SuperCollider in Athens [pre00]

Yorgos Diapoulis [ydiapoulis@gmail.com]

SuperCollider [1] is an environment and programming language for real time audio synthesis and algorithmic composition. It provides an interpreted object-oriented language which functions as a network client to a state of the art, realtime sound synthesis server.

SuperCollider was written by James McCartney over a period of many years, and is now an open source (GPL) project maintained and developed by various people. It is used by musicians, scientists, and artists working with sound.

1 Hello SC !

The SuperCollider application makes use of client/server architecture which separates two functions, respectively one providing and the other requesting services. The client (sc-lang) and the server (sc-synth) communicate through a network.

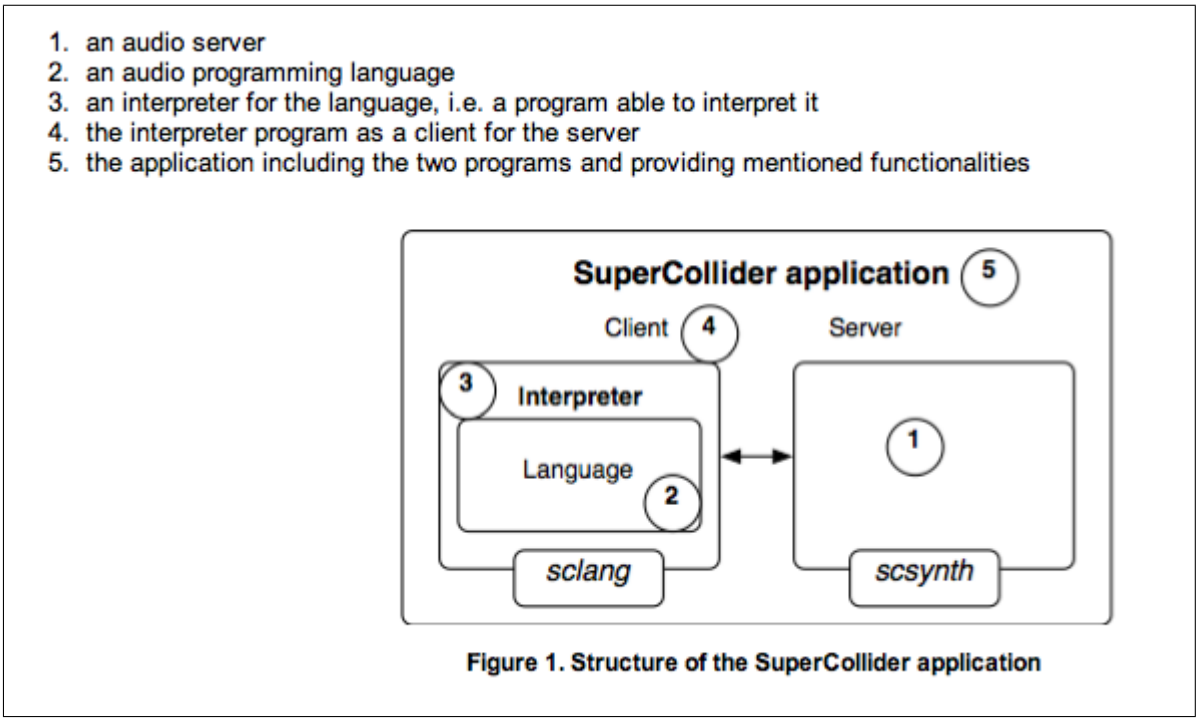

Figure 1: A screenshot from sc help file "Client vs Server"

1. http://supercollider.sourceforge.net/

2 Hello sc-community

Organize your environment in a manner to design your own instrument. SuperCollider users usually develop their own working environments for specific or general purposes.

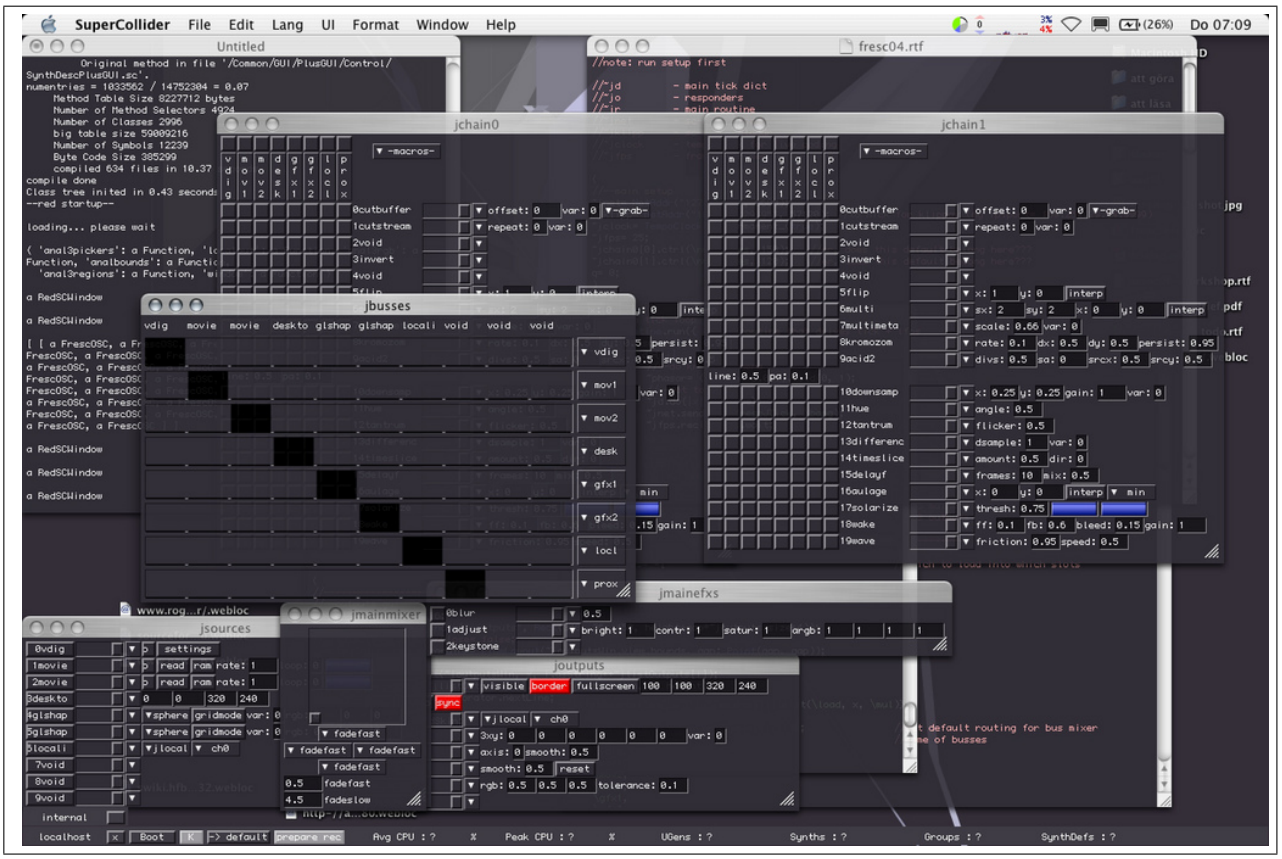

Figure 2: A screenshot by Dan Stowell.

You must be a social being, as long as you are a human. Try to follow sc-users mailing list, and find out more here: `http://supercollider.sourceforge.net/community/`

3 Hello World !

Type the line below and execute, from Menu > Lang > Evaluate Selection. Keyboard shortcut fn+return or ctrl-c or shift+return (mac), ctrl+E (gedit), C-c C-c (emacs)

```
"Hello World !".postln;
// this is a comment
/* ..and this is also a comment,
a multiple lines comment */
```

Now, it is time to listen to sc-synth. Your are about to listen in your left channel a sine oscillator at 440 Hz. Hello sound!

```
// the following command boots your server (sc-synth)
s.boot; // 's' is preassigned to your localhost server (s=Server.local;)
x = { SinOsc.ar(440, 0, 0.1) }.play;
x.free;
```

Find out how to use sc-help:

```
Help.gui; // or for keyboard shortcut press cmd+d (mac), F1 (win)
```

SuperCollider has one of the biggest and most active communities on computer music. You can find a lot of academic bibliography for a wide range of topics. Many support for the user is also provided by the Quarks library. A bunch of of code can be found by typing:

```
Quarks.gui;
```

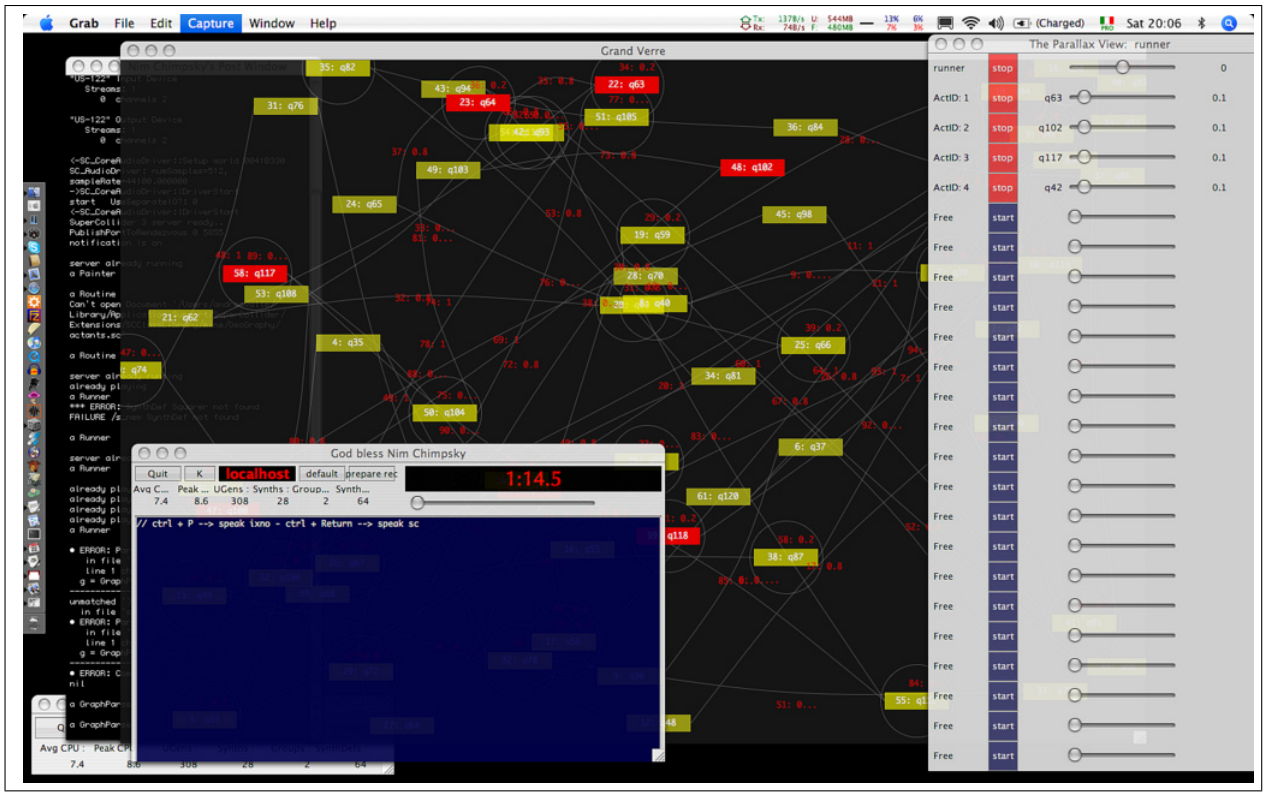

Figure 3: A screenshot by Andrea Valle, from Geography Quark.

4 Musical and research fields

SC is a programming environment for algorithmic composition. It is running on the meta-machine, well-known as personal computer.

Ron Kuivila back in 80s proclaimed (Collins 2011):

> "we have to make computer music that sounds like electronic music"

From 80s to today, a lot of progress have already done, although many people are convinced that there is a difference between software and hardware sound engines, and claim that this is audible to their ears.

On the other hand it is a common truth that the computers literally boost the capabilities for composition, interaction and performance. Using high-level languages such us sc3, it is possible to compose very long songs, or even infinite length songs, to train your musical robot using AI techniques, to do spatial sound applications and research, sound installations and more.

4.1 Live Coding

Another hot-topic these days is that of live coding (Collins et al. 2003). These high-level interpreted environments for sound programming like sc3, impromptu and others, reveals another perspective on musical interaction. The musician slowly turns himself to a "codician". Code is exposed in front of him as an instrument. Live coding moto is "show us your screens" [2] and there is a temporary organization for its promotion [3].

Figure 4: Toplap logo

4.2 sc140

Another interesting thing that proves the simplicity of SuperCollider is that the sc-users are used to send to each other code snippets through twitter. This constrain demands that you can type, up to 140 characters. The well known "sc140" [4] was the product

2. https://vimeo.com/20241649
3. http://toplap.org
4. http://thewire.co.uk/articles/3177/

of this interaction. Check some code below and listen to the full release here: `http://supercollider.sourceforge.net/sc140/`

```
// 06 Batuhan Bozkurt (refactored by Charles Celeste Hutchins)
f={|t|Pbind(\note,Pseq([-1,1,6,8,9,1,-1,8,6,1,9,8]+5,319),\dur,t)};Ptpar([0,
f.(1/6),12,f.(0.1672)],1).play//#supercollider reich RT @earslap

// 07 Thor Magnusson
play{x=SinOsc;y=LFNoise0;a=y.ar(8);(x.ar(Pulse.ar(1)*24)+x.ar(90(a*90)+
MoogFF.ar(Saw.ar(y.ar(4,333,666)),a*XLine.ar(1,39,99,99,0,2)))!2/3}
```

5 Time for coding

This section includes some interactive code to warm up your engines. Try to alter the examples below with care.

```
// open a frequency analyser
FreqScope.new;

// the line below gives you interaction with the mouse on x-axis
{LFSaw.ar(MouseX.kr(10, 1000, 1))}.play;

// keyboard shortcut to _stop_ sound!
// cmd + . (mac)
// esc (gedit)
// C-c, C-s (emacs)

// open your localhost levels
s.meter;

// check that your mic is working properly, else adjust it and s.reboot;
{DelayN.ar(SoundIn.ar, 1, 1)}.play // mic input with 1sec delay

// open server volume GUI slider
s.volume.gui;

// run this with care !! use your mic
{FreeVerb.ar(DelayN.ar(SoundIn.ar,1,1),0.5,0.6,1)}.play;

(
{
        var pitch = Pitch.kr(SoundIn.ar).poll;
        FreeVerb.ar(DelayN.ar(SinOsc.ar(pitch, 0, 0.1), 1, 1), 0.5, 0.5)
}.play(outbus:1);
)
```

Many thanks to Fredrik Olofsson for allow us to use and alternate the followings two code blocks.

```
(
f = {
        var amp = 0.3;
        var src = Amplitude.ar(DelayN.ar(SoundIn.ar, 1, 1), 0.01, 0.5);

        SinOsc.ar(Latch.ar(src, Impulse.ar(4)).poll.linexp(0, 1, 1200, 5000),
        0, src.lag(0.1)*amp)!2
        }.play;
)
f.free;

(
s.waitForBoot{
        Spec.add(\freq, #[20,2000, \lin]);
        Spec.add(\amp, #[0.0, 1.0, \lin]);

        Ndef(\sineGUI).play;
        Ndef(\sineGUI, { |freq=400, amp=0.1|
                Splay.ar(SinOsc.ar({|i| Stepper.kr(Amplitude.kr(SoundIn.ar).poll>0.1)
                *freq/(i+1)}.dup(8), 0, amp))
                });
        Ndef(\sineGUI).edit;
        s.meter;
};
)
```

6 Build your own working environment

You can find your startup file in the following path. When you open sc or recompile your library, the included code in the startup.scd file will be executed.

```
// check out the path of startup file
Platform.userConfigDir +/+ "startup.scd"

// open it in sc
(Platform.userConfigDir +/+ "startup.scd").asString.openDocument;
```

Try to put in your `startup.scd` file, the command `Server.local.boot;` in order to boot your server its time you recompile the class library.

7 Related help files

Highlight "ClassName" and press *cmd+J* , or Menu > Lang > Open Class Def. For methods, highlight "methodName" and press *cmd+Y*, or Menu > Lang > Implementations

```
Sclang Startup File
> Classes
        Client vs Server
        ServerOptions
        Platform
        UGen
                SinOsc
                LFSaw
                MouseX
                SoundIn
                Pitch
                FreeVerb
                DelayN
                Amplitude
                Impulse
                Latch
                Stepper
                Splay
                XLine
        FreqScope
        String
        Document

> Methods
        .postln
        .boot
        .play
        .meter
        .lag
        .linexp
        .dup
```

References:

Collins, N. 2011. Semiconducting – Making Music After the Transistor. In *"Technology and Aesthetics" Symposium* NOTAM, Oslo, May 2011

Collins, N., McLean, A., Rohrhuber, J., and A. Ward. 2003. Live coding in laptop performance. *Organised Sound*, 8(03): 321-330.

Wilson, S., Cottle, D. and N. Collins, eds. 2011. *The SuperCollider Book*. MIT Press.