The development of the digital technologies linked to the performance and the composition of electronic music

Use of SuperCollider in a live performance with assisted composition

Giovanni Onorato
Supervisor: Damiano Meacci

Introduction

- Investigate about the use of digital technologies in musical contexts
- Project: developing of an algorithm capable to generate and perform a composition within established parameters, with whom a performer can improvise

Technical and artistic aspects

- Tools: use SuperCollider for developing; ‘sound making objects’ for performing

- Materials: synthetic – glitches; real – field recordings

Historical background

- From CSIRAC and Ferranti Mark I to the invention of Digital-to-Analog Conversion module (DAC)
- Bell Labs, Max Mathews; Music-N e Csound
- “The MAX paradigm”, Miller Puckette at Giuseppe Di Giugno’s IRCAM studios
- Pyrite, J. McCartney writes SuperCollider, an OOP SmallTalk-like language

Project

- **Compositional aspects:** *Accumulazione* **S. Sciarrino and** *Sintagmi* **F. Giomi**
- **Technical aspects: Ableton, M4L and Jack audio**
- **Why using SC then?**
- *Affordances* **and** *Hacking*

Code

- From the global structure to specific functions
- SynthDefs: blueprints for DSP
- Scheduling and the SystemClock object
- Sonic events and Multichannel Expansion
- Patterns: a language inside a language

From the global structure to specific functions

```
//server config [ … ]
    o = Server.default.options; o.inDevice_("JackRouter");
//global var
    ~tempo;~scale;~bufPath; ~times;
//global func
    ~chrono; ~arrFunc; ~times_N; ~stops_N; ~cleanup; ~makeBusses; ~makeNodes; ~makeBuffers;
    ~texture_1 = {}; ~scapes; etc …
//register funcs
     ServerBoot.add(~makeBuffers); [ … ]
//boot
     s.waitForBoot({
         SynthDef.new(\vOsc, {}).add; [ … ]
         s.sync;
         ServerTree.add(~makeNodes);
         [~times_1, ~stops_1].flop.do { |pair|
               ~texture_1.value(pair[0], pair[1])};
          ~chrono.value
      });
```

SynthDefs: blueprints for DSP

```
SynthDef.new(\rhythmBuf, {
   arg atk=0.00001, sus=1, rel=0.2,
   buf=0, rate=1,
   del = 0, delOut = 0,
   amp=1, out=0, pan=0;
   var sig, env;
   env = EnvGen.kr(Env([0,1,1,0],[atk,sus,rel],), doneAction:2);
   sig = PlayBuf.ar(1, buf, rate*BufRateScale.ir(buf));
   sig = sig - OnePole.ar(sig, exp(-2pi * (50 * SampleDur.ir)));
   sig = sig*env*amp;
   sig = Pan2.ar(sig, pan);
   Out.ar(out,sig);
   Out.ar(delOut, sig * del);
}).add;
```

Scheduling and the SystemClock object

```
~texture_1 = { |start, stop|
    var synth, scale, chords;
    scale = ~scale.choose; chords = 16.collect{[ … ].clip(1, 18000)};
    SystemClock.sched(start, {
        synth = Array.fill(16, {
            Synth.new(\vOsc, [
                \freq, chords.choose,
                \amp, rrand(0.01, 0.25),
                [ … ]
                \rev, rrand(0, 0.99),
                \del, rrand(0, 0.2),
            ], ~srcGrp)});
    });
    SystemClock.sched(stop, {
        synth.do(_.set(\gate, 0));
    });
};
```

Sonic events and Multichannel Expansion

```
bpSynth = Array.fill(8, {Synth(\BPF, [\in, ~bpBus, \out, ~scapeBus,
    \freq, exprand(120, 18000),[ … ]
], ~fxGrp, \addToHead)});
scapestrato = Array.fill(12, {
    Synth.new(\fieldBuf, [
        \rate, if(rrand(0.75, 1).coin, exprand(0.85, 1.5), rrand (0.1, 5)),
        \amp, if(rrand(0.15, 1).coin, rrand(0.5, 1), 0),
        \dly, rrand(0.1, 15), \atk, exprand(15, 29.0),
        \dcy, rrand(5, 20.5), \sus, rrand(0.1, 0.6),
        \rel, if(rrand(0, 0.01).coin, 0, rrand(10, 20.5)),
        \crv, rrand(2, 9.0) * [-1,1].choose,
        \buf, samples.choose,
        \out, ~bpBus,
        \del, rrand(0, 0.5),
        \delOut, ~delBus
    ], ~bufGrp)});
```

Patterns: a language inside a language

```
SystemClock.sched(start+(rrand(4,11)), {
    sines = Pbind (
        \instrument, \rhythmBuf,
        \dur, Prand([
            Pwrand([1/4, Rest(1/4)], [0.01, 0.09].normalizeSum, inf),
            Pwrand([1/4, Rest(1/4)], [0.9, 0.1].normalizeSum, inf)
        ], inf),
        \amp, Pwhite(0.005, 0.3, inf),
        \buf, ~sinesDir[[0,1].choose],
        \rate, Pwhite(0.0001, 13, inf),
        \pan, Pwhite(-1.0, 1.0, inf),
        \out, ~bufBus,
        \group, ~bufGrp,
        \del, Prand([0, Pexprand(0.0001, 0.5, inf)], inf),
        \delOut, ~delBus
    ).play(~tempo, quant:1);
});
```

Conclusions

- This heterogeneity of the existent digital tools allows the end users – or musicians – to build and use ad-hoc devices for every function or part of their project.
- SuperCollider is a useful and rich environment for defining a lot of musical and sonic algorithms
- Ableton Live and the Max paradigm are also valid tools for a straightforward and a stable development of musical ideas

In my project Supercollider, Ableton and M4L are cooperating for the actual realization of the musical idea: the processing of the electric bass signal is handled with Ableton and Max, while SuperCollider is implemented to overcome the limitations of a system like Ableton Live.

The Game

Thanks

The Game