AudioCycle: Browsing Musical Loop Libraries

Stéphane Dupont, Thomas Dubuisson, Jérôme Urbain,
Raphaël Sebbe, Nicolas d'Alessandro
Faculté Polytechnique de Mons
TCTS Lab
Boulevard Dolez 31
7000 Mons, Belgium
stephane.dupont@fpms.ac.be

Christian Frisson
Université Catholique de Louvain
TELE Lab
Place du Levant, 2
1348 Louvain-la-Neuve, Belgium
christian.frisson@uclouvain.be

Abstract

This paper presents AudioCycle, a prototype application for browsing through music loop libraries. AudioCycle provides the user with a graphical view where the audio extracts are visualized and organized according to their similarity in terms of musical properties, such as timbre, harmony, and rhythm. The user is able to navigate in this visual representation, and listen to individual audio extracts, searching for those of interest. AudioCycle draws from a range of technologies, including audio analysis from music information retrieval research, 3D visualization, spatial auditory rendering, audio time-scaling and pitch modification. The proposed approach extends on previously described music and audio browsers. Concepts developed here will be of interest to DJs, remixers, musicians, soundtrack composers, but also sound designers and foley artists. Possible extension to multimedia libraries are also suggested.

1. Introduction

Music production is more and more relying on pre-recorded material. One of the dominant musical instruments synthesis paradigm is based on libraries of high-quality recordings of real instruments, covering their different articulations and expressive modes, leading to very large synthesis libraries, close to the gigabyte for a single instruments, with physical modeling probably being at the other extreme in terms of computation/memory trade-off [24]. Also, the production of many music styles heavily relies on pre-recorded musical phrases, organized into libraries. This is most apparent for instance in styles like hip-hop or remixing, where these phrases, often referred to as loops (for instance drum loops), are indeed looped, sequenced, and mixed together to form full audio tracks

and compositions. Third, granular synthesis techniques have also been successfully used in music creation, as a way for discovering new musical textures, or in live settings [25, 31].

However, the tools available today for browsing through large musical libraries hinders the creative process. Indeed, loops can be searched through rigid file system hierarchies, or through the use of symbolic descriptors (music style, instruments used) stored as meta-data in the library. Also, this processes being mostly off-line can not be used during performances. In the music performance area, some DJ products provide assistance to the performer, for instance through real-time tune synchronization capabilities, but searching through the library is still being done the usual way [29]. More generally, with the growing availability of multimedia content, there is a still larger demand for more flexible and efficient tools to access content and search for data. Information indexing and retrieval can rely on automatic technologies to describe contents on one hand and, on the other hand, allow formulating queries and structure the media database to help the user to navigate through it. The work presented here envisions a convenient and fast way of exploring large audio and music extract libraries, relying on similarity analysis and musically relevant audio analysis and transformation, involving rhythm, harmony and timbre [20, 23, 18, 27]. Other features that are specific to the application to music content have also been implemented. Loops can be played in a beat-synchronous fashion, relying on a phase vocoder algorithm to adjust the tempo without affecting the pitch. AudioCycle differs from previously published approches. Compared to SoundTorch [16] for instance, it provides a range of audio analysis approaches (not only timbral) and enables to weight their importance. On the usability and rendering side, AudioCycle enables to play synchronously any combination of loops, even if they are very distant on the similarity map.

The paper is organized to describe the different blocks of

the AudioCycle architecture, schematized in Figure 1. First, an *Audio Analysis (2)* is performed on a set of loops (*1*) loaded by the user. It consists of extracting features representative of the musical properties of rhythm, harmony, and timbre. This is presented in Section 2. The specificities of the algorithms that have actually been implemented are also described. This step generates a database gathering features and meta-data (including tempo and key found in some audio file formats) for each loop (*3*). Section 3 is interested by *Visualization (4)* techniques that can help users to navigate and search inside large digital media libraries. These techniques rely on the previously created Features Database (*3*), and approaches for mapping this large dimensional space to a 2D or 3D structured representation of the library content are explored. Users can interact with this representation, influencing the subsequent *3D Visual Rendering (5)* - also addressed in Section 3 - and *3D Audio Rendering (6)*, which spatializes, synchronizes and plays the selected loops - presented in Section 4. Improvements to be implemented in terms of visualization and interaction and suggestions for future activities are finally proposed in Section 5.
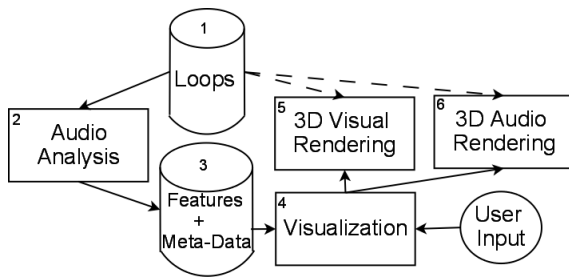

Figure 1. AudioCycle Architecture

2 Music Analysis

To characterize music chunks and organize them according to their similarities, a set of relevant mathematical descriptors is needed. They will be introduced hereunder. Then, methods for computing an important rhythmic notion, the tempo (expressed in Beats Per Minute - BPM), and estimating the downbeat positions in a song will be described. Localizing the downbeats is useful for appropriately cutting a music file into bars, which are not always similar, and synchronizing musical segments. To conclude this Section, results of experiments assessing the relevance of the chosen timbral features are presented.

2.1 Feature Extraction

Following a content-based analysis approach, features providing information about the timbre, the harmony and

the rhythm of music excerpts are extracted from frames whose length is 30ms and whose hopsize is 10ms.

2.1.1 Timbre

Timbre is defined as the quality of tone distinctive of a particular singing voice or musical instrument [7]. It is linked to the relative intensities of a sound harmonics, independantly from its pitch and intensity. It has been shown in many studies [18, 20, 21] that timbre can be efficiently characterized by the instantaneous spectral envelope of the sound, using a non-linear frequency scale. A popular non-linear frequency scale is the Mel scale, built to model the human perceptions of pitches. Based on this scale, the frequency content of a frame can be summarized by the Mel-Frequency Cepstral Coefficients (MFCCs) [30]. To characterize the timbre in our application, we use a filterbank of 20 filters covering the frequency range between 0Hz and the Nyquist frequency (22050Hz for the music excerpts we used), and keep only the first 12 MFCCs (+ the energy).

2.1.2 Harmony and Melody

Harmony can be defined as the combination of simultaneous musical notes in a chord [7]. In addition, the melody is defined as a rhythmic succession of single tones organized as an aesthetic whole [7]. In consequence, to characterize the harmony and melody of a musical signal, features revealing the presence of particular notes, their evolution and combination must be extracted. Since notes are defined by frequency values, harmony and melody analysis is generally performed in the frequency domain. A widely used method is to project the power spectrum of each frame on a chromatic scale, which decomposes each octave into 12 equally broad bands on a base 2 logarithmic scale, called "semitones". The "chromas" characterizing the harmony of a frame are then the 12 sums of the energetic contributions of each semitone on all the (considered) octaves. They represent the combination of notes played at a given time, independently from their octave. To some extent, the "chromas" also contain an indication of the melody, although melody can span several octaves.

2.1.3 Rhythm

Rhythm is defined as the aspect of music comprising all the elements (as accent, meter, and tempo) that relate to forward movement [7]. From a signal processing point of view, it is closely related to the notion of periodicity. Various analyses of a signal periodicity can be proposed. For characterizing the rhythm of musical excerpts, one popular way is to determine onsets instants - which can also serve to decompose a musical chunk into bars, beats, etc. To estimate these onset instants, it is common to compute the Perceptual Spectral

Flux (PSF) [18], which can be seen as an integration of the local derivatives of the spectrogram frequency bins. The algorithm proposed in [20] for computing the PSF, also called "onset function" is:

- To compensate for human ear attenuation, weight the power spectrum of each frame with the inverse of the Equal Loudness Curve (EQL) defined at 40 phons [17] (see Eq. 1 where f denotes the frequency).

$$EQL(f) = (\frac{f^2}{f^2 + 1.6 * 10^5})^2 * (\frac{f^2 + 1.44 * 10^6}{f^2 + 9.61 * 10^6}) \tag{1}$$

- Compute the Perceptual Spectral Flux:

$$PSF(n) = \sum_{k=1}^{N_b/2} ((a_n^k)^{1/3} - (a_{n-1}^k)^{1/3}) \tag{2}$$

where n stands for the frame index, N_b for the number of frequency bins and a_n^k for the compensated power spectrum at the kth frequency bin of current frame n. The exponent $1/3$ is introduced to simulate the intensity-loudness power law. As in [12], the power spectrum was obtained by applying a Mel filterbank decomposition on top of the usual Fourier Transform.

In [20] and [8], the rhythmic information is computed by calculating local autocorrelation functions on overlapping frames of the PSF.

Another approach has been designed here, based on the notions of bar, beat and tatum, which are the onset events related to the perception of rhythm (see Section 2.2). We have chosen to save as rhythmic features the onset values in the neighbourhood of the beats and tatums positions for one bar. The window length is set to 11 frames: PSF values are summed using a mask from 5 frames before and until 5 frames after the metrical event position. Negative onset values are discarded from the sum, since rhythmic events are perceived when a particular sound is striking, not disappearing. Since rhythm can be characterized by the contributions of different frequencies to the onset function, it has been decided to split the onset function in three frequency bands: low frequencies (0 to 100 Hz), medium frequencies (100 to 8000 Hz) and high frequencies (8000 Hz to the Nyquist frequency).

A 4/4 rhythmic signature is assumed, meaning a bar contains 4 beats - the first one being the downbeat - and 4 tatums, located halfway between two beats. Thus, to capture the beat and tatum information of one 4/4 bar, features around 8 particular instants must be computed. The vector characterizing the rhythm of each music excerpt is in consequence composed of 8 values (PSF sums within a mask region) for each frequency band. The frequency band decomposition of the onset function and mask applied for computing the rhythm features are illustrated in Figure 2. It can be

seen that the mask - generated independently from the onset function - will catch the main peaks of the onset functions, corresponding to beats and tatums. Only the first 8 steps of the mask are used to compute the rhythmic features.
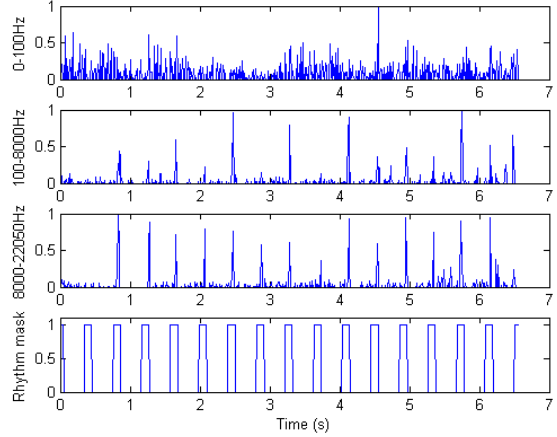

Figure 2. Top 3 graphs: band frequency decompositions of the onset function (normalized amplitudes). Bottom graph: Mask applied for computing the rhythmic features.

The described method requires the knowledge of the downbeat position and the interval between two successive beats. The latter can be easily computed from the Beats Per Minute (BPM) value, frequently available as meta-data with professional music files. Loops used for DJing applications generally start at the downbeat, solving the problem of localizing this event. When the BPM and downbeat position are a priori unknown, it is still possible to compute them and use the proposed method for extracting the rhythmic features thanks to the algorithms that will be described in Section 2.2.

2.1.4 Summarizing the information

The targeted application must be able to deal with thousands of musical extracts, fastly compare them and display them in a suitable way (see Section 3). This objective has two implications on the feature sets we provide. First, the features vectors must be of reasonable length, otherwise computation times would dramatically increase and the application would not be efficient in real-time. Second, characteristic vectors of each musical excerpt should have a constant length, independently from its duration, in order to facilitate comparisons of the signal properties.

Regarding the rhythm characterization, the method aforementioned already provides a constant number of 24 rhythmic features per music excerpt. However, as the num-

ber of timbre and chroma features is proportional to the music excerpt duration, this information is summarized into fixed size vectors by storing only the mean and variance of each timbre and harmony feature. It is shown in Section 2.3 that even if information is lost when summarizing, the mean and the variance of the features still contain enough information to efficiently recognize the kind of instruments involved in the extract, which is the aim of timbre characterization.

2.2 Beat and Downbeat Estimation

In the Music Information Retrieval domain, the notion of "beat" is often expressed by musicians and listeners through the act of foot tapping along to the music. Downbeat is defined as the first beat mark at the beginning of a bar.

Beat tracking systems form the basis of applications like automatic accompaniment, transcription, computer-assisted audio editing and evaluation of music similarity.

2.2.1 State of the Art

The onset function (see Section 2.1) is commonly used to extract the tempo (expressed in BPM). The method proposed in [12] estimates it by computing the autocorrelation of the onset function and by favoring autocorrelation values located in a range defined by the user. It is performed by weighting the autocorrelation function by a Gaussian window centered around the most likely BPM value and whose spread is set by the user. Once the largest peak is identified in the weighted autocorrelation, a second value of BPM is found as the largest value of autocorrelation at 0.33, 0.5, 2 or 3 times the first value of BPM. This system was tested on the database of the MIREX 2006 contest [1] and achieved 77 % of agreement with the manual BPM.

Among the other beat tracking methods based on the onset function, the system exposed in [22] computes the onset function from a reassigned spectrogram, in which the energy of the bins (at frequency ω_k at time t_i) are reassigned to the frequency ω_r and time t_r corresponding to their center of gravity. BPM is then estimated by combining periodicity measures of the onset function (Discrete Fourier Transform and Autocorrelation). In [15], the structure of a musical excerpt (in terms of bar, mid-bar and beat) is estimated by computing an onset function from different frequency bands in the spectrogram. Chord-change possibilities and drum patterns are then estimated at beat locations and the structure of the music excerpt is inferred from these informations.

Fewer studies focus on the difficult problem of downbeat estimation altough it has a perceptual sense for us. A method for downbeat extraction based on beats locations and BPM is presented in [10]. The signal is cut into segments synchronized at beat locations and whose length is the beat period. The difference between the spectrum of successive segments is then computed. For each bar, the downbeat is located where value of this difference is highest. In [19], Support Vector Machines are trained in order to predict the downbeat locations by using spectral features synchronized on the local maxima of the onset function. In the testing phase, the system takes as input the spectral features of several music segments and provides an estimation of the downbeat locations in the next segments.

2.2.2 Implementation

As the onset function appears to be very popular for beat period estimation, it has been chosen in the present study. Onset functions from [12, 9, 15, 22] have been compared and appeared quite similar. Because Ellis' method [12] provides good performances and because the whole source code is provided on his website, it has been selected as the beat period estimation in our system.

Concerning the downbeat estimation, a method inspired by previous litterature [19] has been developed. The idea is to find the best alignment between a template of bar, beat and tatum locations and the onset function. As in [19], it has been chosen to consider only templates for 4/4 time signature in this study. The template is built using the estimated BPM value. An example is shown in Fig.3 for 144 BPM. The best alignment is achieved by computing the cross-correlation between the onset function and a lagged version of the template. This correlation is computed for lags from 0 to one bar period. The lag corresponding to the highest correlation value is used to set the downbeat.
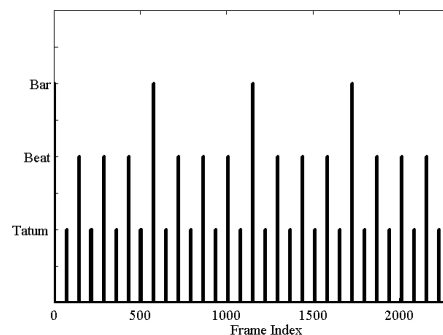

Figure 3. Template for 144 BPM

The algorithms implemented for computing the beat period and localizing the downbeat position also enable us to perform "slicing" (i.e. decomposition into bars) of audio files. Not only loops but also full songs can thus be analyzed by the application. The algorithm designed for slicing consists of first cutting the files into 15s-windows (with a 10s-overlap), then launch the computation of the beat period and finally localize the downbeats position in the cen-

tral 5s of the window. Resulting bars are then processed by the application as if they were loops.

2.3 Instruments Classification

In order to determine to what extent the chosen timbre features are indeed representative of timbre perception and to define which set of features is the most appropriate for the similarity estimation and visualization in AudioCycle, their performance for classifying instruments has been assessed. The mean and variance of the timbre features have been considered in this evaluation.

2.3.1 Database

The features are extracted from musical loops of the ZeroG ProPack ACID database [6]. This database is a product of ZeroG Ltd. and contains 11428 loops and samples of various instruments in diverse musical styles. We have manually tagged the files using 7 instruments classes: Brass, Drums, Vocals, Percussion, Electric Bass, Acoustic Guitar and Electric Guitar. Only the files for which the instrument class is clear have been taken into account in this study. 4380 files have been used in total.

2.3.2 Dimension Reduction

Before classifying the feature vectors, they have to be manipulated in order to assess if their dimension can be reduced. For this purpose, the Generalized Fisher criterion and Principal Component Analysis (PCA) have been tested.

The Generalized Fisher criterion [14] is a class separation criterion based on the features and the class labelling. For a given feature k, its discriminant power between C classes is defined as

$$D_k = \frac{\sum_{c=1}^{C} p(\omega_c)(\mu_{ck} - \mu_k)^2}{\sum_{c=1}^{C} p(\omega_c)\sigma_{ck}^2} \quad (3)$$

where $p(\omega_c)$ stands for the percentage of representation of class c in the database, μ_{ck} for the mean of the feature k in the class c, μ_k the mean of feature k for all the classes and σ_{ck} for the standard deviation of feature k in the class c. A dimension reduction is possible by selecting the features associated to the highest values of discriminant power.

Principal Component Analysis (PCA) [14] prepares the features in a classification system by linearly transforming them in order to find the best representation space (in terms of least square error). If X represents the normalized features matrix, the new features matrix Z is obtained by

$$Z = U^T X \quad (4)$$

where U is the linear transformation matrix. The matrix U leading to the best final representation consists of the eigen vectors of the covariance matrix XX^T. The dispersion of features around each new axis of representation is given by its associated eigen value. A reduction of features dimensionality is possible by selecting the axis of representation associated to the highest eigen values.

2.3.3 Results

Three ways of reducing the dimension of features vectors have been investigated: applying the Fisher criterion alone, applying PCA alone and applying Fisher criterion followed by PCA. These different approaches have been applied on the mean and variance of the timbre features. For each configuration, a Multi-Layer Perceptron was trained with 60% of the objects, validated with another 10% and tested with the remaining 30%. The operation was repeated 100 times and the objects were always randomly selected. Table 1 shows the average classification performance for each configuration of features.

Table 1. Performance of instruments classification using timbre features(in %)

Description	Mean	Var
No selection	73.1	50.8
6 most discriminant after Fisher	66.2	52.1
2 most discriminant after Fisher	54.3	47.2
No selection after PCA	93	70.9
6 most discriminant after PCA	85	67.7
2 most discriminant after PCA	66	52.6
No selection after Fisher and PCA	93	68.1
6 most discriminant after Fisher then PCA	86.9	59.8
6 most discriminant after Fisher then 2 most discriminant after PCA	75	56.3
2 most discriminant after Fisher then 2 most discriminant after PCA	64.1	51.9

Classification performance for timbre information is very good (up to 93%), supporting our decision to summarize the timbre information (see Section 2.1.4) and confirming that these vectors still contain enough information to distinguish instruments with a very high accuracy. Since performance is significantly better using means than variances and further tests shown that no improvement was achieved by combining means and variances, only timbre means are finally used for the AudioCycle application. Similar tests are under way to assess the opportuneness of summarizing the chroma information the same way.

3 Visualization

3.1 Visualization Techniques

The representation of large multimedia database has been a subject of research in various cases of media content. As stated in [26], the visualization has to meet three requirements: overview (faithful overview of the distribution of excerpts in the collection), structure preservation (the relations between music excerpts should be preserved in the projection in the visualization space) and visibility (visually understanding of the content of each music excerpt). Thus, if the similarity between music excerpts is used, the ideal is to build a representation in which the similarity distances are preserved, what our choice of the node-link visualization paradigm, great for emphazising two Gestalt laws (similarity and connectedness) [28], ensures.

When displaying the whole collection of music excerpts is desired, the idea is to associate to an object x_i a vector y_i aiming at transposing the similarity between x_i and its neighbours in a 2D or 3D representation. Three kinds of approaches are described hereafter.

3.1.1 Dimensionality Reduction

The goal of this approach is to preserve the distance between points in a high dimensional space when they are projected in a 2D subspace. This can be achieved by e.g. PCA or by Linear Discriminant Analysis (LDA) [11]. This latter projects the features vectors from a d-dimensional space to a D-dimensional space (with $D < d$) with the constraint to maximize the ratio between the between-class covariances and the within-class covariances of the new features. This is done by applying the linear transformation

$$Z = W^T X \tag{5}$$

where W is the transformation matrix whose dimension is $d \times D$.

Among the LDA techniques, one can cite the Fisher method which consists on finding $C - 1$ linear discriminant functions in a problem of C classes. It can be proved that the columns of W that maximize the ratio between the between-class covariance and the within-class covariance correspond to the highest eigen values of

$$S_B W_i = \lambda_i S_W W_i \tag{6}$$

where W_i stands for a column of W and λ_i for the ith eigen value. As only $C-1$ eigen values are different of 0, the new system of representation consists of only $C-1$ dimensions.

3.1.2 Graphs and Multidimensional Scaling

When using graphs for visualization [26], each excerpt is represented by a vertex in the graph and the arcs between neighbouring excerpts are used for navigation. The objective of the graphs is to preserve the neighbourhood of excerpts and to enhance data structures. For retrieval and analysis purposes, the radial tree structure and global graph are particularly adapted. The radial tree structure provides a focus on a part of a graph, analogous to a hierarchical representation. The global graph aims at representing a large amount of excerpts with various levels of details, but with a larger point of view than radial tree structures.

Multi Dimensional Scaling (MDS) aims at trading with the high dimension of data by using graphs. The objective is to iteratively optimize a function that measures the difference between the distance of two excerpts in the 2D space and their original dissimilarity. The original method requiring a high computational cost, other approaches have been developed. Among them, one can cite Isomap, aiming at clustering first the data before applying MDS or Visumap, which maps the excerpts in a 2D map so that Euclidian distance visually approaches as much as possible the dissimilarities of the excerpts.

3.1.3 Clustering

Clustering is another way to organize a large amount of data. An original approach has been developed here. It relies on a K-Means algorithm and aims at grouping loops characterized by their extracted features vectors. Clusters of loops are built by iteratively constructing partitions through association of each loop to the closest centroid. The Euclidian distance has been chosen and based on the extracted features defined earlier. The emphasis of the clustering can be changed by the user by scaling differently the feature space dimensions related to timbre, harmony, and rhythm. The number of clusters is chosen in advance by the user and, in order to preserve near real-time interaction, the clustering is stopped before convergence, after a limited number of iterations. The initial cluster centers are randomly chosen among the loops but this random sequence is made reproducible to provide back and forth navigation.

Clusters can be represented in a flower-like scheme, where the center is the currently selected loop and other loops are gathered into groups around it. The 2D radial distance between a loop and the selected centered element is proportional to the distance between them (in a feature space with scaling factors complementary to the one used for clustering), while the distance between the loop and the cluster center (in the scaled space used for clustering) is used to compute the 2D angular coordinate to locate it in the cluster. This scheme allows to explore organization schemes where the radial and angular coordinates are associated to complementary musical properties, e.g. timbre and rhythm.

3.2   3D Imaging, OSG and OpenGL

Visualization of the audio loops is made with Open Scene Graph (OSG) [4], a scene graph visualization library running on top of OpenGL and permitting the definition of high level objects. These objects can either render geometry or represent 3D transformations, and, when associated as a graph, they define a rendered image.

The scene graph is built by adding cubes representing each loop and linked to an OpenAl source (see Section 4). Each of these cubes has a parent node specifying its position, which can change over time, when the user changes the clustering properties for instance. When a loop is activated, the scene graph is enhanced with a waveform display and a curser indicating the current playback position. Loops can be activated by clicking on the associated objects (cubes). This process is known as picking and is implemented in the usual OSG way by computing intersection with a ray fired at the mouse location. The event is then processed, the waveform is displayed and the OpenAl source is played. A screenshot of the Audiocycle OSG view is shown in Fig. 4. OSG will allow us to extend this work to 3D representations very naturally.
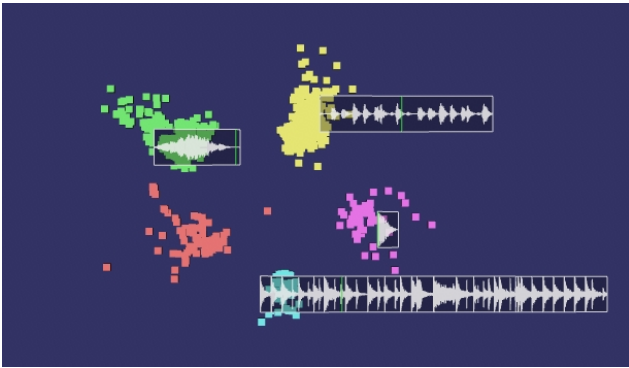

Figure 4. AudioCycle OSG View

4   Audio Rendering

In AudioCycle, several audio rendering features have been designed to facilitate the browsing through large libraries of audio loops. First, the loops are positioned in space in relation with the position of their visual representation on the screen. This tight coupling between the audio and the image is expected to ease the location of the area of interest in the visual representation [13, 16], when the user is looking for specific sounds while several loops are playing together. This also allows to play simultaneously several sounds. This has been implemented using OpenAL (described hereafter) with a specific extension providing HRTF-based (Head Related Transfer Functions) rendering.

Then, as we are dealing with music content, it makes sense to provide the application with facilities to allow beat synchronous playback (through audio time-scaling) of loops that have originally not been played at the same tempo, and key normalization (through pitch shifting) for loops that have originally not been played in the same keys. This is expected to reduce the cognitive load when several loops are playing together, hence facilitating the search for specific sounds. Eventually, this may also provide the basis for extending AudioCycle to a performance tool, where live composition and performance is made possible through beat and key matching.

OpenAL [2] is a cross-platform audio API that has been designed for rendering of multichannel 3D audio. Designed with a style similar to OpenGL, it has mostly been used for 3D modeling and game engines. The API allows to define *source* objects and a *listener*. Sources can be specified using several properties, including position, velocity, sound direction and intensity. The listener is also characterized by properties including position, velocity, direction and a global gain. The buffers containing the audio samples in PCM format are associated to the sources by other API calls. The OpenAL engine then performs the rendering and simulates physical effects such as the attenuation caused by distance between the sources and the listener, as well as the Doppler effect.

Additional functionalities can be provided through extensions to the API. Among the available plugins, an AU (Audio Unit) plugin making use of HRTF when rendering for a stereo output has been used in the application. This conveys an enhanced realism to the sound scene.

Coming back to sources, it is important to note that they can be of 2 types: *static* or *streaming*. As static sources are not appropriate for applications where expressive control is required, streaming sources have been preferred. It allows to pass buffers that are queued for real-time playback. Since the API does not provide a callback mechanism that requests audio buffers, this aspect has hence to be managed by the application. In this work, this is done using a thread with the necessary priority level to periodically check whether buffers will be needed by the OpenAL engine.

5   Future Work

First, it is planned to redesign some aspects of the visualization [28], by e.g. respecting the symetry of the 3D node-link diagram paradigm and condensing the mapping of the waveforms around each loop representation [28].

Then, in addition to the usual keyboard/mouse couple support, we are investigating new methods for gestural control of multiple parameters of the AudioCycle application (in terms of navigation, visualization, etc...), such as a bimanual interaction using a "3D mouse" for the navigation

mode and a "jog wheel" for the temporal/spectral modification mode, so as to recenter the user's attention on the task to achieve. Modular environments such as PureData [5] and OpenInterface [3] are being considered for the rapid prototyping.

Also, in order to achieve the fastest possible audio mining, we initiated a user-centered usability approach, featuring context enquiries with experts (sound designers [24], DJ's [29], etc...) and usability tests with software/hardware prototypes, leading to the definition and support of different user and hardware profiles.

Finally, the present work will be extended to multimedia content (images, videos, text, etc...) with scalability to very large archives.

6   Conclusion

A novel approach and prototype implementation of a music loop library browser has been presented. It allows to display the music extracts organized on a map according to timbre, harmony, rhythm, or a combination of these. Spatial sound is also used to enhance the experience when playing several extracts at the same time. Features relevant to music production have also been integrated, including the possibility to play the loops in a beat-synchronous fashion, as well as alter their pitch. Some avenues for extending and improving the concept and technologies involved are also proposed.

7   Acknowledgments

References

[1] Music information retrieval evaluation eXchange. http://www.music-ir.org.

[2] Openal. http://www.openal.org/.

[3] Openinterface. http://www.openinterface.org.

[4] Openscenegraph (osg). http://www.openscenegraph.org/.

[5] Puredata. http://www.puredata.info.

[6] Zerog. http://www.zero-g.co.uk/.

[7] Merriam-webster dictionary online, Consulted on January 14, 2009. http://www.merriam-webster.com/.

[8] L. Couvreur, F. Bettens, T. Drugman, T. Dubuisson, S. Dupont, C. Frisson, M. Jottrand, and M. Mancas. Audio thumbnailing. In T. Dutoit and B. Macq, editors, *QPSR of the numediart research program*, volume 1, pages 67–85, 2008. http://www.numediart.org.

[9] M. Davies and M. Plumbley. Beat tracking with a two state model. In *Proceedings of ICASSP 2005*, 2005.

[10] M. Davies and M. Plumbley. A spectral difference approach to downbeat extraction in musical audio. In *Proceedings of EUSIPCO 2006*, 2006.

[11] R. Duda and P. Hart. *Pattern Classification and Scene Analysis*. John Wiley and Sons, 1973.

[12] D. Ellis. Beat tracking with dynamic programming. In *Proceedings of MIREX 2006*, 2006.

[13] M. Fernström and E. Brazil. Sonic browsing: An auditory tool for multimedia asset management. In *International Conference on Auditory Display (ICAD2001)*, Helsinki, Finland, 2001.

[14] K. Fukunaga. *Introduction to Statistical Pattern Recognition*. Academic Press - Second Edition, 1990.

[15] M. Goto. An audio-based real-time beat tracking system for music with or without drums-sounds. *Journal of New Music Research*, 30(2):159–171, 2001.

[16] S. Heise, M. Hlatky, and J. Loviscach. Soundtorch: Quick browsing in large audio collections. In *125th AES Convention*, San Francisco, CA, Oct 2008.

[17] H. Hermansky, N. Morgan, and P. D. Korn. Auditory model for parametrization of speech (united states patent 5,450,522), 1995.

[18] T. Jehan. *Creating Music by Listening*. PhD thesis, September 2005.

[19] T. Jehan. Downbeat prediction by listening and learning. In *Proceedings of WASPAA 2005*, 2005.

[20] K. Jensen. Mutliple scale music segmentation using rhythm, timbre and harmony. *EURASIP Journal on Advances in Signal Processing 2007*, 7, 2007.

[21] G. Peeters. Toward automatic music audio summary generation from signal analysis. In *Proceedings of the International Symposium on Music Information Retrieval (ISMIR)*, 2002.

[22] G. Peeters. Time variable tempo detection and beat marking. In *Proceedings of ICMC 2005*, 2005.

[23] G. Poliner, D. Ellis, A. Ehmann, E. Gomez, S. Streich, and B. Ong. Melody transcription from music audio: Approaches and evaluation. *IEEE Transactions on Audio, Speech and Language Processing*, 15(4):1247–1256, 2007.

[24] M. Russ. *Sound Synthesis and Sampling*. Music Technology. Focal Press, 3 edition, 2008.

[25] D. Schwartz. Musical applications of real-time corpus-based concatenative synthesis. In *ICMC 2007*, Copenhagen, Aug 2007.

[26] A. Veroust-Blondet. State-of-the-art on advanced visualisation methods. Technical Report D7.2, VITALAS project, Sep 2007.

[27] H. Vinet, P. Herrera, and F. Pachet. The cuidado project: New applications based on audio and music content description. In *Proc. of ICMC*, pages 450–454, Göteborg, Sweden, Sep 2002.

[28] C. Ware. *Information Visualization: Perception for Design*. Morgan Kaufmann, 2 edition, 2004.

[29] S. Webber. *DJ Skills: The essential guide to Mixing and Scratching*. Focal Press, 2007.

[30] F. Zheng, G. Zhang, and Z. Song. Comparison of different implementations of mfcc. *Journal of Computer Science and Technology*, 16:582–589, 2001.

[31] A. Zils and F. Pachet. Musical mosaicing. In *Proc. of COST G-6 Conference on Digital Audio Effects (DaFx-01)*, pages 39–44, Limerick, U.K., Dec 2001.