WIKIPEDIA

SuperCollider

SuperCollider is an environment and programming language originally released in 1996 by James McCartney for real-time audio synthesis and algorithmic composition.[4][5]

Since then it has been evolving into a system used and further developed by both scientists and artists working with sound. It is an efficient and expressive dynamic programming language providing a framework for acoustic research, algorithmic music, interactive programming and live coding.

Originally released under the terms of the GPL-2.0-or-later in 2002, and from version 3.4 under GPL-3.0-or-later, SuperCollider is free and open-source software.

SuperCollider

Original author(s)	James McCartney
Initial release	1996
Stable release	3.11.2 / 15 November 2020[1]
Repository	github.com/supercollider/supercollider (https://github.com/supercollider/supercollider)
Written in	C++
Operating system	FreeBSD,[2] Linux, macOS, Windows
Type	Audio programming language
License	GPL-3.0-or-later[3]
Website	supercollider.github.io (http://supercollider.github.io)

Contents

Architecture

Starting with version 3, the SuperCollider environment has been split into two components: a server, *scsynth*; and a client, *sclang*. These components communicate using OSC (Open Sound Control).[6]

The SC language combines the object-oriented structure of Smalltalk and features from functional programming languages with a C-family syntax.[6]

The SC Server application supports simple C and C++ plugin APIs, making it easy to write efficient sound algorithms (unit generators), which can then be combined into graphs of calculations. Because all external control in the server happens via OSC, it is possible to use it with other languages or applications.[6]

The SuperCollider synthesis server (*scsynth*)

SuperCollider's sound generation is bundled into an optimised command-line executable (named *scsynth*). In most cases it is controlled from within the SuperCollider programming language, but it can be used independently. The audio server has the following features:[6]

- Open Sound Control access
- Simple ANSI C and C++11 plugin APIs
- Supports any number of input and output channels, including massively multichannel setups[7]
- Gives access to an ordered tree structure of synthesis nodes which define the order of execution
- Bus system which allows dynamically restructuring the signal flow
- Buffers for writing and reading
- Calculation at different rates depending on the needs: audio rate, control rate, demand rate

Supernova, an independent implementation of the Server architecture,[8] adds multi-processor support through explicit parallel grouping of synthesis nodes.

The SuperCollider programming language (*sclang*)

The SuperCollider programming language is a dynamically typed, garbage-collected, single-inheritance object-oriented and functional language similar to Smalltalk,[5] with a syntax similar to Lisp or the C programming language. Its architecture strikes a balance between the needs of realtime computation and the flexibility and simplicity of an abstract language. Like many functional languages, it implements functions as first-class objects, which may be composed. Functions and methods can have default argument values and variable length argument lists and can be called with any order of keyword arguments. Closures are lexical, and scope is both lexical and dynamic. Further features typical of functional languages are supported, including creation of closures via partial application (explicit currying), tail call optimization, list comprehensions, and coroutines. Specifics include the implicit expansion of tuples and the stateless pattern system. Its constant-time message lookup and real-time garbage collection allows large systems to be efficient and to handle signal processing flexibly.[6]

By supporting methods of reflective, conversational, and literate programming, SuperCollider makes it relatively easy to find new sound algorithms[9] and to develop custom software as well as custom frameworks. With regards to domain specific knowledge, it is both general (e.g., it allows the representation of properties such as time and pitch in variable degrees of abstraction) and has copious example implementations for specific purposes.[6]

GUI system

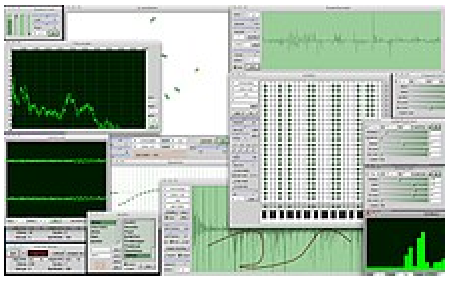
The SuperCollider language allows users to construct cross-platform graphical user interfaces for applications. The standard class library with user interface components may be extended by a number of available frameworks. For interactive programming, the system supports programmatic access to rich-text code files. It may be used to generate vector graphics algorithmically.[10]

Interfacing and system support

Clients

Because the server is controlled using Open Sound Control (OSC), a variety of applications can be used to control the server. SuperCollider language environments (see below) are typically used, but other OSC-aware systems can be used such as Pure Data.[6]

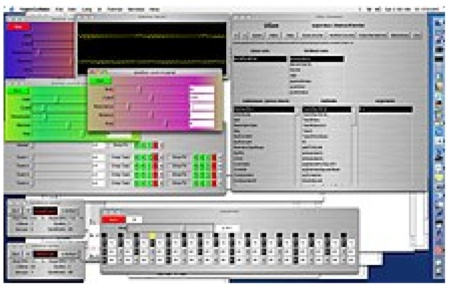
"Third-party" clients for the SuperCollider server exist, including rsc3, a Scheme client, hsc3, based on Haskell, ScalaCollider,[11] based on Scala, Overtone, based on Clojure, and Sonic Pi.[12] These are distinct from the development environments mentioned below because they do not provide an interface to SuperCollider's programming language, instead they communicate directly with the audio server and provide their own approaches to facilitating user expression.[6]


Screenshot of SuperCollider running the ixiQuarks GUI tools.

Supported operating systems

SuperCollider runs on macOS, Linux, Windows and FreeBSD. For each of these operating systems there are multiple language-editing environments and clients that can be used with SuperCollider (see below).[6]

It has also been demonstrated that SuperCollider can run on Android[13] and iOS.[14]

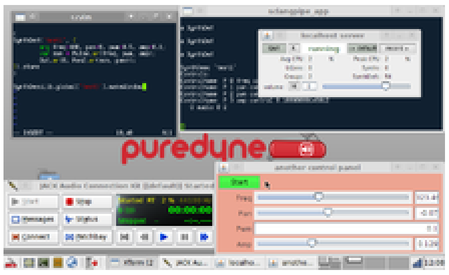

Screenshot of SuperCollider on Mac OS X with various user-generated GUI elements.

Editing environments

SuperCollider code is most commonly edited and used from within its own cross-platform IDE (which supports Linux, Mac, and Windows).

Other development environments with SuperCollider support include:

- Emacs (Linux, Mac, Windows)[15]
- Vim (Linux, Mac)
- Atom (Linux, Mac, Windows)[16]
- gedit (Linux, Windows)
- Kate (Linux)[17]



Screenshot of SuperCollider Vim on puredyne linux.

Code examples

```
// Print "Hello world!"
"Hello world!".postln;
```

```
// Play a mixture of an 800 Hz sine tone and pink noise
{ SinOsc.ar(800, 0, 0.1) + PinkNoise.ar(0.01) }.play;
```

```
// Modulate a sine frequency and a noise amplitude with another sine
// whose frequency depends on the horizontal mouse pointer position
{
    var x = SinOsc.ar(MouseX.kr(1, 100));
    SinOsc.ar(300 * x + 800, 0, 0.1)
    + PinkNoise.ar(0.1 * x + 0.1);
}.play;
```

```
// List iteration: multiply the elements of a collection by their indices
[1, 2, 5, 10, -3].collect { |elem, idx| elem * idx };
```

```
// Factorial function
f = { |x| if(x == 0) { 1 } { f.(x-1) * x } };
```

Live coding

As a versatile dynamic programming language, SuperCollider can be used for live coding, i.e. performances which involve the performer modifying and executing code on the fly.[18] Specific kinds of proxies serve as high level placeholders for synthesis objects which can be swapped in and out or modified at runtime. Environments allow sharing and modification of objects and process declarations over networks.[19] Various extension libraries support different abstraction and access to sound objects, e.g. dewdrop_lib[20] allows for the live creation and modification of pseudo-classes and pseudo-objects.

See also

- List of music software
- Comparison of audio synthesis environments

References

1. "Releases" (https://github.com/supercollider/supercollider/releases). *Github*. Retrieved 15 November 2020.
2. asynth. "SuperCollider" (http://sourceforge.net/project/showfiles.php?group_id=54622). Retrieved 20 June 2015.
3. "SuperCollider Licensing" (http://doc.sccode.org/Other/Licensing.html). Archived (http://web.archive.org/web/20200807141456/http://doc.sccode.org/Other/Licensing.html) from the original on 2020-08-07.
4. J. McCartney, SuperCollider: A new real time synthesis language (http://www.audiosynth.com/icmc96paper.html), in Proc. International Computer Music Conference (ICMC'96), 1996, pp. 257–258.
5. J. McCartney, Rethinking the computer music language: SuperCollider (https://dx.doi.org/10.1162/014892602320991383), Computer Music Journal, 26 (2002), pp. 61–68.
6. Scott Wilson; David Cottle; Nick Collins (2011). *The SuperCollider Book* (https://web.archive.org/web/20110501130100/http://supercolliderbook.net/#). The MIT Press. ISBN 978-0-262-23269-2. Archived from the original (http://supercolliderbook.net/) on 2011-05-01. Retrieved 2011-05-26.
7. "SuperCollider mailing lists" (https://web.archive.org/web/20091106111225/http://www.beast.bham.ac.uk/research/mulch.shtml). Archived from the original (http://www.beast.bham.ac.uk/research/mulch.shtml) on 6 November 2009. Retrieved 20 June 2015.
8. T. Blechmann, supernova, a multiprocessor-aware synthesis server for SuperCollider (http://lac.linuxaudio.org/2010/papers/32.pdf), Proceedings of the Linux Audio Conference, Utrecht 2010.

9. J. Rohrhuber, A. de Campo and Renate Wieser. Algorithms Today. Notes on Language Design for Just in Time Programming (http://www.wertlos.org/~rohrhuber/articles/AlgorithmsToday.pdf) Archived (https://web.archive.org/web/20110728172850/http://www.wertlos.org/~rohrhuber/articles/AlgorithmsToday.pdf) 2011-07-28 at the Wayback Machine. In *Proceedings of the International Computer Music Conference*, Barcelona, 2005.
10. The vector graphics interface is provided by the Pen class. Various examples can be found in Audiovisuals with SC (http://www.fredrikolofsson.com/f0blog/?q=node/316), blog by Fredrik Olofsson, 02.05.2009 (updated 11.05.2012)
11. Rutz, H. H. (2010). "Rethinking the SuperCollider Client...". *Proceedings of SuperCollider Symposium*. Berlin. CiteSeerX 10.1.1.186.9817 (https://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.186.9817).
12. "Systems interfacing with SC" (http://supercollider.sourceforge.net/wiki/index.php/Systems_interfacing_with_SC). Retrieved 20 June 2015.
13. SuperCollider Android project (https://github.com/glastonbridge/SuperCollider-Android/wiki) on GitHub
14. Tiny Music System (http://cylob.blogspot.co.uk/2009/11/tiny-music-system.html) - Cylob Blog, 04.11.2009
15. "SuperCollider with emacs: scel" (http://supercollider.sourceforge.net/wiki/index.php/SuperCollider_with_emacs:_scel). Retrieved 20 June 2015.
16. "supercollider" (https://atom.io/packages/supercollider). *Atom*. Retrieved 20 June 2015.
17. "jleben/Scate" (https://github.com/jleben/Scate). *GitHub*. Retrieved 20 June 2015.
18. Collins, N., McLean, A., Rohrhuber, J. & Ward, A. (2003), Live Coding Techniques for Laptop Performance, *Organised Sound* 8(3): pp 321-30. doi:10.1017/S135577180300030X (https://doi.org/10.1017%2FS135577180300030X)
19. J. Rohrhuber and A. de Campo. Waiting and uncertainty in computer music networks (http://akustik.hfbk.net/publications/Uncertainty_and_waiting.pdf). In *Proceedings of the International Computer Music Conference*, Miami, 2004.
20. One of the numerous user contributed libraries known as "Quarks", and published in the SuperCollider Quarks repository (https://github.com/supercollider-quarks).

External links

- Official website (http://supercollider.github.io)

Retrieved from "https://en.wikipedia.org/w/index.php?title=SuperCollider&oldid=1026488327"