SONIC WINDOW #1 [2011] – A Real Time Sonification

Andrea Vigani

Como Conservatory
Electronic Music Composition Department
Italy
anvig@libero.it

ABSTRACT

This is a real time audio installation in Max/MSP. It is a sonification of an abstract process: the writing on Twitter about music listening experiences on the web from people around the world. My purpose is not to sonify the effects of this process on a musical structure of the songs listened to, like a real-time-echo-web-mix or a new version of J. Cage "Imaginary landscape n°4", but to sonify the structure of the process itself, with its language transducers, its media and its rules. For this purpose, I created a musical instrument played by the data, like a wind chime, but here all the sounds are created by the web data itself, as if the material of a wind chime were the wind itself. It's like an open window on the web listeners where you can observe the action of listening and talking about music, but you don't hear the music listened to and you search for connections, reactions, interactions among the listeners, the transmission media and the code language.

1. DATA USED

Social Genius has created a web service: Twitter Music Trends, which listens to a vast selection of music-related tweets, and automatically tries to detect if each, at that moment, is discussing as a single musician or as a group http://twittermusictrends.com/latest.json (updated every 2 seconds). Information about Twitter music data and the latest artists can be identified from the Twitter stream and the latest 10 IDs of associated tweets.

2. LISTENERS - WRITERS

First of all, the listening process and the tweet process from twitter users; people listen to music and then write tweets about it: it's a human thought about listening to music expressed in a verbal language and syntax. People think, listen and interact with the process and the media with a GUI that translates an information flux. This translation is from a human thought(with its specific language and syntax) to a universal ASCII number code or numeric streams; characters are the same, but syntax changes (ASCII numbers are the common atoms [letters] among different languages) according to an internet code data: **language and syntax change, but information doesn't change.** (Figure 1.)

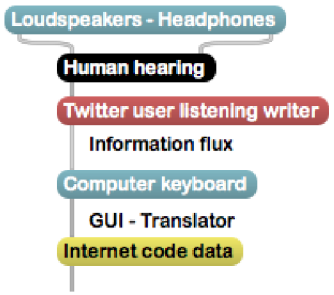


Figure 1.

3. INTERNET CODE DATA ANALYSIS

At this point of the process (that I want to sonify), there is a transduction of the language: the code data from twitter is analysed and the information flux changes: **language and syntax (code) are the same, but information changes:** information is about the process itself, not the original information thought and posted on the web by the twitter users, but a new thought about the first action: the **new information is always a consequence of the previous thoughts.** (Figure 2.)

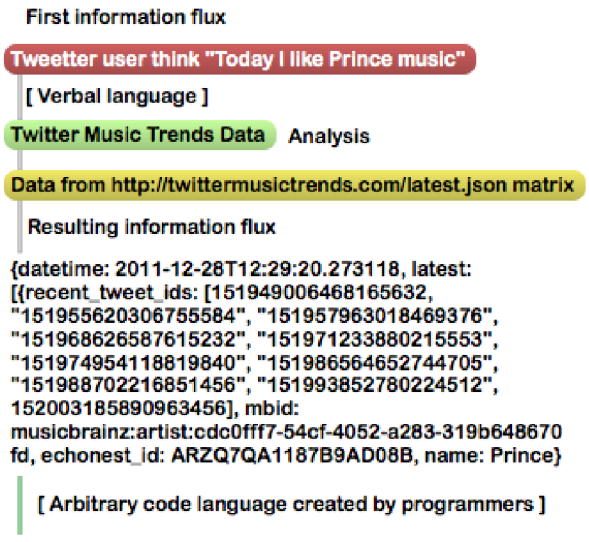


Figure 2.

4. INFORMATION USED

For this sonification I used only one kind of information: NAMES: 1) the Artist Name ; 2) the last 10 Twitter IDs that wrote about the artist (names translation in a code language). In this way, I have a list of 11 names in two different languages (spoken and codified) and these names are connected by a common thought in different ways: the 10 ID names write about the musical actions created by the artist name: names change but the process is always the same, like the musical language…these data becomes in different ways the sound itself and also the score.

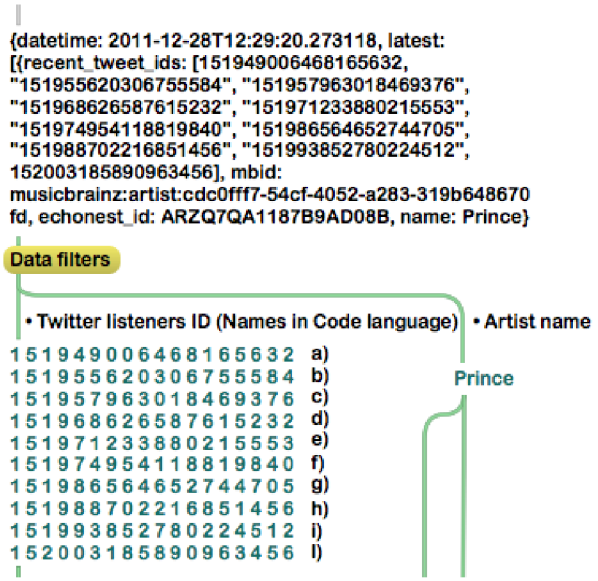


Figure 3.

5. WAVETABLE PLAYER – BACKGROUND NOISE

I used the "last" ten ID numbers scaled from -1 to 1 as amplitudes of a wave-table *(each ID = 18 numbers = 180 numbers * 5 (downsampling factor of 2) = 900 samples stored in the wave table) (*Figure 4). They are updated every 2 seconds, according to a choice of the Social Genius programmers and so I programmed a linear interpolation of ID values between the updated triggers, to simulate that the process is continuous.

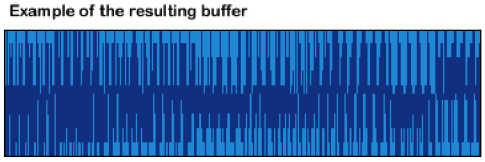


Figure 4.

The wave-table is then played back in a loop at a frequency that varies cyclically from 0.1 to 1.5 Hz, and it's a musical representation of the twitter code web rhythm (a background noise from a portion of the web) morphed by the twitter users almost in real time. At the end of the process, I use a cyclic stereo pan and a cyclic fade-in fade-out to give more sense of "web data waves", as if the web data were a living entity with its own cycles of life.

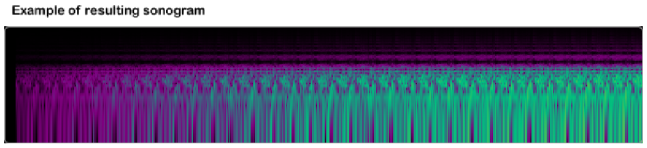


Figure 5: Listen to audio file "1-Background_noise.mp3"

6. SPEECH SYSTEM PLAYER

I use the Artist Name data in two different ways:

1) The Artist Name is translated by the Speech computer software (at each new name, the voice, which reads the name, changes randomly, depending on the computer speech software); then the speech signal passes into a granular synthesis module with a buffer of 10 seconds:

Twitter IDs control in real time:
- grain duration (Min/Max),
- rests between grains ((Min/Max-Voice numbers),
- grain amplitudes and
- grain pan-pot (MIDI)

In this way, the multitude of twitter users voices listening to the artists and also the translation process are represented; at the beginning of the process, the spoken words are translated in ASCII numbers and these numbers are the code "letters-phonenmes"; at that point, with a granular synthesis, I deconstruct the spoken languages (English, French, Italian, etc.) into phonemes (musical language).

Language conversions:

- thoughts (spoken language)→Words written on keyboard→ASCII code→ web code data
- web code data→ASCII code→Spoken language →Phonemes (musical language)

2) The previously obtained "twitter ID background noise" is then filtered by the "last artist name", as if the name could sculpt its profile in the noise: the noise passes into a bank with a maximum of 18 pass filters and frequencies of each filter are given by a conversion of ASCII numbers in frequencies.

Example:

Beatles =
66 101 97 116 108 101 115 (ASCII-Midi Pitches) =
369 2793 2217 6644 4186 2793 6271 Hz (Filter bank center frequencies)
The bandwidths of the filters are given by one of the twitter IDs (scaled from 0.1 to 4 Hz) that is listening to the Beatles:

Twitter IDs: 1 5 0 0 9 6 8 5 4 9 0 0 6 7 8 6 5 6
Bandwidths: 0.8 2.4 0.4 0.4 4. 2.8 3.6 2.4 2. 4. 0.4 0.4 2.8 3.2 3.6 2.8 2.4 2.8 Hz

Each Artist Name is updated every 2 seconds, so the timbre changes without an interpolation every 2 seconds like a "bell signal" and gives a regular beat to the time.

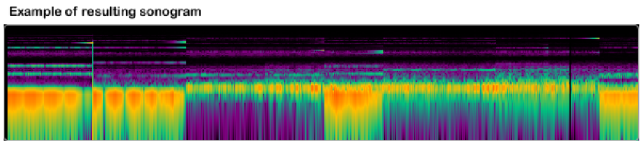


Figure 6: Listen to audio file "2-Timbre_name.mp3"
Listen to audio file "3-Voice_grain.mp3"

7. DATA GLITCHES

One of the last ID listeners gives a small amount of samples stored in a wave- table and played immediately; the amplitudes, which are not scaled and are from 0 to 9 , are afterwards clipped to 1 (wave-shaping) with a linear interpolation between samples. Then the signal is passed through a resonant bandpass filter with a central frequency set to 2000 Hz, bandwidth of 23 Hz and a resonant factor of 3; this gives a "percussive mallet" sound. A quartic envelope is applied to the signal, which has been extracted from the artist name, and the resulting signal enters in a variable delay with a feedback of 1%. This because "the latest artist" scrolls back in position on time... and 2 seconds later he is not ' the latest one' but it's always listened to on twitter; in this case, it doesn't disappear but becomes like an "aura", which gives this sense of slow down and fading, passing through a granular synthesis.

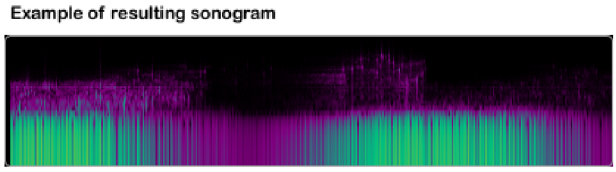


Figure 7: Listen to audio file "4-Data_glitches.mp3"

8. SINE WAVES OSCILLATOR BANK

The last sound generator is an additive synthesis with 18 partials (the number of numbers in a single Twitter ID ; 5 Twitter IDs are mapped according to:

- Frequencies of each partials
- Detuning factor of each partials
- Relative amplitudes of each partials
- Relative durations of each partials
- Relative attack times of each partials

As the IDS are from different people, I applied a granular synthesis to simulate the contemporary presence of 5 different people (the Ids), that are producing the same sound together.

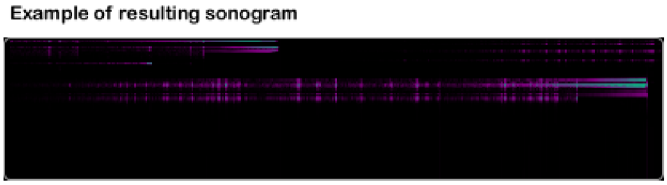


Figure 8: Listen to audio file "5-Oscilbank.mp3"

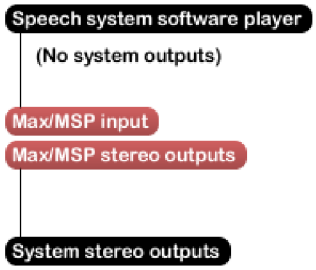
9. EQUIPMENTS AND DIFFUSION

1 Apple computer
1 Internet connection
1 or more Headphones or
1 Audio cart
1 Mixer console table
from 2 to 32 Loudspeakers

It is possible to listen to this audio installation from different computers and headphones or to diffuse the sound on several loudspeakers, to obtain a double interaction: on the other side of the web the listeners create the sounds and on this side other people diffuse this sound in a room and it may be that twitter users, who are present in the room, can change the sound itself…

10. TECHNICAL DETAILS

This software is a Max/MSP patch and you can launch it as an alone application or inside Max/MSP, according to externals used in the patch until now; it is possible to run it only on Apple computers. If you listen to it directly from your computer audio device, it is necessary to do an internal routing; in fact, audio from speech system player will not diffuse out directly, but only after being processed by Max/MSP.



It is possible to route it internally with the software "Sound flower" (from Cycling74 or "Jack") or externally with a sound card, which is present in the room and can change the sound itself…

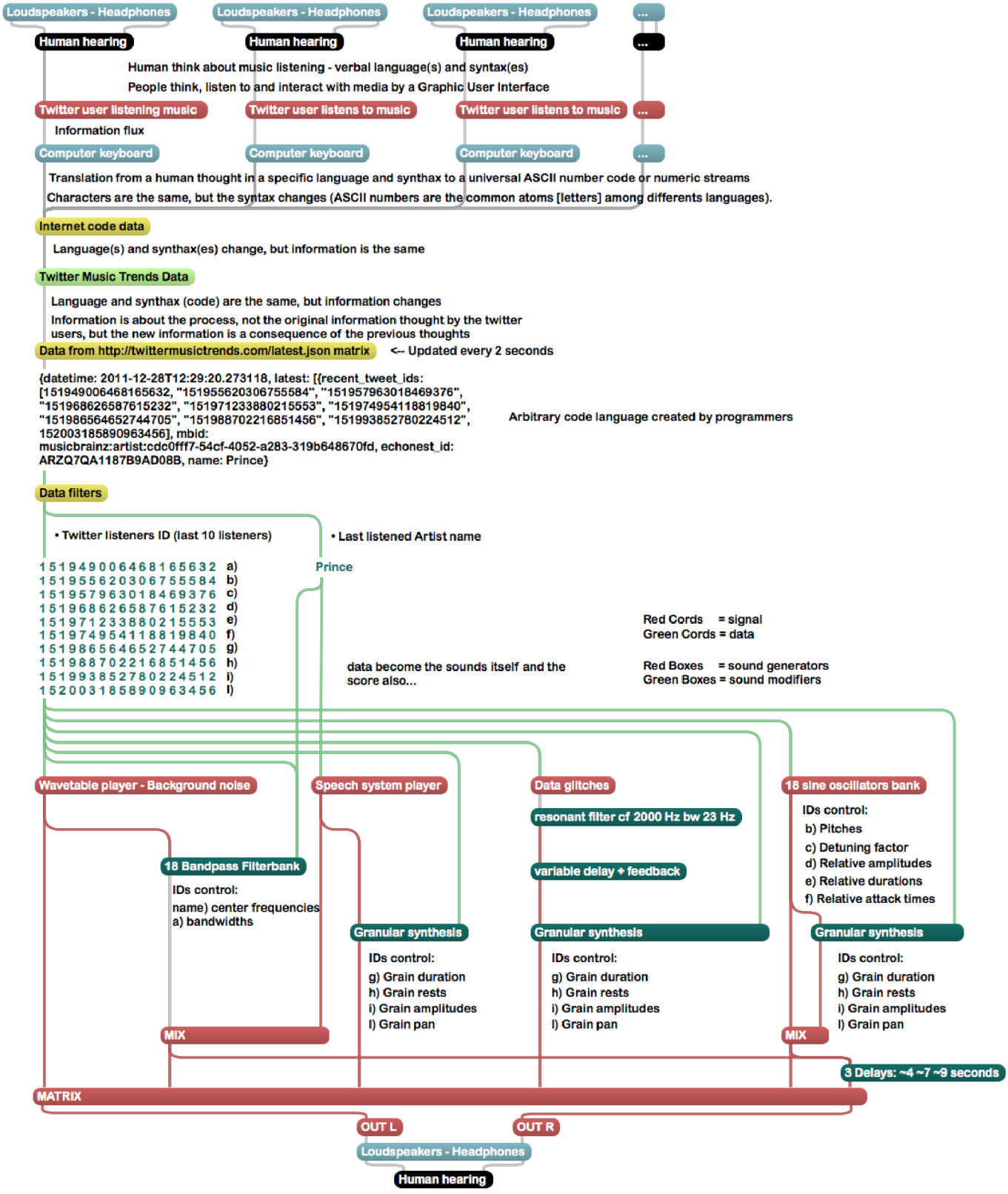


Figure 9: Main Block Diagram