Workshop: Synthesis and Performance in SuperCollider

H. James Harkins

March 23, 2014

Contents

1. Preface

1.1. Preface

Project history
In autumn 2013, 张睿博 (Zhang Ruibo, Shenyang Conservatory and head of CHEARS, the China ElectroAcoustic Resource Survey) approached me to present a week-long introductory SuperCollider workshop in April 2014. I wrote this material in the first couple of months of 2014.

Overview
This document includes all the presentation slides from the workshop, plus further explanation in text, an Appendix with additional topics (interesting, but too much for one week!), and an index and glossary of terms. The material serves two purposes: first, to guide the workshop sessions (which don't need to follow the slides exactly), and second, to provide documentation which CHEARS may translate into Chinese.

The workshop aims to simplify individual units of code by imitating *modular* synthesis in code. This approach avoids several of the trickier challenges of learning SuperCollider, related to code structure and server architecture. It is, however, a new approach; "canonical" SuperCollider style is different in some important ways.

Why not simply teach the canonical style? When approaching any programming language, small code blocks are easier to understand, while larger ones, even ten lines, may be scary at first. Canonical style binds many synthesizer components into one self-contained unit called a `SynthDef`; dividing it into more digestible pieces requires the new user to understand details of the server that are likely to be confusing at first. The modular style presented here allows each module to handle one small piece of synthesis *and* relieves the new user from the burden of understanding advanced concepts too early.

This modular style depends on some new *extensions* to the language. Installation instructions are in Part I. The specific extensions are:

- Improved handling of modules that represent constant numbers.
- A modifier, `\psSet`, to act on many modules at once using a central code interface.

Some of these may be incorporated in future SuperCollider versions. At that time, the workshop extensions package will have to change.

Typographical conventions
Code examples appear in a monospaced sans-serif font, Inconsolata:[1] `Server.local.boot`.

Keyboard shortcuts are boxed: `Ctrl-Return`.

Numbered code examples are collected into `scd` files. Each Part has two collections: one with all examples, and the other with only selected examples. I strongly recommend using the selected examples only. You will learn more by typing the examples yourself, instead of passively executing them.

The document is written using *org-mode* 8.2.3[2] running in Emacs 23, and typeset using LaTeX (*xelatex* from TexLive 2012[3]) with *Beamer* extensions.[4]

Copyleft and licensing
This document and all associated materials are released under the Creative Commons Attribution-ShareAlike 4.0 license.[5] You may use these materials as the basis of a new workshop or course, if you:

- Give credit to me as the initial author (Attribution);
- Release your materials under a similar (more permissive) license (ShareAlike). That is, you should allow others to modify your version and publish their version.

You may not lift large passages of this work and claim them as your own, or publish them under a copyright-style license that forbids further modification. This work is meant to be part of a conversation about art and programming. Copyright stops the conversation, so I don't want it.

Acknowledgments
I won't attempt a complete list of thanks, but here's a good start:

- The entire community of SuperCollider users and developers, without whom, none of this. In particular:
  - James McCartney, for inventing the thing;

[1] `http://www.levien.com/type/myfonts/inconsolata.html`
[2] `http://orgmode.org`
[3] `http://tug.org/texlive/`
[4] `https://bitbucket.org/rivanvx/beamer/wiki/Home`
[5] `http://creativecommons.org/licenses/by-sa/4.0/`

  - Julian Rohrhuber and Alberto de Campo, for considering improvements to JITLib that make this modular-synthesis usage much easier.

- The *org-mode* development team, for a spectacularly customizable authoring tool (plus calendar, to-do list and in-general everything tool).
- Users of `http://tex.stackexchange.com` for advice on a few sticky LaTeX problems.