previous section allows us to tell apart sounds for which it makes sense to compute rhythm features from those where it

Listing 1 Model and cross-validation results for simple object noise/note

```
J48 pruned tree
------------------

spectral_flux <= 0.022925: note (97.0/6.0)
spectral_flux > 0.022925: noise (601.0/26.0)

Number of Leaves  :  2

Size of the tree :  3


=== Detailed Accuracy By Class ===

TP Rate   FP Rate   Precision   Recall  F-Measure   Class
  0.99      0.265      0.949     0.99      0.969    noise
  0.735     0.01       0.935     0.735     0.823    note

=== Confusion Matrix ===

   a   b   <-- classified as
 575   6 |   a = noise
  31  86 |   b = note
```

doesn't. Regarding pitch, since we have made the distinction regarding the spectral flux, we assume that sounds with a varying pitch may be classified as *unpitched*. This is not bad regarding musical uses of both classes (e.g. sounds with a stable pitch are typical in traditional use of samplers to imitate musical instruments, while short sounds with a varying pitch or no pitch at all are both common in rhythm composition) but in order to preserve perceived similarity we don't remove pitch features. We do remove pitch for composite objects and rely on hpcp histograms. Introduc-

Listing 2 Model and cross-validation results for drumloop/rain

```
JJ48 pruned tree
------------------

peak2_val <= 0.055556: rain (398.0/17.0)
peak2_val > 0.055556
|   rhythm_onset_rate <= 1.395535: rain (32.0/6.0)
|   rhythm_onset_rate > 1.395535: drumloop (571.0/14.0)

Number of Leaves  :  3

Size of the tree :  5


=== Detailed Accuracy By Class ===

TP Rate   FP Rate   Precision   Recall  F-Measure   Class
  0.947     0.043      0.968     0.947     0.957    drumloop
  0.957     0.053      0.929     0.957     0.943    rain

=== Confusion Matrix ===

   a   b   <-- classified as
 549  31 |   a = drumloop
  18 403 |   b = rain
```

ing partitions in the database helps reducing the cost of computing distances. Since we are interested in distances that are meaningful in a musical context, this allows us to use relatively large feature sets that are focused to the musical properties of sound objects. In order to obtain similarities, we normalize features locally inside the class and compute the euclidean distance, i.e: $D_E = \sqrt{\sum_{i=1}^{N} (x_i - y_i)^2}$ gives the distance between sound x and sound y described by N features. Given this distance, we compute a table of the 10 nearest neighbors of each sound in its class in order

to substitute sounds in genetic mutation. The table is computed using a python wrapper [2] to the ANN library [3]

Simple pitched	pitch
	envelope
	hpcp
	mfcc
	barkbands
	spectral shape
Simple unpitched	pitch
	envelope
	hpcp
	mfcc
	barkbands
	spectral shape
Composite rhythmic	envelope
	mfcc
	barkbands
	hpcp
	ioi
	ioi peaks
	spectral shape
Composite arrhytmic	envelope
	mfcc
	barkbands
	hpcp
	spectral shape

Table 3.3: Selection of features for class local distances

[2]http://scipy.org/scipy/scikits/wiki/AnnWrapper
[3]http://www.cs.umd.edu/ mount/ANN/

Chapter 4

Patching samples

4.1 Motivation

The potential of the web as a platform for collaborative music creation was investigated in various projects reviewed in 2.2. In general, interaction on the web is asynchronous, both because of the latency imposed by the medium, and because of the different usage patterns and time zones of the users. Moreover, the web poses the challenge of engaging casual, anonymous users whose background is not known. Many of those projects had to use some way of organizing data to express a musical composition. Ideally, such compositions should be accessible to users with different levels of expertise, and be adapted to the usage pattern of the web: they should allow to quickly share ideas and receive feedback; because the limited interaction possibilities available on browser based technologies, they should be easy to create. In other words, the point is not replicating the available tools, but to exploit the potential for collaboration and interaction with other users.

A good definition of the problem was provided by the *Musical Prototyping* (Miletto et al. 2005) concept, in the sense that in the context of joint work, the musical creation should not be deemed as a finished piece of work, but rather as something built progressively. Hence, the metaphor of software prototyping. One problem with the approach proposed in this paper, however is the utilization of traditional form based interfaces that are used for business. In this sense, the implementation of the software prototyping analogy puts the user in a situation of considering music creation at the same level of a paid job: a boring task that needs to be done efficiently and in the less time possible. This is clearly not the way in which most casual internet users approach music creation. While some may be music professionals, in most cases music is an activity developed without a clear connection with an economic gain, and is not carried by accomplishing a set of predefined tasks that deterministically lead to a goal.

One interesting explanation regarding the distinction between working and playing, and its relation to the arts was provided by the work of Hans-Georg Gadamer. In *The Relevance of the Beautiful* (Gadamer 1977), Gadamer stressed the difference between two different experiences of time. *Work time* is related to the traditional spatial conception of time. Under the point of view of work, time is like a line that can be divided, an empty container to be filled with events. In contrast, *play time* does not exist with independence of events. When playing, time is conceived as a property of the activity itself.

Most software audio sequencers are based on time lines that allow the visualization of the whole work, with independence of the moment that is being played. This interface has become the *de facto* standard for music production, especially for all the phases in the process that can be easily considered *work*, and carried out by a single person in control. For obvious reasons, audio sequencers of this kind have not found great acceptance in live performance situations, or for collective creation. Some programs, such as Ableton Live [1], became very popular by allowing a more real-time approach to audio sequencing. The simple excel-style interface offered by Live still inherits the limitation of the mixing desk metaphor. Sounds are no longer necessarily forced into a time grid, but they are still forced into channels. With computers, especially with nowadays computers capable of playing hundreds of tracks, the mixing desk metaphor is a limitation that is not needed when dealing with sound samples.

We propose the concept of a sample patch as a way to sequence sound objects that is not based on a time grid neither on a mixing desk, but makes it possible to quickly create short musical compositions on top of a large database of sounds. Patching interfaces have been very common in computer music since the beginning. The Music V paradigm allowed to define music programs as signal flow graphs. With the popularization of Max many musicians without a background in computer programming were able to create such programs by using graphical patchers. The patching interface has since become widespread and implemented in a wide variety of programs. Incidentally, graphical patchers were also popular during some time as code generators in some software development environments, so an analogy with software prototyping is mantained. Because of many usability problems associated with visual patching as a way to program, text based interfaces for music have become increasingly popular, especially as computer users become less scared at text based programming, and live coding has emerged as a way to use such interfaces

[1] http://www.ableton.com

in live performances. In a somehow opposite direction, visual patching has been brought to the tradition of musical instruments. The Reactable (Jordà et al. 2007) implemented *dynamic patching* (Kaltenbrunner et al. 2004) with a tangible interface. The massive success of the Reactable outside of the world of research has showed the general acceptance of the patching interface as a way to allow the end user to access infinite recombination possibilities.

4.2 Related work

Besides modeling networks of signal flow, graphs have been used in computer music mainly to model networks of transition probabilities. Less common is the usage of graphs as deterministic sequencers, and even less as sound object sequencers. The first iteration of the sample patcher was developed by the author in collaboration with Anna Xambó as a project for the real-time interaction course taught by Sergi Jordà and Günter Geiger at UPF. Initially, we weren't aware of any graph based sample sequencer. However, with the generalization of the use sound files as building blocks for music composition and the popularization of patching interfaces, the combination of both has become a somehow obvious step to create sample sequencers that escape the mixing desk metaphor

Graph theory (Freeman 2008) explores the possibilities of graphs to allow the audience of a concert to define the actual score of a musical piece by voting their favorite path through a web site. While the aim of the project is to generate a score for a classical music concert, the interface is based on samples to allow users to have an idea of how the result will sound like. Thus, samples are limited to the instrumentation defined for the pieces, and user activity is reduced to selecting among a range of previously defined paths.

Mash! [2] is an online java applet aimed to allow users to make collaborative mash-ups by cutting loops from music songs and remixing them using a graph-like interface. Although the implementation is still in a very rough alpha state, the interface allows to load some loops and make connections. Sequencing is based on a pre-established tempo, and thus the collection of available loops seems to be limited by their accordance to the tempo. Creating multiple paths or having several entry points doesn't seem to be possible. Loops are not represented by transitions, since all objects are loops and their duration is defined as the number of cycles.

[2] http://www.sonasphere.com/mash/

Geography (Valle 2008) is a graph based sequencer of sound objects built using the SuperCollider language that aims at bridging between the algorithmic composition and the live performance traditions by supporting both graphical patching and text based interaction. Sound objects are represented by vertices in a graph, and edges are assigned durations. Thus, multiple edges are possible between two given vertices, in addition to loops. Music sequences are played by actants that navigate the graph randomly according to a certain distribution. This creates a rather complex sequencing environment to be used presumably by a single experienced composer / performer.

4.3 The Sample Patch

With the goal of providing a simple but flexible data structure to express music ideas as organizations of sounds in a database, we propose the sample patch. A sample patch is defined as a directed graph where vertices represent sounds from the database and edges represent transitions between them. Transitions are triggered when the sound of their source vertex finishes playing, and immediately start playback of the target vertex sound. Cycles are allowed, which results in patches of infinite duration. In particular, one vertex can be connected to itself to play it as a loop. Each vertex can have several incoming and outgoing edges, but only one edge in each direction is possible between two given vertices. Vertices are always followed and thus patches are deterministic.

The idea is that this model is equivalent to the classic "brickwall" model that is used by conventional audio sequencers, but silence must be supplied by silent samples from the database. The main difference is that instead of focusing on the timeline, the interface is focused on the sounds and transitions among them, and durations and time structures are determined by the available samples. This restriction is obviously a major one. Our hypothesis is that it will force users to abandon any intention to do "serious work" and allow a more playful attitude that is focused on discovering new sounds.

In order to provide the possibility of several entry points (i.e. not forcing all patches to start with one sound), a root node with no sound neither duration is defined. From the root, as well as from any other vertex, multiple outgoing edges will correspond to multiple audio tracks. The situation is more complex if two paths converge to a single target. This means that one transition may be fired when the

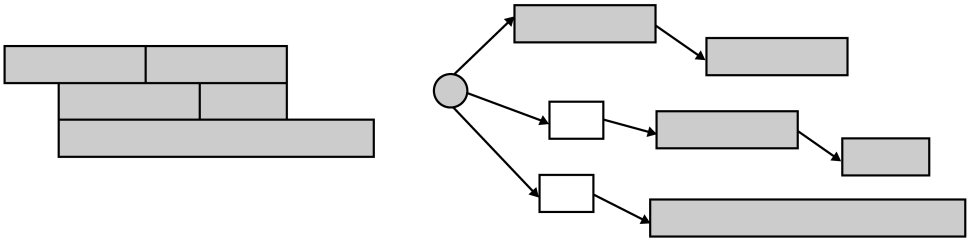


Figure 4.1: Equivalence of sample patch to traditional sequencer representation

target vertex is already playing. In such situation, the second transition will be simply ignored. The reason is that if we chose to restart the target vertex each time one of the sources ends, the target vertex may never reach to its own end (e.g. if one of the sources is in a loop that is shorter than the target). Trying to fix this brings even more complications that move us further away from the behavior the user would expect. The option of always allowing the sound to reach its end is less suited for repetitive rhythmic patterns, unless the duration of sounds involved in loops is chosen carefully, but it allows more complex structures. Once more, we resort to the database to provide the material that determines the rhythmic structure.

4.4 Implementation

The sample patch was designed to allow users to create and share simple musical constructs based on a large collaborative database of sounds. Although nothing prevents its use as a compositional tool in other contexts, this application was evaluated by implementing prototype to create sample patches in the ActionScript language on the Adobe Flex[3] platform on top of the *Freesound* datbase. The implementation is designed to work in real time so that the model can be played at the same time it is being modified.

4.4.1 Sequencer

Real time audio applications typically employ either a callback mechanism or a blocking mechanism to feed the audio output with the necessary amount of sam-

[3]http://opensource.adobe.com/flex

ples at a given rate. One important concept is the control block, which defines the speed at which user actions are polled and decisions are made to influence the signal processing mechanism. In our case, while the sequencer lives inside the Flash player runtime and has no direct access to the sound card, one such callback mechanism can be hooked to the completion event of a very short sound. We use the popforge library [4] to provide this callback . However, since no synthesis is required, and the Flash player already provides sample playback, no other feature of this library is used.

The sequencer maintains a list of events where the time remaining to finish each of the currently playing sounds is stored. Each callback, the buffer time is removed from each event. If the event falls inside this buffer, all children of the ending vertex are fired. Thus a new list of events is generated for the current callback and latter added to the existing list. A stage of removal of duplicate events is performed to avoid multiple parallel paths among the same vertices.

4.4.2 Interface

The interface is based on a graph visualization provided by the flare [5] library. Interaction is implemented to allow the user to modify the underlying graph. Custom renderers were implemented for sounds to display a summary of the waveform (provided by *Freesound*). Also, edge renderers are customized to allow curves (only for nodes pointing to themselves) and arrows. Users can drag samples from the database to the playground area, connect nodes by clicking once on each, and disconnect them by clicking again in the same sequence. A more detailed view of each sound is provided in an auxiliary display which displays the author of the sound and its duration, along with a larger waveform.

[4] http://code.google.com/p/popforge/
[5] http://flare.prefuse.org/

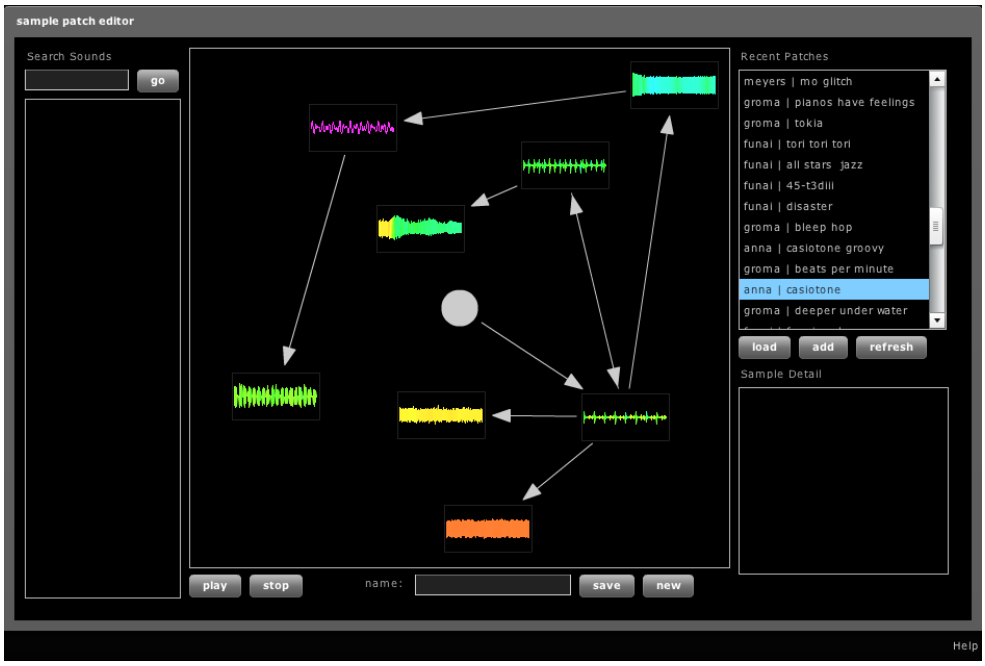


Figure 4.2: Sample patch editor interface

Chapter 5

Evolution

5.1 Introduction

The proposal of an evolutionary interpretation of collective knowledge building was briefly reviewed in section 2.3. We have also acknowledged the existence of a strong tradition in computer music of using evolutionary algorithms. By implementing an evolutionary model as a workflow for users to produce knowledge, we may consider both human and artificial agents performing the genetic operators. The advantage of this approach is that it takes into account the potential asymmetry in the tasks realized by users voluntarily. In the *Freesound* website, the number of people who actually contribute content is tiny compared to the number of people who merely download content. This asymmetric model of producers and spectators has been inherited from non interactive media distribution, and is reflected for instance in the different upload / download capacities of asymmetric DSL lines. Hence, we propose a framework that can produce an infinite amount of combinations of samples for consumption, but at the same time allows users to participate in the process, and eventually produce all of the content. The algorithm is based on the sample patch data structure presented in chapter 4. It evolves sample patches by recombining and mutating existing ones. We expect user submitted content to be generally more interesting than automatically generated content. However, by performing crossover on user-submitted patches, we implement a sort of automated collaboration that in most cases wouldn't happen spontaneously. Given the limited interaction possibilities of the web, this method may become a useful way to speed up collaboration between users without even knowing each other.

5.2 Initialization

Typically, in genetic algorithms with automatic fitness function, initialization may be random. Random initialization has also been used in many creative applications of genetic algorithms . In these cases, the algorithm is regarded as a random search, and from the creative point of view they are regarded as ways to obtain new objects that wouldn't be imagined by humans. Clearly, random initialization puts a burden on the fitness function, which in our case relies on subjective judgement of the listeners. In a context where users are not expected to have any further goal than listening to the music and specifying their preferences, random initialization is not affordable, as confirmed by early experiments with the genetic algorithm. Thus, initialization is performed by users with the sample patch editor described in the previous chapter. For an infinite process, like a radio providing a continuous soundscape, this initialization is only needed once. However, as more patches are provided by users, they are involved in the reproduction process, which results in more interesting combinations and a greater population diversity.

5.3 Selection

Several methods exist in genetic algorithms to select among the best individuals in the population, which in our case is determined by collective rating. Tournament selection (Miller et al. 1995) is one common method that preserves some of the population diversity. We implement the deterministic variant where the best individual is chosen from a random sample of the population whose size is adjustable through a *selection pressure* parameter. This process is repeated until the desired amount of parents is obtained. We then perform crossover with a certain probability to obtain new patches. Both original and newly generated patches have also a probability of mutation.

5.4 Crossover

Graph crossover algorithms have been developed mainly for evolution of drug molecules and circuits. It usually requires splitting graphs in two halves and connecting spare pieces. We implemented a simplified version of the algorithm described in (Al Globus and Wipke 2001). The main complication is introduced by

cycles, which in the case of sample patches are important. A random edge is chosen for breaking up the patch. Then, if the edge is part of a cycle, other paths may remain that connect both nodes; for each of these, a random edge is removed until no paths exist between them. Since one of the graphs ends the process without the root node, we connect a new root to the best candidate identified as the node that allows access to most nodes. This allows us to connect two graphs by simply replacing the root node in one with a random node in the other. A quality control stage is introduced to ensure that all nodes are accessible and that the number of nodes and edges remains within limits. This method can be improved by imposing some musical constraints to the breakup and merge points. The problem is obviously who is in charge of deciding these constraints. We store the counts of transitions between samples of the different classes in user submitted patches in order to maintain a musical tradition that can bias this crossover process.

5.5 Mutation

Mutation operations that can be applied to graphs include adding or removing nodes and edges or replacing nodes. Substitution of nodes is consistent with the analogy of music to natural language: in the *Traité*, Schaeffer reviewed the linguistics of the time and studied the applicability to *Musique concrète* of the rules of *selection* and *combination* of basic units into more complex units of meaning. Selection implies that term may be replaced by another of similar meaning. For example, the note of an instrument can be replaced by the recording of the same note form another instrument (Schaeffer 1966). We can implement this philosophy thanks to the class local similarities we have defined in chapter 3. Mutation is performed by substitution of a sample by the most similar sample in its class.

5.6 Implementation

The algorithm was implemented in *Freesound Radio* in a server that continuously generates playlists in real time. A corresponding flash client plays them using the sample patch sequencer described in chapter 4. The playlist is a population of patches that may be rated by users while they hear them. When a playlist is complete a new one is formed by incorporating recent user submitted patches and generating new ones by mutation and crossover from the previous generation.

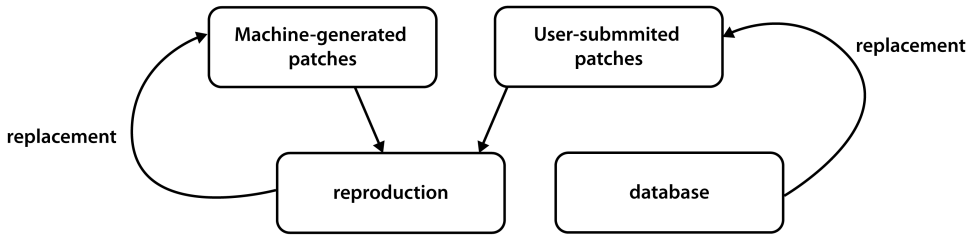


Figure 5.1: Playlist management

The player interface includes buttons for rating different patches and allows bookmarking samples and tags. Bookmarking a sample decreases its probability of disappearing from a patch by mutation, while bookmarking both tags and samples biases choices in of suitable replacement samples from the nearest neighbors table. This allows users to influence the evolution of patches along with ratings.

A chat is also provided to support mutual awareness. In (Shamma et al. 2007) chats are implemented on the basis of psychological studies regarding how humans enjoy more media if they can share the experience. Acousmatic concerts are an obvious example in music. In (Tanaka et al. 2005), chats are used to allow users to form groups. In order to allow users to coordinate strategies, user votes are displayed in the chat window.

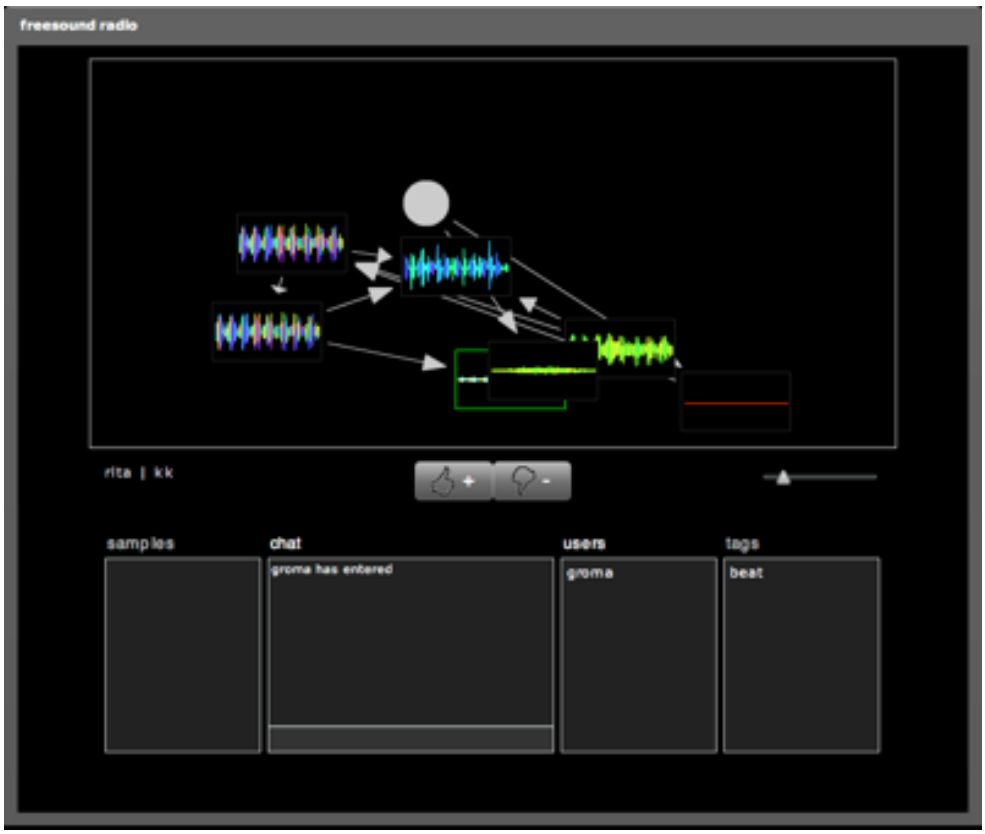


Figure 5.2: Player interface

Chapter 6

Analysis and discussion

6.1 Content usage

The *Freesound Radio* prototypes were put online initially internally in our research lab, and later on on the *Freesound* website open to general public. We analyzed 70 patches submitted by 20 distinct users during this initial period, using the general classes outlined in chapter 2 to obtain a rough overview of what samples are used in this kind of application. Regarding conceptual classes, the prevalence of the electronic class was predictable, since most musical concepts fell under the electronic category in the taxonomy. The role of noise is less obvious. This category is related with all people who use *Freesound* to exchange glitches and weird sounds they find interesting. We assume that the radio allows the continuation of this activity at a more compositional level.

The use of sounds regarding to the coarse morphological categories in the music oriented taxonomy was biased towards rhythmic material, which was also predictable. Less rhythmic composite fragments were also very common. In a way, graph based sequencing allows more experimentation with this kind of material that wouldn't fit the time grids of traditional sequencers. Simple sounds also find some use, but clearly have an accessory role. Pitched sounds as discriminated by our classifier trained with the 'note' tag are rather scarce in the database and few of them are used.

Finally, the table of transition counts shows the most common transitions among the morphological categories. Clearly most of the action is among composite sounds, allowing them to determine the rhythmic structure of the piece. However, a few very interesting patches were produced by experimentation with faster sequences of short sounds.

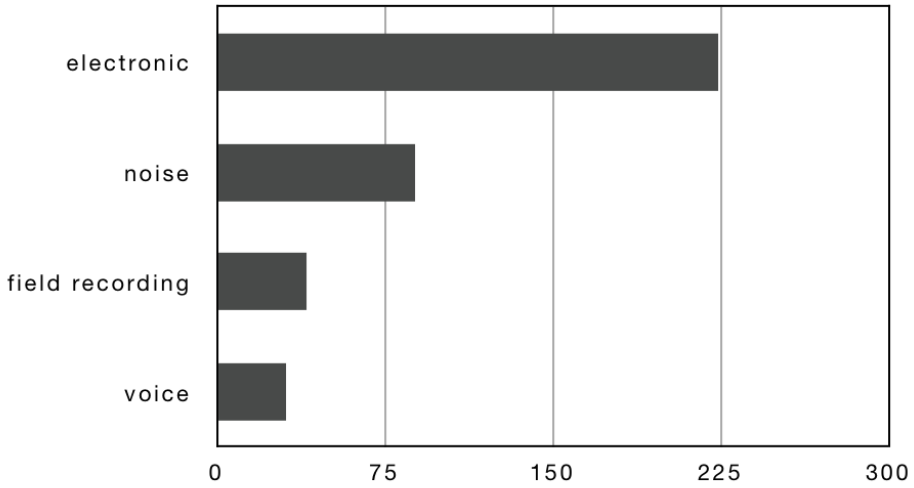


Figure 6.1: Sounds used in patches per conceptual class

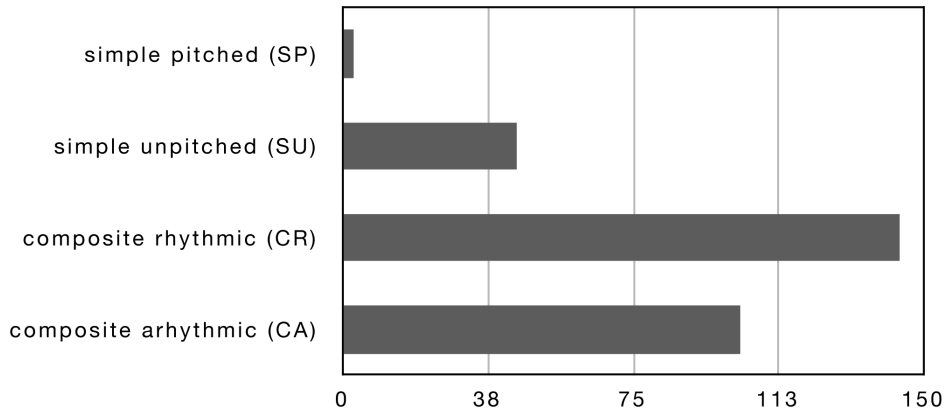


Figure 6.2: Sounds used in patches per morphological class

	SP	SU	CR	CA
SP	0	0	1	0
SU	1	19	9	15
CR	2	14	125	25
CA	0	14	24	63

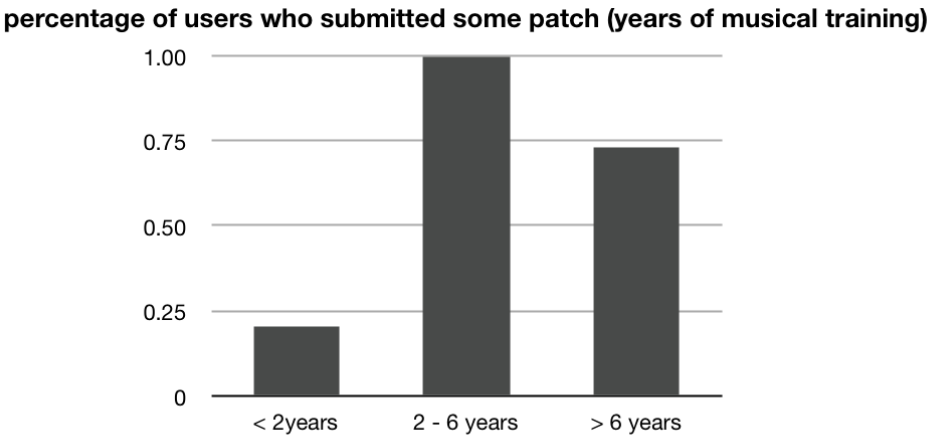
Table 6.1: Number of transitions between morphological classes (rows: source node class, columns: target node class

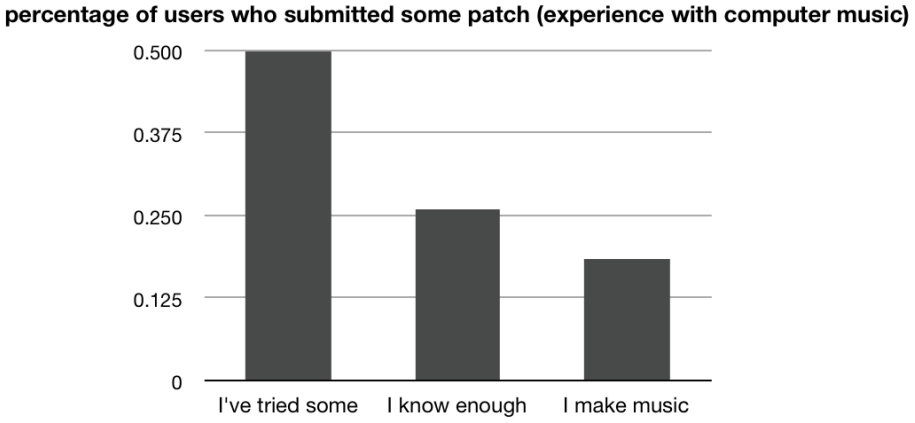
6.2 Sample patch editor

The sample editor included a small survey for users upon registry, and logged the time to complete and the number of searches realized per patch, as well as the number that samples were played in the preview box. An online help tooltip was included with a minimal set of instructions (four lines). In general users found no problems using the interface and no lengthy manual was required. The questionnaire included questions about age, musical training (in four levels) and experience with computer music (also four levels). Once registered, users could listen to other people's patches and submit new ones.

The 20% of users who registered submitted sample patches. From these, there was none with absolutely no musical training. This could mean that there is still some entry barrier, and probably a more friendly help file would be needed, but it may also be related to the level of interest in users towards music and music creation. Since we only recorded submitted patches, we don't have much information to analyze what happened with those who didn't contribute. A higher level of musical training seems to be related with motivation and confidence. Interestingly the opposite is true with experience with computer music. While again no user who reported zero experience submitted a patch, interest seems to decay as users are more accustomed to existing interfaces.

While we expected the interface to allow composing samples quickly, we do not seek to account this activity as a task. We checked that time on the interface was spent in exploring the database, searching and listening to sounds and not on complications with the interface. In extreme cases, there were hundreds of sound previews.





6.3 Evolution

In order to support manual mutation and crossover, the sample patch editor interface provided loading and saving, as well as an *Add* button that allowed merging different patches. In general, the latter didn't find much use. In some cases, users saved snapshots of their own patches and kept working on them. Thus, collab-

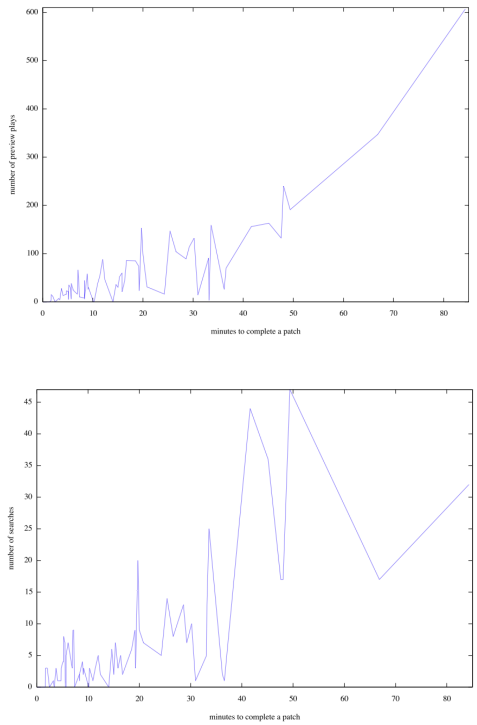


Figure 6.3: Number of searches and previews per patch, sorted by time to complete

oration was mostly provided by the radio player, and this resulted in some very interesting combinations of the different patches submitted by users. Rating was rather spontaneous in users, and in general no explanation was needed about how an evolutionary algorithm works.

One problem in the initial implementation of the algorithm was the tendency to converge, which produced very repetitive patches. This was solved by continuously allowing user-submitted patches to be selected for reproduction, and increasing the mutation probability. Thus, mutation is the main operator to allow exploring the database. A general observation in the evolution of sample patches is that they seemed to converge more than what they actually did. This is due to the predominance of certain samples over the mix which have stronger emotional or musical information. In this sense, some users suggested the need of a sample banning fea-

ture. Some analysis of this difference among predominant and accessory samples could be interesting from a general point of view to understand how samples are used in electronic music.

Regarding tag and sample bookmarks, tag bookmarking was much more used in order to influence the algorithm. Users created more than 500 tag bookmarks but only about 70 sample bookmarks. A single reproduction process with a population size of 6 patches could hit more than 20 tag bookmarks. This highlights the importance of tags in the way users interact with samples in a shared database.

The population size was keep small in order to allow users to understand the effects of their actions. Even so, many users would stay for short periods. In this sense the evolution of patches was easier to understand when used by a single user or a small group of already connected people. This means that for real world usage the output of the radio would benefit from a larger population in the algorithm, since users can pass by at any moment and give their opinion, without the need to actively and consciously follow the evolution of the system.

Similarly, in real world usage the chat was rarely used. The chat window publicly displayed votes of each participant in order to promote shared goals, but again this only seemed to work with a certain level of acquaintance. However, as users came and went, the persistence of the chat display provided a sense of mutual awareness.

Chapter 7

Conclusions and future work

During the development of *Freesound Radio*, many questions arose regarding how musical collaboration at a higher level can emerge from the activity of sharing sounds. We hope we are now closer to an answer some of them, or at least, to foresee which directions have more possibilities to succeed.

In chapter 3, we presented some experiments to determine how to deal with the diversity in the database both from a conceptual point of view and by automatic sound description. By applying a taxonomy extraction algorithm, we obtained a clearer view of the main cultures that are involved in the site. An even clearer map can be obtained by performing some cleanup of the folksonomy and providing aids to users to help maintaining some consistency. A deeper characterization of the usage of the site and its history could be relevant to an understanding of how people are using sounds to create music and multimedia products. This information can lead to further work in organizing and indexing sounds. Regarding the bridging of content description to user semantics, we showed that it is quite possible to establish links that are solid enough. However, for machine learning approaches to succeed great quantities of data are needed. Even an apparently large database like *Freesound* shows some limitations in this respect. Especially with respect to rhythm, the site offers an interesting playground for further research on sound indexing.

The sample patch editor interface presented in chapter 4 worked quite well as a compositional tool on top of a large database. Many users reported it was fun and easy to use, and some of them quickly started to seek the limits of the concept and quickly demanded new features. It could be extended in several ways to allow more complete compositions, like adding attributes for edges (for example the number of times it will be followed) or providing an abstraction mechanism to convert patches into new building blocks. Both the concept and the interface showed potential for collaboration. Using samples recorded or created by other people is obviously already one way to collaborate. On top of that, users would submit their creations and listen to other users' ones, without the complications involved in the traditional

workflow of the music industry, form buying equipment to marketing. One possible improvement regarding collaborative work is allowing several users to create a patch collectively, in real time. On the other hand, further evaluation is required to analyze wether the tool is appropriate for users with absolutely no musical training, possibly using more traditional HCI methodologies.

Finally, the evolutionary algorithm and the radio interface were the most experimental aspects of this work. In this sense, the general approach regarding the generation of new content was proved necessary and useful, on one hand because users were quite shy at starting collaboration by themselves. We assume that traditional music practice in general warrants that such collaboration should be possible. On the other hand, the player interface tended to receive many more visits, which shows the potential of a solution that involves listeners and producers in the same process and tends to erase the distinction among both. In recent times, applications such as Last.fm [1] have become quite successful in implementing a concept of radio station that is not so focused on concurrent interaction but provides features that support the formation of communities and social networks. Further work could focus on allowing personal and shared spaces (radio channels) to be created on demand, as it might be easier to automatically create mixes of sounds that suit personal tastes of users or are evolved upon agreement.

The rules to organize sounds can be learnt from community activity. In this sense, we have completed an initial cycle in supporting music as collective organization of sounds.

The radio can be accessed at **http://radio.freesound.org**

[1] http://last.fm

Bibliography

Al Globus, Sean Atsat, J. L. and Wipke, T.: 2001, Graph crossover, *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO-2001*.

Bacon, F.: 1862, *The works of Francis Bacon*, Brown and Taggard.

Baeza-Yates, R. and Ribeiro-Neto, B.: 1999, *Modern Information Retrieval*, Addison Wesley.

Barbosa, A.: 2003, Displaced soundscapes: A survey of network systems for music and sonic art creation, *Leonardo Music Journal* **13**.

Barbosa, A.: 2005, Public sound objects: a shared environment for networked music practice on the web, *Org. Sound* **10**(3), 233–242.

Biles, J.: 2003, Genjam in perspective: a tentative taxonomy for ga music and art systems, *Leonardo: Journal of the International Society for the Arts, Sciences, and Technology* **36**(1), 43–45.

Bonada, J. and Serra, X.: 2007, Synthesis of the singing voice by performance sampling and spectral models, *IEEE Signal Processing Magazine* **24**(2), 67–79.

Brossier, P.: 2006, *Automatic Annotation of Musical Audio for Interactive Applications*, PhD thesis, Queen Mary University of London, UK.

Bryan-Kinns, N.: 2004, Daisyphone: the design and impact of a novel environment for remote group music improvisation, *DIS '04: Proceedings of the 5th conference on Designing interactive systems*, ACM, New York, NY, USA, pp. 135–144.

Cano, P.: 2007, *Content-Based Audio Search from Fingerprinting to Semantic Audio Retrieval*, PhD thesis, Ph.D. Dissertation. UPF.

Cano, P., Fabig, L., Gouyon, F., Koppenberger, M., Loscos, A. and Barbosa, A.: 2004, Semi-automatic ambiance generation, *Proc. of the 7 Int. Conference on Digital Audio Effects (DAFx'04)*.

Casey, M.: 2001, General sound classification and similarity in mpeg-7, *Organised Sound* **6**(2), 153–164.

Casey, M.: 2005, Acoustic lexemes for organizing internet audio, *Contemporary Music Review* **24, 6**, 489–508.

Chion, M.: 1983, *Guide des objets sonores*, Buchet/Chastel.

Coleman, G.: 2007, Mused: Navigating the personal sample library, *International Computer Music Conference (ICMC)*, ICMC.

Dawkins, R.: 1976, *The Selfish Gene*, Oxford University Press.

de Cheveigné, A. and Kawahara, H.: 2002, Yin, a fundamental frequency estimator for speech and music, *Journal of Acoust Society America* **111**(4), 1917–1930.

Donnadieu, S.: 2006, *Analysis, Synthesis, and Perception of Musical Sounds: The Sound of Music (Modern Acoustics and Signal Processing)*, Springer.

Freeman, J.: 2008, Graph theory: Linking online musical exploration to concert hall performance, *Leonardo* **4**(1).

Gadamer, H. G.: 1977, *Die Aktualität des Schönen*, Reclam.

Gómez, E.: 2006, *Tonal Description of Music Audio Signals*, PhD thesis, Ph.D. Dissertation. UPF.

Gouyon, F.: 2005, *A computational approach to rhythm description. Audio features for the computation of rhythm periodicity functions and their use in tempo induction and music content processing*, PhD thesis, Ph.D. Dissertation. UPF.

Herrera, P., Dehamel, A. and Gouyon, F.: 2003, Automatic labeling of unpitched percussion sounds, *Proceedings of the Audio Engineering Society, 114th Convention*.

Herrera, P., Peeters, G. and Dubnov, S.: 2003, Automatic classification of musical instrument sounds, *Journal of New Music Research* **32**.

Heymann, P. and Garcia-Molina, H.: 2006, Collaborative creation of communal hierarchical taxonomies in social tagging systems, *Technical Report 2006-10*, Stanford University.

Holland, J. H.: 1975, *Adaptation in natural and artificial systems*, University of Michigan Press.

Horner, A., Beauchamp, J. and Hake, L.: 1993, Machine tongues xvi: Genetic algorithms and their application to fm matching synthesis, *Computer Music Journal* **14**(4), 17–29.

Jehan, T.: 2005, *Creating Music by Listening*, PhD thesis, Massachussets Institute of Technology.

Jordà, S., Geiger, G., Alonso, M. and Kaltenbrunner, M.: 2007, The reactable: exploring the synergy between live music performance and tabletop tangible interfaces, *TEI '07: Proceedings of the 1st international conference on Tangible and embedded interaction*, ACM Press, New York, NY, USA, pp. 139–146.

Jordà, S. and Wüst, O.: 2001, Fmol: A system for collaborative music composition over the web, *Proceedings of Web Based Collaboration DEXA 2001*, Munich, Germany.

Kaltenbrunner, M., Geiger, G. and Jordà, S.: 2004, Dynamic patches for live musical performance, *Proceedings of the 4th Conference on New Interfaces for Musical Expression (NIME 04)*, Hamamatsu, Japan.

Kersten, S., Maestre, E. and Ramirez, R.: 2008, Concatenative synthesis of expressive saxophone performance, *Music Computing Conference*.

Kosorukoff, A.: 2001, Human based genetic algorithm, *IEEE International Conference on Systems, Man, and Cybernetics*, Vol. 5, pp. 3464–3469 vol.5.

Lazier, A. and Cook, P.: 2003, Mosievius: Feature driven interactive audio mosaicing.

Miletto, E. M., Pimenta, M. S., Vicari, R. M. and Flores, L. V.: 2005, Codes: a web-based environment for cooperative music prototyping, *Org. Sound* **10**(3), 243–253.

Miller, B. L., Miller, B. L. and Goldberg, D. E.: 1995, Genetic algorithms, tournament selection, and the effects of noise, *Complex Systems* .

Mitchell, T. M.: 1997, *Machine Learning*, McGraw Hill.

Moholy-Nagy, L.: 2004, *Audio Culture: Readings in Modern Music*, Continuum, New York, USA, chapter 8, pp. 331–334.

Peeters, G.: 2005, A large set of audio features for sound description (similarity and classification) in the cuidado project, *Technical report*, CUIDADO I.S.T. Project Report.

Rabiner, L. and Juang, B.-H.: 1993, *Fundamentals of speech recognition*, Prentice-Hall, Inc., Upper Saddle River, NJ, USA.

Ramirez, R. and Hazan, A.: 2005, Understanding expressive music performance using genetic algorithms, *Applications on Evolutionary Computing*.

Reich, S.: 2002, *Writings on music*, Oxford University Press.

Ricard, J. and Herrera, P.: 2004, Morphological sound description: computational model and usability evaluation, *Proceedings of AES 116th Convention*, Berlin, Germany.

Rohrhuber, J.: 2008, *The Cambridge Companion to Electronic Music*, Cambridge University Press.

Schaeffer, P.: 1966, *Traité des objets Musicaux*, Seuil, Paris.

Schafer, R. M.: 1977, *The tuning of the world*, Random House.

Schwarz, D.: 2004, *Data-Driven Concatenative Sound Synthesis*, Thèse de doctorat, Université Paris 6 - Pierre et Marie Curie, Paris.

Schwarz, D.: 2005, Current research in concatenative sound synthesis, *International Computer Music Conference (ICMC)*, Barcelona, Spain.

Schwarz, D.: 2006, Concatenative sound synthesis: The early years, *Journal of New Music Research* **35**(1), 3–22. Special Issue on Audio Mosaicing.

Schwarz, D.: 2007, Corpus-based concatenative synthesis : Assembling sounds by content-based selection of units from large sound databases, **24-2**, 92–104.

Secretan, J. and Beato, N.: 2008, Picbreeder: evolving pictures collaboratively online, *CHI '08: Proceeding of the twenty-sixth annual SIGCHI conference on Human factors in computing systems*, ACM, New York, NY, USA, pp. 1759–1768.

Serra, X.: 2007, State of the art and future directions in musical sound synthesis, Chania, Crete.

Shamma, D. A., Shaw, R., Shafton, P. L. and Liu, Y.: 2007, Watch what i watch: using community activity to understand content, *MIR '07: Proceedings of the international workshop on Workshop on multimedia information retrieval*, ACM, New York, NY, USA, pp. 275–284.

Sturm, B. L.: 2006, Adaptive concatenative sound synthesis and its application to micromontage composition, *Comput. Music J.* **30**(4), 46–66.

Tanaka, A., Tokui, N. and Momeni, A.: 2005, Facilitating collective musical creativity, *MULTIMEDIA 05: Proceedings of the 13th annual ACM international conference on Multimedia*, ACM, New York, NY, USA, pp. 191–198.

Tillmann, B., Bharucha, J. J. and Bigand, E.: 2000, Implicit learning of music: A self-organizing approach, *Psychological Review* **107**, 885–913.

Tzanetakis, G. and Cook, P.: 1999, Multifeature audio segmentation for browsing and annotation, *in Proc.1999 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics, WASPAA99*.

Valle, A.: 2008, Geography: a real-time, graph-based composition environment, *Proceedings of the 8th Conference on New Interfaces for Musical Expression (NIME 08)*.

Weinberg, G.: 2005, Interconnected musical networks: Toward a theoretical framework, *Computer Music Journal* **29**(2), 23–39.

Whalley, I.: 2004, Piwecs: enhancing human/machine agency in an interactive composition system, *Org. Sound* **9**(2), 167–174.

Witten, I. and Frank, E.: 2000, *Data Mining: Practical machine learning tools with Java implementations*, Morgan Kaufmann.

Zils, A. and Pachet, F.: 2001, Musical mosaicing, *Proceedings of the COST G-6 Conference on Digital Audio Effects (DAFx-01)*, Limmerick, Ireland.