Initial experiments with Multiple Musical Gestures

Jensen, Kristoffer; Graugaard, Lars

Published in:
Proceedings of the Sound in Interactive Media Workshop

Publication date:
2005

Document Version
Early version, also known as pre-print

Link to publication from Aalborg University

Citation for published version (APA):

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

? Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
? You may not further distribute the material or use it for any profit-making activity or commercial gain
? You may freely distribute the URL identifying the publication in the public portal

Take down policy
If you believe that this document breaches copyright please contact us at vbn@aub.aau.dk providing details, and we will remove access to the work immediately and investigate your claim.

Downloaded from vbn.aau.dk on: November 26, 2021
ABSTRACT
The classic orchestra has a diminishing role in society, while hard-disc recorded music plays a predominant role today. A simple to use interface in 2D for producing music is presented as a means for playing in a social situation. The sounds of the music are produced by a low-level synthesizer, and the music is produced by simple gestures that are repeated easily. The gestures include left-to-right and right-to-left motion shapes for spectral envelope and temporal envelope of the sounds, with optional backwards motion for the addition of noise; downward motion for note onset and several other manipulation gestures. The initial position controls which parameter is being affected, the notes intensity is controlled by the downward gesture speed, and a sequence is finalized instantly with one upward gesture. The synthesis employs a novel interface structure, the multiple musical gesture interface. The actual control mechanism is a color sensitive camera that by filtering out certain colors can identify the movements of a laser pointer. The gestures of the pointer control either sound, note or timbre parameters depending on the angle of incidence of the light. Several synthesis methods are presented and the control mechanisms are integrated into the multiple musical gesture interface. This enables a number of performers to interact on the same interface, either by each playing the same musical instruments simultaneously, or by performing a number of potentially different instruments on the same interface.

1. INTRODUCTION
The common understanding of a musical instrument is that it controls the note pitch, length and dynamics to some degree, and furthermore has some possibilities of changing the timbre, most noticeable in the singing voice and less in other acoustic instruments. In modern music, even less control is often obtained, because of the use of pre-sampled sequences in the creation of the music.

The actual control structure of digital instruments, which is most often MIDI-based, does not easily permit more continuous control of an instruments timbre, by its inherent note-onset structure. Whereas this is indeed an appropriate control structure in many situations, the lack of sound timbre control can be disadvantageous.

The continuous control of the timbre is often, in today’s computer-based music tools, replaced by short sequences that are concatenated to produce pleasing music, often with standard rhythmic structure. While this method shows the need for the easy production of music with standard orchestration and rhythm, it also pin-points the popularity of such sequences in today’s music. As for the continuous control of timbre versus note-onset control structure, the automatic approach will become more popular, and research into appropriate interaction modalities is necessary to permit the easy manipulation and creation of sounds and music sequences in real-time.

Music synthesis has developed a great deal since the advents of the first FM synthesis methods [1]. While the gain in instrument sound fidelity is unquestioned, the use of pre-recorded samples often prevents an easy real-time transformation of the timbre. In contrast, more parametric models, for instance the additive synthesis method with appropriate high-level parameterization, such as the timbre model [3] can permit vast changes easily. With the novel parameterized synthesis methods, there is a need for appropriate, flexible and intuitive interfaces.

While the advent of flexible music programs widen the scope of musical sounds, it also limits the musical interaction in many situations. The performance of a musical sound is a vital step in music production by the addition of crucial timing and dynamics [8]. Even more so, the simultaneous performance of several instruments additionally contains complex synchronisation issues that both heightens the music quality, but that can also potentially alter the music in new directions.

2. SOUND MODELS
The interface is expected to have several, if not many sounds playing at the same time. This, in combination with the interface algorithms, put a heavy load on the host computer. For this reason, two relatively cheap, albeit powerful algorithms are used to produce the sounds that constitute the music; one is a traditional harmonic synthesis with perceptual relevance, the other an unvoiced synthesis method that creates Geiger or cymbal sounds and anything in between. In addition, a high-level additive model, the timbre model, is presented first.

2.1 The Timbre Model
A particular implementation of the additive synthesis, the Timbre Model [3], is chosen as the sound model first implemented in this work. In this model, a number of sinusoids with quasi-harmonic frequencies and time-varying amplitudes are made slightly irregular by adding band-limited noise on the frequencies or amplitudes of the sinusoids. The amplitude and spectral envelope are not defined in this work, as they are controlled by the performer. Given an amplitude envelope, the segments are found using the derivatives of a smoothed envelope [4], or by identifying the first 90% of the maximum as the end of attack, and
the last 70% of the maximum as the start of release. Only the
decay/sustain part of the sound is scaled when controlling the
synthesis. The sustain part is not affected, while the decay
amplitude is decreased by a given dB value per second. The
structure of the timbre synthesis model can be seen in fig. 1.

\[
B = \frac{T_b}{T_p - 1}
\]

Thus, by a simple formula, a sound is created that allows one to
to control the amplitude (linked to the perceptual loudness),
fundamental frequency (pitch) and length of the sound. In
addition, the principal timbre attribute, the spectral centroid
(brightness) is also controlled. By putting an envelope [4] on the
amplitude, other important timbre attributes, such as the attack
time, decay rate, etc. are easily created. Only the decay/sustain
part of the sound is time-scaled when controlling the synthesis.
The sustain values are not affected, while the decay amplitude is
decreased by a given dB value per second. Finally, by adding a
band-limited noise on the amplitude and fundamental frequency,
several additional timbre attributes are controlled, having to do
with irregularity, noise component, etc. Noise on the amplitude is
called shimmer, and noise on the frequency, jitter. The synthesis
method has much in common with the timbre model [3], although
the different timbre attributes are only controllable for all
harmonics together, not individually.

2.3 Atomic Noise
The atomic noise [5] is a synthesis method for unvoiced sounds
that can create a large variety of sounds with subtle variations. By
adding a variable number of atoms with random center frequency
and width, different noise categories are created, including Geiger
(ticking noise) and cymbal (inharmonic sinusoids). The formula
for one atom is,

\[
a(t) = a_0 \cos(2\pi \omega_0 t)e^{-\sigma^2 (t-t_0)^2}, \text{if } r > p,
\]

where \(a_0\) is the amplitude, \(\omega_0\) is the center frequency, \(t_0\) is the
center time, and \(\sigma\) is the standard deviation (width) of the atom.
The atom is realized if the random variable \(r\) is greater than the
probability threshold \(p\). The atomic noise is then the sum of a
large number of atoms. All of the parameters are random
variables with uniform or Gaussian distribution. A recent addition
to this model is the loudness dependent inverse auditory filter that
creates a perceptual white sound.

The distribution parameter that is controlled is the range, the
maximum value in the uniform case, or the standard deviation in
the Gaussian case. The amplitude range is not changed, as this
would change only the perceived loudness. It is possible to
change the distribution of the frequency and time to obtain, for
instance, a voiced sound, by making a periodic probability density
function of the time or frequency random variable [5]. What are
controlled are the probability \(p\), the width of the atom \(\sigma\), the
auditory system loudness dependent inverse filter and the
frequency and time periodicity and period. By setting a low
probability, very few atoms are realized, creating either a Geiger
sound, if \(\sigma\) is small, or a cymbal sound, if \(\sigma\) is large. By increasing the
probability, more dense sounds are created, resulting finally in a
white noise, if the combined \(p\) and \(\sigma\) values are large enough.

3. MULTIPLE GESTURES INTERFACE
The interface used in this work is the multiple musical gesture
interface [6]. In this interface, which can use any pointing device,
a pointer is entering a valid square from different angles, thereby controlling either sound or note parameters. It is possible to profoundly modify the sound in the same interface as the notes are played.

The actual control is dependent on the sound and note models. In [6], the sound model, the timbre model [3], has individual control of amplitude, frequency and irregularity on each partial. The note model was a standard note-onset model, with dynamics and vibrato control.

The gestures in the multiple musical gestures interface are the sound, note and timbre gestures. This is illustrated in figure 1. If the pointer enters the valid square from the left or right, the sound is gestured, and if it enters from the top, the note is gestured. If the pointer enters from below, the end-of-sequence gesture is made.

Figure 1. The multiple musical gesture interface. A pointer is controlling either the identity of the sound, or the note sequence.

3.1 Brightness control function
The brightness control function [3] (BCF) is rather similar to the timbre model, although significantly easier and cheaper to synthesize. The sound control, as illustrated in figure 1 (left incidence direction), is also valid for the BCF. As for the timbre model, the irregularity is also added in the BCF control by reversing the pointer direction during the sound gesture, as shown in figure 2.

Contrary to the timbre model control, the BCF irregularity control has no frequency location control. Instead, both shimmer and jitter are added on the time location corresponding to the distance in the direction of the x axis of the reversed gesture.

The note control is the same for the BCF as for the timbre model control [6]. A new note is initiated when the pointer is entering the valid square from the top. The note is sounded, assuming a valid sustain/decay segment has been identified, as long as the pointer is inside the valid square. A vibrato is added, depending on the undulations in the x axis direction during the downward motion.

Figure 2. Irregularity is added by reversing the sound gesture direction. Shimmer is affected when gesturing the temporal envelope, and jitter when gesturing the spectral envelope.

The timbre gestures are made when the pointer enters and leaves inside the valid square, as illustrated in figure 3.

Figure 3. The timbre gestures for the BCF.

The pointer has to enter inside the valid square. When this is done, it alters the parameters of the 'timbre' graphs it enters. When the pointer exits, the value at the extinction point is retained. As an example figure 3 shows a timbre gesture in which the brightness is modified. The 'timbre' graphs are Spectra, in which the brightness value and dynamic sensitivity [9] are controlled, the irregularity and noise graphs, in which the low-frequency and high-frequency irregularities are controlled, and the voice graph, in which a formant filter F1 and F2 frequencies are controlled.

3.2 Atomic synthesis
Atomic noise [5] is in no way similar to a note-oriented synthesis. It has no pitch, no amplitude envelope, and no spectral envelope. A recent extension, the perceptual atomic noise, changes the uniform frequency distribution and uniform amplitude values into a frequency distribution that is supposed to give approximately the same number of atoms for each critical band and amplitudes that should give the same perceptual level for all frequencies, approximately inversing the filtering made by the auditory system. Since this filtering is dependent on the intensity, the perceptual atomic noise dynamic is easily controlled.
devices are not made to function independently and they also
device has yet to be defined, though. Ordinary computer pointing
different types of sound synthesis methods. The actual pointing
The multiple musical gesture interface has been shown to control
simultaneously to another controlling the sound parameters.
performer adds one sound/music, and the collaboration takes
understood here primarily in a traditional way, where each
performer controlling the note sequence simultaneously in the same
interface, or to manipulate several aspects of one stream concurrently. This complicity is believed to enable a convivial musical activity.

4. SOCIAL SYNTHESIS
The goal of this work is to create a simple, intuitive and cost-effective interface for control over the synthesis and note sequences in a collaborative environment. The collaborative is understood here primarily in a traditional way, where each performer adds one sound/music, and the collaboration takes place in the creation of the resulting music, but it is also possible to have, for instance, one performer controlling the note sequence simultaneously to another controlling the sound parameters. The multiple musical gesture interface has been shown to control different types of sound synthesis methods. The actual pointing device has yet to be defined, though. Ordinary computer pointing devices are not made to function independently and they also
generally lack in freedom of extra-control movement, although they are efficient in navigation tasks [7]. In order to make a cheap interface that can control several synthesis methods at the same time, a camera device is utilized instead. By pointing a laser pointer at and across the imagined valid square, and having a camera detecting this movement, a simple and cost-effective method for controlling the pointing device is obtained. Although this approach does not give any force-feedback, it is believed that the combined auditory (music) and visual feedback remedies this. Finally, by using flashlights with different colors, several different synthesis methods are controlled simultaneously, without additional cost.

The actual camera detection algorithm and synthesis are implemented in max/msp and jitter\(^1\). The unfortunate latency of common usb cameras is somehow ignorable by supposing that the actual valid square top border is situated higher than it is in reality. While the full details of the different models are given in this paper, more work remains before all models are implemented and tested appropriately.

5. CONCLUSIONS
The multiple musical gestures interface contains the traditional note-onset metaphor, but permits additionally the in-detail shaping of the timbre of the musical sound, creation of musical sequences, and real-time control of many musically meaningful sound parameters.

Several sound models are presented, the timbre model, the harmonic brightness creation function, and the atomic noise. These sound models are chosen for timbre space is represents, the relatively cost-effective synthesis, the varied sound spectra produced and the possibility of real-time control of important timbre attributes. Each sound model is integrated into the multiple musical gesture interface by appropriate mapping of the sound parameters to the incidence angles of the valid square of the interface. This allows the performer to create a musical sound by short sound gestures, to produce a musical note by a note gesture, to produce a sequence of notes by a long musical gesture, or modify the sound in real-time by continuous timbre gestures.

The multiple musical gestures interface is a color-sensitive camera that tracks the movement of a laser pointer in real-time. This enables the performer(s) to produce several musical streams simultaneously in the same interface, or to manipulate several aspects of one stream concurrently. This complicity is believed to enable a convivial musical activity.

REFERENCES

\(^1\) Max/msp and jitter are available from http://www.cycling74.com/.

Figure 4. The atomic noise model has the same note/sound control as the BCF. Pitch is replaced with atom probability, however.
The sound and note control of the atomic noise model is the same as for the harmonic sounds, with the spectral and temporal envelope control to the left, and the note onset on top of the valid square. The note dynamics controls the auditory inverse filter, which gives an effect similar to many musical instruments, with an increase in brightness for higher intensities. The timbre gestures affect a different set of parameters than the BCF parameters, as illustrated in figure 5.

Figure 5. Atomic noise timbre graphs.
The ‘timbre’ graphs of the atomic noise affects the probability and width of the atoms, the periodicity weight and period of the time and frequency respectively and the repetition rate and acceleration rate, in case the noise is repeated. In figure 5, the atom width is increased, before the pointer exits the graph.


