Live-Coding-DJing with Mixxx and SuperCollider

Antonio José Homsi GOULART
Computer Music Research Group
Instituto de Matemática e Estatística - Universidade de São Paulo
Rua do Matão, 1010 - Cidade Universitária - 05508-090 - São Paulo, SP - Brazil
ag@ime.usp.br

Abstract

This paper suggests a modified dance music DJ performance, based on common DJing techniques enriched with live coding moments either mixed with records or not, instead of only reproducing previous-made tracks. That way, all the different possibilities offered by live coding are put together with commercial tracks, promoting live coding while maintaining the dance music atmosphere and opening more improvisation possibilities for the DJ. It is also a funny way to start learning programming and live coding. All software involved are open-source and the workflow is based on the author's. The primary intention here is to stimulate DJs to try live coding, at the same time helping to promote its sonority to audiences other than experimental music enthusiasts.

Keywords

live coding, DJing, live performance, education

1 Introduction

A DJ work consists of researching a lot to find the most suitable tracks for a specific venue and selecting a good order to play them as a set. Also important is the ability to mix, which consists of beat-matching 2 tracks and make a transition from one to another, maintaining a continuous flow instead of separately starting each track [Broughton and Brewster, 2006] [Collins et al., 2013]. Mixxx[1] [Andersen, 2003] is an open-source digital DJing software suited for that.

Another type of live music performance is live coding, also called on-the-fly programming, in which the programmer/performer augments or modifies code while it is running and generating real-time sound, without the need to stop or restart the program [Wang and Cook, 2004]. A lot of languages are suited for this task, and one of the most popular is the open-source SuperCollider[2] [McCartney, 2002].

[1] http://www.mixxx.org/

[2] http://supercollider.sourceforge.net/

In this paper a workflow based on a mixture of DJing and live coding will be described. I believe that documenting this process, which is a very simple one but presents some technicalities, might benefit or stimulate seasoned DJs to incorporate new tools to their sets. At the same time I hope that it helps promoting live coding at mainstream venues and open-source software to more users.

In sections 2 and 3 DJing and live coding practices will be detailed. Section 4 describes the live-coding-DJing method. Conclusions and perspectives are analyzed in section 5.

2 DJing

The term DJ comes from Disk Jockey, referring to the act of playing vinyl records for an audience. In a typical configuration, a DJ would work with two turntables and a mixer, so two songs could be reproduced simultaneously, an important issue to make the transitions.

When CDs became available more and more people switched to the smaller and cheaper discs and CD Decks, and nowadays many DJs work only with a laptop computer. "(...) If they lugged large boxes of records with them in the 1990s, after the millennium a tendency was evident toward lighter-weight luggage allowed by fully digital track management in hard-drive disk jockeying" [Collins et al., 2013].

No matter which media is chosen the DJ task is the same: playing records one after the other, mixing them, which consists of beat-matching (getting the tracks to play at the same tempo, with their beats synchronized [Broughton and Brewster, 2003]) and then making the transition, which can be a blend (a gradual fade out of the first track and fade in of the second) or a cut (a sudden change of track being played).

The beat-matching stage is typically done with a headphone: while the current song is reproduced for the audience, the next one is reproduced on the DJ's headphones. Then the

tempo might be adjusted and after that the beats synchronized. Instead of using the headphones, one could beat-match by looking at the tracks' waveforms, assisted by tempo and beat detection software. There is also the more extreme possibility of doing it by ear dispensing headphones aid, mixing while both songs are reproduced to the audience.

Some techniques [Broughton and Brewster, 2003] might make the blend more interesting:

- a simple blend using the faders: fade one track out and the other one in (fade durations depend on the genres);
- matching phrases: dance music tracks are divided in 4-bar phrases. Usually there are clues at the end of these phrases, e.g. a cymbal crash, extra drum beats or an instrument finishing a solo. When mixing, it is important to match phrases and beats;
- take advantage of keys, and avoid key clashes: some tracks sound bad when mixed because their notes are not in the same key. Some alternatives in this situation are mixing when one (or both) is just percussion or pitch-shift one of them;
- equalization can be used to hide parts of a song while keeping others: one example is removing the bass line from a track and introducing the bass line of a new song, then do the same with the other parts;
- matching rhythms: some tracks fit together better than others, because besides their harmonies and melodies match, their rhythms fit perfectly together. Difficulties arise when mixing two tracks with syncopation, or too many drumbeats.

Instead of blending a track with the subsequent one a DJ may cut, which is switching sharply from one record to another without losing the beat. Cuts tend to sound better with sparse, percussive music, and bad with tracks containing continuous melodies [Broughton and Brewster, 2003]. Other alternatives are doing stops (stop current track and then start the next record after a while) or spin-backs (reverse current track and then cut to the next one).

Some techniques were discussed but we shall keep in mind that, as emphasized by Broughton and Brewster, the most important thing about DJing is choosing the records and the order to play them, and after that the crucial decision is where to put the joins. "Where the mix occurs is more important than how it occurs" [Broughton and Brewster, 2003].

A DJ software (or any task-specific application) ties the user to its paradigms. All actions, control structures, and interaction possibilities are defined in advance. With a rigid interface, they offer high visibility of available operations and immediate gestural control for the live performer to adapt sound immediately and continuously, but with reduced potential for creative exploration [Blackwell and Collins, 2005].

Mixxx's interface (Figure 1)[3], for example, contains buttons for loading tracks in 2 different virtual decks, playing, stopping and looping them, setting cue points (important points in the track, likely to be replayed), knobs and faders for adjusting effects parameters, filters, tempo and playback rate for each track, among other functionalities, including a waveform visualizer for the loaded tracks.

Figure 1: Mixxx interface (Late Night Blues).

3 Live-coding

On the other hand comes the possibility of working with interpreted computer languages, modifying running algorithms on-the-fly and, in the case of audio programming, getting real-time sound as the program's output. Brown [Brown, 2006] defines live coding as a practice where "digital content is created through computer programming as a performance".

The interface for live coding is only a text editor[4] and a shell for feedback (although fancy IDEs are available), which means that all the actions are hidden in text commands. All the exploratory possibilities of computer languages are available for the performer, but no inter-

[3] picture from http://www.mixxx.org/press/

[4] For text-based languages. Graphical programming languages, where programming is made linking objects in a canvas are also available, e.g. PureData.

face with buttons prepared for interaction are present. Therefore code must be written for all the actions involved in sound production, which can cause a high mental load specially in the scarce-time situation of a performance.

Blackwell and Collins argument that besides aesthetic reasons a key concern for someone to choose the challenge of live coding instead of using a pre-made software is that these "are biased towards fixed audio products in established stylistic modes, rather than experimental algorithmic music which requires the exploratory design possibilities of full programming languages" [Blackwell and Collins, 2005].

Different approaches can be taken in a live coding session, from a low level and mathematical one, writing complex algorithms that output sound as they evolve, to a more high level and simple approach, describing synthesis models and then sequencing sounds by specifying values for the models' parameters. These values can result from routines evaluation or be directly chosen by the user. When working with this instrument plus sequencer paradigm, code can be viewed as a description of instruments and a score (a scheduling of musical events).

This last approach represents in my opinion the easiest way to live code, provided the artist can describe a synthesis model, even if a simple one. Programming notes for melodies/harmonies and rhythm patterns is simple, so it can perfectly be combined with dance music. This might be a naive approach for live coding, but it is enough to synthesize sounds within the LCDJ paradigm to be proposed.

3.1 The performance

In live coding performances usually the code is projected for the audience. According to Brown, most people like it but some people find it a display of virtuosity and distraction to the listening experience [Brown, 2006]. Alex McLean, who plays raves and programmers gatherings said "I prefer it when the audience is dancing and doesn't care how we're making the music" [Andrews, 2006].

A live coding session may start either from scratch or with previous written code. In this last case, the artist may augment, modify or fill blanks. As time is scarce in a performance an interesting trick is to prepare snippets of code with the basic structure and syntax of the language (or even snippets with the main sounds and patterns of a personal production).

An alternative approach for live coding is doing it in duos or larger groups, or even in an orchestra together with musicians playing acoustic instruments. That takes the pressure off a single performer and, in the case of a live coding group, shares the algorithmic complexity between the members [Collins, 2011].

4 Why not doing both? LCDJ!

If someone is neither restricted to perform like a DJ nor like a live coder, and if someone can access tools for both occupations within the same computer, why not playing DJ styles of music adding some live coding moments, or why not live code supported by nice tracks playing along? Why not doing it if both tools are running on the same sound server?

In an interview, dance music duo Coldcut said "The future of DJing is not about whether vinyl will survive. The future of DJing is about media mixing. The DJ with two SL1200s [turntables] will fade out, but if he is clever, he'll evolve into a multi-armed posse manipulating various sound and vision sources. There should be a new name for this, maybe a media-jockey" [Broughton and Brewster, 2003].

4.1 Suggestions for LCDJing

A LCDJ may start the performance playing a record or coding (releasing initial sounds after a while). In the second case the mood can be set according to improvisational decisions made exactly at the time of the performance, so a sound that perfectly invokes the intended atmosphere can be synthesized, instead of having to pick one from the finite set that is the hard drive.

A LCDJ is able to jam with records via live code, inventing new sounds or mimicking/emphasizing/satirizing the record's. Stopping the track for a solo is also a good move. In the case of playing a personal production, different versions of it can be improvised by modifying code used to generate it (and that's a good appeal to produce music using code); that can also be a way to tease the audience revealing pieces of the upcoming track, an effect similar to cutting back and forth between two beat-matched tracks [Broughton and Brewster, 2003].

Another interesting option is routing audio from Mixxx to SuperCollider, processing tracks with infinite possibilities of effects, instead of only applying some high-pass or low-pass filters or common effects like flanging or ring modulation (these are the options available in Mixxx's and other similar programs interfaces).

In my experience, mixing is where LCDJ true potential is revealed. Instead of blending or cutting like a DJ, a live-coding-DJ may live code between tracks. A simple guide for that can be:

1. choose the next song to play in the set and load it in Mixxx;
2. choose an instant in the current song to start interacting;
3. start live-coding and interact (in any way) with the current track;
4. when current track ends, take the live-coding to a sonority suited to welcome the subsequent track;
5. choose a moment and start the new track;
6. when it is suited, stop live coding and let the new track fly.

Step 4 can be made at any pace and some ways to welcome a track are by:

- invoking its rhythm;
- invoking its bass line, melody or harmony;
- making a sparse and percussive sound, preparing for a cut;
- making a totally non-sense sonority that brings tension to be released with next track (perfect for tracks with a sweet and melodic intro).

Of course there is always the option of not welcoming a track and live code for hours.

The blending techniques mentioned earlier can be adapted for LCDJing. Some suggestions:

- matching phrases: mimic a bass line or melody of current song and keep playing it for a while until the song vanishes. Then adapt it to an element present in next track, start it and interact for a while;
- keys: start coding with current track, at the same key. When it ends, progressively add notes from another key, but without clashing. When the sonority has been taken to the same key as the next track, all is set for a good entrance;
- equalization: a great chance to modify tracks. Cut some parts of the current track, for example, the bass line, and live code a new one. When suited, introduce a sonority that resembles next track and call it;
- match rhythms: be the drummer, along the current track, then solo, then with the new one. Link tracks using percussive lines.

The techniques presented are the ones I could come up with and test in some occasions, but there is no limit for the possibilities in LCDJing, besides what one can do with code. Feeling the audience, the mood and the venue style are important clues for how far from mainstream a LCDJ can go in a session.

4.2 Software involved

There are lots of nice applications for DJing[5] and languages suited for live coding[6]. Depending on personal choices any combination of software can be used for LCDJing. One could even dispense DJ software and LCDJ using only a language. In my experience Mixxx and SuperCollider is a good pair for Live-Coding-DJing performances because:

- both are very efficient, so LCDJing is possible even with a NetBook;
- Mixxx is very easy to learn and provides all the tools a DJ need[7], so common DJ actions can be readily done instead of having to code them;
- although a first contact with SuperCollider might be frightening, its syntax makes it easy and fast to code synth models and sequence patterns (all we need to LCDJ);
- both connect to Jack[8], which allows audio routing between software, so Mixxx and SuperCollider can communicate, sending and receiving audio to and from each other. Some advantages and possibilities have already been discussed;
- both are open-source, with all the related advantages.

4.3 Simple Mixxx, to collide with SuperCollider

Both newcomers and artists used to other DJ applications will find it intuitive to work with Mixxx. Its interface is really simple and everything necessary for LCDJ is available as a shortcut in the computer keyboard. A quick

[5] http://linux-sound.org/ddj.html
[6] http://toplap.org/wiki/ToplapSystems
[7] http://www.mixxx.org/features/
[8] http://jackaudio.org/

read in the wiki[9] and one is familiarized. The community forums[10] are also good resources.

In the author's opinion a good practice for DJing, specially LCDJing, is to avoid mouse (time is scarce) and external controllers (save money and space in the cabin/backpack and make use of the laptop hardware as a whole).

4.4 Simple SuperCollider, to mix with Mixxx

As it was highlighted above, the live coding strategy proposed here is very simple; only instrument definitions and a way to sequence sounds are necessary. For the instruments definitions the class *SynthDef* is used. Its syntax is shown in Figure 2, with a simple sawtooth oscillator and an ADSR envelope being defined.

```
SynthDef(\saw_ex, {
    |out=0, gate=1, pan=0, freq=100,
     a=0.1, d=0.2, s=0.8, r=0.3,//env args
     mix=0.5, room=0.8| //reverb arguments
    var env = EnvGen.kr(Env.adsr(a,d,s,r),
                  doneAction:2, gate:gate);
    var snd = Saw.ar(freq);
    snd = snd * env;
    snd = Pan2.ar(snd, pan);
    snd = FreeVerb.ar(snd, mix, room);
    Out.ar(out, snd);
}).add;
```

Figure 2: Defining an instrument.

The classes *Pdef* and *Pbind* might be used for the sequencing of sounds. The syntax is presented in Figure 3, with a pattern that repeats the notes D,F,A (the degrees 1,3,5 are converted intro *freq* values) sequentially, along with values for each note duration and panning. *Pseq* picks the argument vector values in a sequence.

Notice that each note attack time and reverb mix (dry=0/wet=1) will have a random value (*Prand* randomly picks values from the argument vector), so the instrument sound will always be changing. Models with more parameters can offer a wide timbre variation.

The equivalent of this 30-line simple and flexible implementation would hardly (if even possible) be attained in more rigid interfaces. Other options for sequencing and more complex synthesis models (and much more information) can be found in the learning SuperCollider page[11] and SC community[12].

[9] http://www.mixxx.org/wiki/doku.php
[10] http://www.mixxx.org/forums/index.php
[11] http://supercollider.sourceforge.net/learning/
[12] http://supercollider.sourceforge.net/community/

```
Pdef(\play_saw,
    Pbind(
      \instrument, \saw_ex,
      \degree, Pseq( [ 1,3,5 ] , inf),
      \dur, Pseq( [ 1,1,2 ] , inf),
      \a, Prand(0.1*[5,11,22,33], inf),
      \d, 0.5, \s, 0.95, \r, 0.15,
      \pan, Pseq([-1,1,-0.5,0.5], inf),
      \mix, Prand([0.2,0.5,0.9], inf),
      \room, 0.5
    )
).play;
```

Figure 3: Specifying parameters values and sequencing sounds.

5 Conclusions

DJing is a long-time established practice and live coding a not so old one but it certainly has already established its importance in contemporary music practice. The workflow described in this paper does not intend to dismiss such practices, only mix them in a way that is simple for the seasoned DJ or anyone to try live coding and benefit. At the same time it is a funny and stimulating way to start programming, dive into synthesis studies and learn more about open-source software. Surely it only opens new possibilities for the artist.

Describing synthesis models, although a difficulty task at first, is definitely worth. The sound palette of the producer will grow to the point that besides having personal productions/tracks, a characteristic sound can also be achieved. In a world where most commercial electronic dance music sound so alike producing tracks and timbres is a good way for promotion. Sharing snippets of code with nice sounds and/or patterns also seems to be a good idea.

A pure live coding session aiming experimental music requires much more than only these simple concepts presented here. With this approach, however, a good level of interaction with dance music is possible because of its structure, which is usually rhythmic and well defined, with distinctive melodies and harmonies.

Whatever the genre of electronic music the DJ wants to play, interaction with live coding is possible - from abstract Ambient sounds to the rhythmic beats of mainstream House - even with the simple paradigm described in this paper. Synthesis models can be as varied as the creativity/ability of the artist; the instruments can be sequenced with a fixed or (widely) vary-

ing timbre; the rhythm and notes patterns can also be freely specified, even randomly (Why not going for a track that wasn't intended, just because a random pattern resembled it?).

Although studio productions are not the intended output of a LCDJ session, extra care must be taken with the rawness of the synthesized audio. Mainstream dance music records are equalized, pre-mixed, well-balanced, compressed and mastered, so in order to fit in new sounds some sculpting is necessary, otherwise they get the foreground and mask the record. Usually, extra equalizing in the record plus a little reverb and good positioning with panning (that's why they are in the example) in the live coding sounds are enough to find them a spot and prevent clashes.

A true improvising door opens with LCDJ. Although DJs know specific tracks to invoke different types of emotions, and DJing is based on improvisation according to the audience mood, the set of possibilities is finite, unless music is created on-the-fly.

The same way that a performance in group relieves part of the pressure on each artist, live coding along records also has the same effect. More time is available to analyze and shape sounds, impose a rhythm and write code.

Screen projection, although explored in pure live coding sessions, may be discarded in LCDJ. Code may be too simple, it would be a distraction for dancers and a spoiler for the set. However it depends on the venue, as more advanced programmers and specific audiences might like.

Of course the practice is not restricted to Mixxx and SuperCollider. Great software and languages are available for DJing (xwax, terminatorX, etc.) and live coding (ChucK, PureData, etc.). However, Mixxx's interface might be more familiar for seasoned DJs, especially those who work with turntables/decks or OSs other than Linux, and SuperCollider efficiency, along with Patterns Library - easy to learn and use - makes it a good option to start.

LCDJing would also be possible dispensing the DJ software and using only a live coding language. However, a DJ application facilitates performing common DJ tasks (creation/management of a playlist in the performance, adjusting tempo with a knob twist, cueing points in tracks and scratching), relieving the mental load that coding every move would create.

My impression on playing as a LCDJ is that people accept rhythmic live coding moments as unknown yet good track passages, when appropriately presented and not overdone. More abstract coding moments brings tension and curiosity, which calls that magical record.

6 Acknowledgements

I would like to thank CAPES for financial support and Flávio Schiavoni for suggestions on this work, and I wish that those who haven't tried live coding yet feel stimulated to do it, specially children learning music, music producers and DJs. Starting LCDJ in duos is also great!

References

Tue Haste Andersen. 2003. Mixxx: Towards novel dj interfaces. In *Proceedings of the 2003 Conference on New Interfaces for Musical Expression*, NIME '03, pages 30–35, Singapore, Singapore. National University of Singapore.

Robert Andrews. 2006. Real djs code live. *Wired, technology news.* Retrieved 01/20/2014 from ≪ http://www.wired.com/science/discoveries/news/2006/07/71248 ≫.

Alan Blackwell and Nick Collins. 2005. The programming language as a musical instrument. In *Proceedings of Psychology of Programming Interest Group*, pages 120–130.

Frank Broughton and Bill Brewster. 2003. *How to DJ right: The art and science of playing records.* Grove Press, New York.

Frank Broughton and Bill Brewster. 2006. *Last night a DJ saved my life.* Headline Book Publishing, London.

Andrew R. Brown. 2006. Code jamming. *M/C: a journal of media and culture*, 9(6), December. Retrieved 01/19/2014 from ≪ http://journal.media-culture.org.au/0612/03-brown.php ≫.

N. Collins, M. Schedel, and S. Wilson. 2013. *Electronic Music.* Cambridge Introductions to Music. Cambridge University Press.

Nick Collins. 2011. Live Coding of Consequence. *Leonardo*, 44(3):207–211.

James McCartney. 2002. Rethinking the computer music language: Supercollider. *Comput. Music J.*, 26(4):61–68, December.

Ge Wang and Perry R. Cook. 2004. On-the-fly programming: Using code as an expressive musical instrument. In *Proccedings of the International Conference on New Interfaces For Musical Expression*, pages 138–143.